Linux

Base d'administration pour le superutilisateur – Exposé.

Adaptation de Rute User's Tutoriel and Exposition

de Paul Sheer.

Thierry Lepoint.
Institut Supérieur d'Electronique et du Numérique
41, boulevard Vauban, 59046 Lille Cedex (France).

8 juillet 2006

Droits et devoirs d'utilisation.

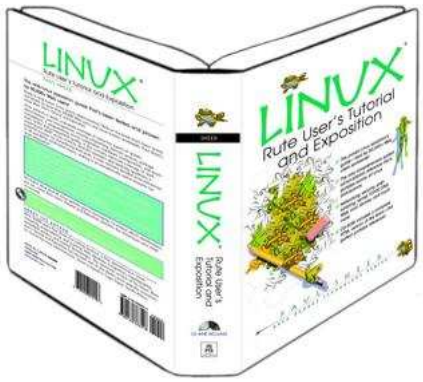
Sommaire.

Préface de l'édition originale.

Lorsqu'en 1994 j'ai commencé à travailler avec GNU/Linux, je venais directement du monde DOS. Bien qu'Unix me fût peu familier, les livres traitant de Linux supposaient que toute personne utilisant ce système d'exploitation avait migré des systèmes V ou BSD dont je n'avais jamais entendu parler. C'est une entreprise délicate que de créer, pour certains de partager, la méthode dont vous aimeriez disposer. Je ne suis pas convaincu que, même à l'heure actuelle, un texte unificateur existe à côté du présent ouvrage. Quoiqu'il en soit, je vous propose celui dans un état que je juge malheureusement incomplet.

J'espère que le lecteur aura un texte unique pour le guider parmi toutes les facettes du système GNU/Linux.

Paul Sheer.



Préface de la traduction française.

Bien des introductions au système GNU/Linux existent. Depuis mes premiers pas avec LINUX (octobre 2001) et comme pour beaucoup d'utilisateurs, le livre de Paul Sheer (qui peut être consulté sur http ://www. icon.co.za/~psheer/-book/) m'a donné l'occasion d'avoir un texte regroupant de nombreux aspects souvent disséminés dans plusieurs ouvrages. Son but est d'aider à maîtriser le système GNU/LINUX au point de passer une habilitation Linux. Il vient à point pour compléter un autre livre qui ne quitte pas mon bureau : "*Le système Linux*" de Matt Welsh, Matthias Kalle Dalheimer et Lar Kauffman publié chez O'Reilly (http ://www.oreilly.fr/). J'espère de tout coeur que la traduction française de Rute User's Tutorial and Exposition apportera une porte d'entrée supplémentaire dans le monde de GNU/LINUX. Tout lecteur qui souhaite aborder des aspects plus théoriques et techniques d'administration, de réseaux, de sécurité, etc. aura une base extrêmement solide pour poursuivre.

La présente traduction peut être grandement améliorée. Aussi, serai-je reconnaissant à toute personne me signalant des erreurs ou me transmettant des conseils (thierry.lepoint@tiscali.be).

Thierry Lepoint

Elouges, le 15 juillet 2004

Remerciements

Mes remerciements vont à Abraham van der Merwe, rapporteur, et à Jane Bonnell, mon éditeur. Je tiens aussi à remercier Jonathan Maltz, Jarrod Cinman et Alan Tredgold pour m'avoir introduit dans le monde GNU/Linux dès 1994. Ma reconnaissance va aussi aux développeurs de la Free Software Foundation qui développèrent LaTeX, TeX, GhostScript, GhostView, Autotrace, XFig, XV, Gimp, la police de caractères Palatino, les différents styles d'extension de LaTeX, DVIPS, DVIPDFM, ImageMagick, XDVI, XPDF, et LaTeX2HTML, sans lesquels ce document eût été très difficilement réalisable. Pour en nommer quelques-uns : John Bradley, David Carlisle, Eric Cooper, John Cristy, Peter Deutsch, Nikos Drakos, Mark Eichin, Brian Fox, Carstein Heinz, Spencer Kimball, Paul King, Donald Knuth, Peter Mattis, Frank Mittelbach, Ross Moore, Derek B. Noonburg, Johannes Plass, Sebastian Rahtz, Chet Ramey, Tomas Rokicki, Bob Scheifler, Rainer Shoepf, Brian Smith, Supoj Sutanthavibul, Herb Swan, Tim Theisen, Paul Vojta, Martin Weber, Marck Wicks, Masatake Yamato, Ken Yap, Herman Zapf.

Merci à Christopher R. Hertel pour avoir contribué à l'introduction de Samba.

Mes remerciements les plus vifs au projet GNU de la Free Software Foundation, aux développeurs innombrables de la Free Software, et aux nombreux lecteurs qui ont donné un retour précieux sur le site web : http ://www.icon.co.-za/~psheer/book.

Paul Sheer

Remerciements de la traduction française

Le traducteur se joint à l'auteur pour l'immense gratitude qu'il adresse à la communauté du Libre. Il adjoint ses profonds remerciements à l'équipe de LyX et de LaTeX. Il exprime sa plus profonde gratitude à Andrew Szeri (Université de Berkeley, CA, USA) et Jean-Marc Lévêque (Université de Savoie, France) pour l'avoir introduit dans le monde de GNU/Linux, à Stéphane Wirtel (ancien Linux-Mons) pour ses nombreux conseils, ainsi qu'aux membres du forum Unixtech.be (http ://www.unixtech.be) pour leur aide et les excellentes discussions.

Je tiens aussi à exprimer ma sincère reconnaissance aux relecteurs : Esteban Dugueperoux et, en particulier, Eric Gerbier.

Par ailleurs, remerciera-t-on jamais assez Linus Torvalds, Richard Stallman, leurs collaborateurs et tous ceux qui contribuent au développement des logiciels libres et de GNU/Linux, en particulier ?

Thierry Lepoint

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

GNU Linux et les logiciels libres*.[1]

1.1 Historique : UNIX, MINIX et LINUX.

- Dans les années '60, un système CTTS (*Compatible Time Sharing System*) a vu le jour afin d'exploiter les rares et coûteuses ressources informatiques. En 1965, le développement d'un système Multics (*MULTiplexed Information and Computing Service*) basé sur CTTS et destiné à fournir par réseau un ensemble très vaste d'applications –y compris domotiques, s'arrête devant l'ampleur de la tâche. En 1969, Ken Thompson et Dennis Ritchie demandent aux Bell Laboratories (AT&T, une entreprise monopolistique de l'Etat américain) que leur équipe ne soit pas dissoute. Ils disposent d'un PDP-7 de DEC avec 4 Ko de RAM et, s'adaptant à la maigreur des ressources, ils développent un système d'exploitation (*Operating System*) pour mini-ordinateurs afin de fournir aux opérateurs Bell un environnement multi-tâche et multi-utilisateur auquel ils ont goûté avec Multics. L'objectif central est le partage de ressources informatiques onéreuses. Au départ, le langage utilisé est l'*assembler*. Puis, le BCPL developpé en 1967 par Martin Richards fait place au langage B créé par K. Thompson.
- En 1970, Kernighan suggère le nom Unics, par opposition à Multics. Le langage B se transforme en C et, Unics (*UNIplexed Information and Computing Service*) se transforme en UNIX.
- De 1972 à 1973, une nouvelle version d'UNIX est traduite en langage C, inventé par D. Ritchie. En principe, le système est portable. AT&T réalise qu'UNIX peut devenir un outil de production.
- En 1974, K. Thompson part enseigner à Berkeley. Bill Joy, un étudiant, s'intéresse aux travaux de son professeur. Il inventera un autre UNIX, le système BSD (*Berkeley Software Development*). La version 4 d'UNIX est offerte à l'Université de Berkeley (CA). Une divergence s'installe entre l'UNIX d'AT&T et le système BSD, mais le succès d'UNIX devient rapidement extraordinaire.
- Alors que le protocole TCP/IP (*Transmission Control Protocol/Internet*

[1] Les chapitres ou sections additionnés au cours de la traduction sont marqués d'un astérisque.

Protocol) est développé, K. Thompson et D. Ritchie réécrivent une version portable d'UNIX entre 1977 et 1979.
- En 1980, les premières licences d'UNIX System V sont délivrées aux constructeurs. IBM lance son PC 5150.
- En 1984, les fabricants informatiques ayant adopté UNIX créent "X/Open" pour arriver à une normalisation. Le *Massachussets Institute of Technology* (MIT) développe le système de multi-fenêtrage graphique X-Window indépendant d'UNIX. L'idée d'une exploitation libre des logiciels sous une forme structurée naît et la *Free Software Foundation (*FSF) est créée sous l'impulsion de Richard Stallman.
- En 1987, AT&T (propriétaire d'UNIX) et le fabricant américain de stations de travail Sun utilisant BSD concluent une alliance.
- La même année, dans le cadre d'un projet universitaire Andy Tanenbaum crée un clône d'UNIX -nommé MINIX- fonctionnant sur PC et dont le but est pédagogique. Les périphériques pris en charge sont trop peu nombreux pour que MINIX passe dans le monde professionnel.
- En 1988, l'OSF (*Open Software Foundation*) regroupant DEC, HP, IBM, etc. normalise UNIX en OSF1. UNIX International regroupant AT&T, Sun et autres propose UNIX System V.
- En 1991, Linus Torvalds, étudiant à l'Université d'Helsinki, souhaite utiliser un OS non-propriétaire pour son PC équipé d'un processeur 80386. Il s'inspire de MINIX et écrit un premier code en *assembler*. De là, naissent des discussions sur l'internet via *comp.os.minix*. Le 5 octobre 1991, une première version officielle 0.02 est annoncée, puis la 0.03 et directement, la version 0.10. Le code est transposé en langage C. Le noyau LINUX (LINUs uniX) ne comprend pas de code d'UNIX commercial.
- Trente ans après la création d'UNIX, K. Thompson et D. Ritchie sont décorés par le Président américain B. Clinton. Vers le début 2000, grâce au développement de LINUX et des programmes GNU, le public et le marché redécouvrent la supériorité technique d'UNIX.

Le noyau et les programmes sont sous la licence GPL (*General Public Licence*) ce qui permet d'avoir un noyau UNIX totalement libre[2]. La plupart des programmes sont conçus dans le cadre du projet GNU de la FSF (*GNU is Not Unix* est un acronyme récursif). LINUX est désormais un vrai système 32 ou 64 bits, multi-tâches, multi-utilisateurs, réseau et complet. Il supporte de nombreuses architectures (Alpha, Compaq, Digital, PC, PPC, SPARC, etc.). Il effectue tout ce qu'on peut attendre d'un ordinateur moderne en travaillant en réseau, en grappes [pour le calcul en parallèle dit HPC (pour *High Performance Computing*) et pour l'augmentation de disponibilité de services, c'est-à-dire HA (*High Availability*)]. Par ailleurs, LINUX supporte la gestion de super-ordinateurs dotés de plusieurs centaines voire milliers de processeurs fonctionnant en *symmetric multiple processing*. Par exemple, au Lawrence Livermore National Laboratory (USA), en 2002, Linux-Xeon sur un super-ordinateur de 2304 processeurs fonctionnant à 2,4 GHz parvient à réaliser ~5,7 GFlops (5,7 milliards d'opérations en virgule flottante par seconde) et 11 GFlops en crête.

[2] Il est utile de souligner la différence entre logiciels libres et logiciels "Open Source". Un logiciel est libre si ses utilisateurs peuvent disposer des codes sources, ce qui signifie que les codes sont transformables. Les logiciels "Open Source" peuvent être diffusés librement (et éventuellement gratuitement) mais leur code source n'est accessible qu'à certaines conditions, voire inaccessible.

Pour le particulier, les entreprises ou les administrations, il transforme un simple PC en une station de travail en réseau. A titre d'exemples, il s'acquitte de travaux de bureautique classique, de bureautique assistée (LyX basé sur LaTeX, TeXMacs), de présentations, de calculs par tableurs (OpenOffice), de mise en graphique rapide par GNU-plot ou MathPlot, de calculs symboliques (Maxima/Xmaxima, Scilab), de comptabilité (GNUCash), de programmation (C, C++, Fortran, Java, Perl, Ada, Objective, TCL/TK, GTK, etc), de multimedia (lecture/gravure de films, photoscope, DVD, CD, MP3, Ogg, webcam, scanner –pour autant que ces matériels soient reconnus[3]). Linux permet d'utiliser les messageries (courriels et échanges en temps réel, IRQ et ICQ), divers outils web -navigateurs, composeurs HTML, PHP. Il gère des bases de données comme Oracle, PostgreSQL et MySQL. Dans une entreprise, une école ou un Ministère, il sert de serveur de messagerie, de services internet, etc. Il fait tourner des programmes natifs de MS-Windows (win4lin, WineX, etc) et présente de remarquables et puissants outils d'administration locale, d'administration réseau, etc. Linux est très largement utilisé comme routeur et comme serveur en raison de sa remarquable stabilité avec des programmes fameux comme Apache, Postfix, etc.

Au fil du temps, deux organismes de normalisation ont émergés pour définir une norme internationale :

- le groupe X/Open (1984) a pour but d'améliorer l'environnement du système, de donner un guide de portabilité pour les concepteurs d'application et de produire des utilitaires nouveaux,
- le groupe POSIX (*Portable Open System Interface eXchange*) fait partie de l'IEEE (*Institute of Electrical and Electronic Engineers*) et est constitué de sous-groupes rédigeant des guides, langages et extentions pour Unix.

Les travaux du groupe POSIX l'ont emporté. Il est important de noter que les administrations exigent la norme POSIX à laquelle Linux répond.

A côté des organismes travaillant sur les normes, des groupes d'utilisateurs constituent des fora Unix, Linux ou BSD. Ce dernier OS n'existe pas sous forme de distributions mais est accessible seulement par téléchargement (FreeBSD, NetBSD et OpenBSD). Plusieurs associations francophones ont vu le jour :

- l'AFUU : créée en 1982, cette Association Française des Utilisateurs Unix et des systèmes ouverts regroupe des chercheurs, des informaticiens, des ingénieurs et a pour objectif de promouvoir la "culture Unix" (convention Unix, séminaires, conférences, publication de revues, de périodiques, etc.). Elle est constituée en groupe de travail : groupe Sécurité, groupe Portabilité, groupe Réseaux...
- l'AFUL : association à but non-lucratif (loi du 1er juillet 1901). L'Association Francophone des utilisateurs de Linux et des Logiciels Libres cherche à promouvoir les logiciels libres, principalement ceux basés sur la

[3] Quatre remarques : (i) le noyau Linux évolue sans cesse en intégrant de nouveaux pilotes pour périphériques (les développeurs doivent être salués au passage), (ii) les constructeurs appréhendent de mieux en mieux la progression de Linux et s'emploient à développer des pilotes, (iii) ce n'est pas parce que la publicité d'un produit ne le mentionne pas explicitement que ce dernier est d'office incompatible avec Linux (certains fabricants sont moins rapides que d'autres). (iv) Il est judicieux de consulter les sites suivants pour s'assurer de la compatibilité ou d'envoyer un courriel à un fabricant, trop heureux de pouvoir vous vendre du matériel compatible. Sites : `http://www.freinix.fr/unix/linux/HOWTO/Hardware-HOWTO.html` ; `http://www.linuxhardware.org` ; `http://lhd.datapower.com`)

norme POSIX qu'exigent les administrations publiques. Le fonctionnement se fait par groupe. Les LUGs (LINUX *User Groups*) appelés en français GUL (Groupe d'utilisateurs LINUX) y sont associés. Les GUL, organisés par régions, sont une grande source d'échange pour les novices qui ainsi réalisent rapidement de grands progrès, et pour les éléments confirmés qui travaillent au développement du système ou à son administration.

1.2 Les logiciels libres et la licence publique générale GPL.

La Free Software Foundation est une organisation ayant pour but de développer des logiciels libres. Que recouvre le terme "libre" ?

- La GPL (*General Public Licence*, voir le chapitre 50) garantit aux utilisateurs des logiciels qu'elle protège, la liberté d'utiliser, de partager, de modifier et de redistribuer ces derniers. Il s'agit des quatre libertés fondamentales caractérisant un logiciel libre. La GPL garantit qu'ils sont accessibles à tout utilisateur. Lorsqu'un logiciel est placé par son auteur sous licence GPL, personne ne peut refuser à un utilisateur de se servir de ce logiciel. Personne ne peut demander à un utilisateur de renoncer à distribuer des copies ou à modifier le contenu des codes sources, que les logiciels concernés soient gratuits ou payants. *De facto*, la distribution libre permet d'accéder à des logiciels gratuits.
- Par ailleurs, il faut protéger les auteurs et la FSF. Aussi n'y a-t-il pas de garantie sur les logiciels. En effet, si un tiers modifie et redistribue un logiciel déposé par un auteur, la GPL garantit à ceux qui reçoivent le programme qu'il ne s'agit pas de l'original de sorte que, si une copie défectueuse est transmise, la réputation de l'auteur initial ne peut être entâchée.
- La GPL protège les logiciels libres des dépôts de brevets introduits postérieurement à leur développement, garantit le libre usage des logiciels et s'oppose donc aux licences propriétaires.

La GPL a été développée par la FSF pour le projet GNU. Les logiciels GNU ne contiennent pas d'éléments provenant des UNIX commerciaux. Par exemple, il existe des compilateurs C et C++ commerciaux. La GPL protège des compilateurs GNU-C, GNU-C++ tout comme le logiciel GNU-Emacs (Emacs étant un des éditeurs les plus puissants qui existent) et des dizaines de milliers de logiciels. Il faut noter que les programmes libres sont écrits et peaufinés par des passionnés. Ceci garantit la créativité et l'amélioration de la qualité en matière d'informatique. Dans le cas de LINUX, la rapidité et l'efficacité du débogage est sans équivalent. Du coup, compte tenu de la qualité de l'architecture générale du système et de la qualité des logiciels qui le constituent, LINUX est un des meilleurs systèmes qui soient avec BSD.

Chapitre 2

Introduction

Tandis qu'avec d'autres livres vous risquez de n'avoir que les pieds mouillés, celui-ci vous laisse d'abord barboter un peu, vous pousse la tête sous l'eau et vous alimente en oxygène.

2.1 Quelle est la matière traitée dans ce livre ?

Ce livre couvre l'administration système de GNU/Linux pour des distributions bien connues telles que RedHat ou Debian, sous forme d'un tutoriel pour les débutants et d'une référence pour les administrateurs avancés. Sa caractéristique est d'être concis tout en utilisant des explications et des exemples pratiques pour chaque aspect d'un système Unix. Chacun qui souhaite avoir un texte très complet sur ce qui est communément appelé "Linux", ne doit pas aller beaucoup plus loin. Très peu de choses sont omises ici.

2.2 Comment utiliser ce cours ?

L'ordre des chapitres a été attentivement élaboré de manière à effectuer une lecture séquentielle. Vous devriez lire depuis le début vers la fin, de sorte que certains chapitres ne fassent pas référence à des notions qui vous soient inconnues. De nombreux exemples rehaussent le texte et permettent de s'entraîner au cours de la lecture.

2.3 Que faut-il pour débuter ?

Un système Linux doit être installé sur votre ordinateur. Beaucoup de revendeurs proposent des CD-ROMs correspondant à diverses distributions.[1] Une astuce : essayez d'installer un maximum de paquets de sorte que, quand un paquet logiciel est mentionné, vous puissiez l'utiliser immédiatement.[2]

[1] NdT : Ikarios –qui ne vend que des CD/DVD de logiciels libres, des livres ou des objets se rapportant au logiciel libre– est recommandé en raison de son sérieux : http ://ikarios.com

[2] NdT : il est possible de télécharger des images .iso à graver et de se rapprocher d'un GUL (Groupe d'utilisateurs Linux) où de nombreuses personnes très compétentes sont prêtes à vous aider.

2.4 Un livre, un cours.

Le chapitre 17 contient une liste très détaillée des données disponibles pour votre système. Le document que vous êtes en train de lire complémente ces données et propose un tutoriel à la fois complet et indépendant de toute connaissance UNIX préalable. Il a également pour vocation de servir de notes pour tout cours de formation GNU/LINUX. Par exemple, en Afrique du Sud, Paul Sheer –l'auteur de l'édition originale du présent livre– utilise les premiers chapitres comme une partie d'un cours de 36 heures sur GNU/LINUX (12 séances). Les détails du programme de ce cours sont donnés dans le chapitre 46.

Notez que tous les "LINUX" sont composés très largement de logiciels GNU. A partir de maintenant, nous ferons référence, comme beaucoup le font improprement, à LINUX plutôt qu'à GNU/LINUX.

2.5 La documentation que je ne comprends pas.

Toute référence requerra que vous la lisiez trois fois au moins avant d'avoir une idée raisonnable de ce qu'il faut faire. Si vous devez la lire plus de trois fois, cela veut dire que vous avez besoin d'une information qu'il aurait fallu acquérir au préalable. Si vous lisez un document une seule fois, alors c'est que vous êtes impatient.

Il est important que vous compreniez les termes exacts utilisés. Essayez toujours de capter le sens précis d'un mot nouveau avant de poursuivre. Par ailleurs, ce n'est pas une bonne idée d'apprendre de nouvelles notions en se mettant des dates-butoirs. Au lieu de se faire sous pression, votre apprentissage de LINUX devrait s'élaborer avec aisance et une certaine dose de fascination.

2.6 L'Institut des Professionnels de LINUX (LPI) et les conditions pour devenir un ingénieur certifié "Red-Hat" (RHCE)

La différence qu'il y a entre passer un examen et faire quelque chose d'utile est assez grande. Le LPI et le RHCE sont deux certifications relatives à LINUX. A beaucoup d'égards, ce cours couvre bien davantage d'aspects que les deux certifications réunies mais, en certaines occasions, il laisse des points mineurs sous forme d'exercices. Son contenu couvre largement les connaissances nécessaires pour réussir ces certifications.

Les connaissances requises par le LPI et RHCE sont données aux chapitres 47 et 48.

Ces deux certifications sont en réalité des introductions à LINUX. Pour obtenir ces certifications, les utilisateurs ne sont pas tenus d'écrire des scripts de shell afin de réaliser des opérations délicates, ni de comprendre les caractéristiques subtiles et complexes de nombreux périphériques standards, ni d'être capables de saisir seul un nombre énorme d'applications utiles non-standard.

Pour être brutal : quelqu'un peut passer ces certifications et toujours être considéré comme incapable selon les critères des sociétés faisant de l'intégration système. [L'intégration système est un terme propre à l'auteur. Il désigne l'utilisation de LINUX pour avoir accès à des fonctions non-triviales telles qu'écrire

des scripts de shell complexes, mettre en place des réseaux de grande taille, interfacer des bases de données, le web et le courriel.] En fait, ces certifications ne concernent pas la programmation.

2.7 Pas "RedHat" mais "de type RedHat"

Dans ce livre-ci, nous faisons référence à "RedHat" ou "Debian".[3] Nous voulons dire que ce sont des systèmes utilisant des paquets `.rpm` (*r*edHat *p*ackage *m*anager) par opposition à ceux basés sur des paquets logiciels `.deb`. Ceci signifie qu'il n'y a aucune raison d'éviter une distribution comme Mandriva, dont le système de paquets est jugé meilleur que celui de RedHat.

En bref, les noms commerciaux n'ont plus grande signification pour le logiciel libre. Notez que la même remarque s'applique aux systèmes UNIX, un terme qui désigne le dénominateur commun de toutes les variantes, dont RISC,[4] les *mainframes* (ordinateurs centraux), et les variantes System V ou BSD.

2.8 Mises-à-jour et errata.

Les corrections de l'édition anglaise sont postées à l'adresse suivante : `http://www.icon.co.za/~psheer/rute-errata.html`. Verifiez cette page web avant de mentionner une erreurse rapportant à l'édition anglaise.

[3] NdT : le lecteur se référera à l'ouvrage en français : R. Hertzog " *Debian*" , Eyrolles, Paris, 2004 et aussi à la formation d'Alexis de Lattre (voir http ://people.via.ecp.fr/~alexis/formation-linux/formation-linux.html).

[4] NdT : *Reduced Instruction Set Computer*. Il s'agit d'ordinateurs à jeu d'instruction réduit, plus rapide que les CISC car on peut mieux optimiser leur câblage, et parce qu'ils ont des vitesses d'horloge plus importantes, leur petite taille et leur faible nombre de transistors les faisant moins chauffer. CISC : ordinateurs à jeu d'instructions complexes (cas de tous les ordinateurs communs durant les années 1980, ainsi que les PC des années 1990).

Chapitre 3

Les bases

Ce chapitre décrit des bases d'informatique familières à de nombreux utilisateurs. La meilleure manière d'appréhender la méthode qu'utilise un ordinateur pour effectuer des enregistrements et pour gèrer l'information consiste à se demander comment vous feriez personnellement. Très souvent, le mode de fonctionnement d'un ordinateur est similaire à la méthode que vous utiliseriez vous-même pour traiter le même problème la première fois. Les seules limitations sont la faisabilité logicielle et l'imagination. Pour le reste, tout est permis.

3.1 Bases binaire, octale, décimale et hexadécimale.

Nous avons tous appris à compter sur nos dix doigts.[1] La plupart des nombres que nous rencontrons sont représentés à l'aide de la base 10. Les codes postaux anglais utilisent des lettres et des chiffres de manière à être représentés en base 36 (26 lettres et 10 chiffres). La base la plus simple est la base 2 qui ne connaît que les chiffres 0 et 1. Un numéro de téléphone constitué de 7 chiffres permet $\underbrace{10 \times 10 \times 10 \times 10 \times 10 \times 10 \times 10}_{7\,chiffres\,(base\,10)} = 10^7$combinaisons. Un code postal à 4 caractères en base 36 permet 36^4combinaisons tandis qu'un nombre binaire de 8 chiffres n'autorise que 2^8 ou 256 combinaisons.

Etant donné que les ordinateurs fonctionnent en base 2 et qu'il est fastidieux de faire des conversions base 2 $\leftrightarrow$ base 10, les informaticiens en sont venus à utiliser les bases 8 (octale) et 16 (hexadécimale). Les nombres hexadécimaux contiennent des chiffres de 0 à 9 et des lettres de A à F. Le terme hexadécimal est souvent réduit à hex. La base octale n'utilise que des chiffres de 0 à 7.

Considérez un nombre binaire à 4 chiffres (4 *digits*) : il y a $2^4 = 16$ combinaisons possibles. En base hexadécimale, il n'y a qu'une combinaison pour ce même nombre. Par ailleurs, un nombre binaire à 3 chiffres ($2^3 = 8$) ne présente plus qu'une combinaison en base octale. Comme indiqué dans les tableaux 3.1 et 3.2, un nombre binaire peut être représenté en octal ou hex de manière simple.

[1] Le terme "digit" veut dire "doigt" en latin.

TAB. 3.1 – Représentation binaire et octale.

Binaire	Octale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

TAB. 3.2 – Représentation binaire et hexadécimale.

Binaire	Hexadécimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Un nombre binaire comme 01001011 peut être représenté en hex par 4B et en octal comme 113 en le décomposant en 2 groupes de 4 chiffres pour le cas de la base hex et en 3 groupes de 3 chiffres pour la base octale.

Pour l'administrateur et le programmeur, il y a souvent une ambiguïté à déterminer si un nombre est représenté en base hexadécimale, décimale ou octale. Par exemple, le nombre hex 56 s'exprime comme 01010110, le nombre octal 56 est 101110 et le nombre décimal 56 se représente 111000 (après de fastidieuses conversions). Afin de permettre la distinction, les nombres hex sont préfixés avec "0x", les nombres octaux sont préfixés avec "0" et si le premier nombre commence par un nombre de 1 à 9, il s'agit d'un nombre décimal dans la plupart des cas. Nous écrirons donc 0x56 en hex, 056 en octal et 56 en décimal. Une autre représentation consiste à àjouter la lettre H, D, O ou B (ou h, d, o ou b) au nombre pour indiquer sa base.

UNIX fait un usage fréquent des nombres binaires à 8, 16 ou 32 chiffres en les

représentant souvent sous forme de nombres hex à 2, 4 ou 8 chiffres. Vous vous habituerez à voir des nombres comme 0xffff (ou FFFFh) qui représente 65535, exprimé en binaire comme 1111111111111111.

3.2 Fichiers.

Le fichier est un composant de base de tous les ordinateurs. Il contient un seul bloc continu de données. Toute donnée –quelle que soit son type– peut être contenue dans un fichier et, il n'y a pas de donnée qui ne puissent y être stockées. Par ailleurs, il n'y a pas de donnée qui soit stockée ailleurs que dans un fichier. Un fichier contient des données d'un même type ; par exemple, une image est enregistrée dans un seul fichier. Durant sa préparation, ce livre a été enregistré dans un seul fichier. Il est peu commun que différents types de fichiers (par exemple un texte et des images) se trouvent ensemble dans le même fichier car cela n'est pas très commode. Un ordinateur peut contenir typiquement de l'ordre de 10.000 fichiers très variés. Chaque fichier possède un nom. Un nom de fichier sur une machine LINUX ou UNIX peut comporter jusqu'à 256 caractères.

Usuellement, le nom de fichier est explicite. Vous pourriez nommer une des lettres de votre correspondance à une amie ainsi : `Mary_Jones.lettre`. A partir de maintenant, si vous voyez une police de caractères `celle-ci`, il s'agira de mots pouvant être interprétés hors écran. Le nom que vous choisissez n'a pas d'importance du point de vue de l'ordinateur et il pourrait être une autre combinaison de chiffres et de lettres. Cependant, vous ferez référence à ces données-là à l'aide de ce nom de fichier-là lorsque vous souhaiterez traiter ce fichier. Aussi, préférerez-vous que le nom de fichier soit aussi descriptif que possible. [...]

Quel que soit le type de fichier, toutes les données contenues dans ce dernier consistent en une suite de nombres. La *taille* du fichier n'est que la longueur de la liste de ces nombres. Chaque nombre est appelé un *octet* (*byte*). Chaque octet contient 8 *bits*.[2] Chaque bit vaut 1 ou 0 et donc, il y a :
$\underbrace{2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2}_{8\,bits} = \underbrace{256}_{1\,octet}$ combinaisons possibles. Donc, un octet ne peut contenir qu'un nombre inférieur ou égal à 255. Il n'y a pas de type de données qui ne puisse être représenté par une liste d'octets.

La lettre à Mary sera encodée sous forme d'octets pour être enregistrée sur le disque de l'ordinateur. Nous savons tous qu'une image de télévision n'est qu'un ensemble de points (pixels) qui balayent l'écran de gauche à droite. De la même manière, une image peut être représentée dans un fichier, c'est-à-dire comme une séquence d'octets où chacun de ceux-ci est interprété comme un niveau de brillance : 0 pour le noir, 255 pour le blanc. Dans le cas de la lettre à Mary, la convention est d'enregistrer A comme 65, B comme 66, etc. Chaque caractère de ponctuation a son équivalent numérique.

Une table de correspondance appelée *jeu de caractères* ou *table de caractères* permet d'établir le lien entre lettres et nombres. Le jeu de caractères le plus répandu à ce jour est le jeu de caractères *ASCII* qui est l'acronyme d'*American Standard Code for Information Interchange*. Le tableau 3.3 montre le jeu de caractères ASCII entre les caractères et leur équivalent hex, décimal et octal.

[2] NdT : Le terme *bit* est une contraction de *binary digit* (ou nombre binaire).

TAB. 3.3 – Jeu de caractères ASCII.

Oct	Déc	Hex	Caract	Oct	Déc	Hex	Caract	Oct	Déc	Hex	Caract	Oct	Dec	Hex	Caract
000	0	00	*NUL*	033	27	1B	*ESC*	066	54	36	6	121	81	51	Q	154	108	6C	l
001	1	01	*SOH*	034	28	1C	*FS*	067	55	37	7	122	82	52	R	155	109	6D	m
002	2	02	*STX*	035	29	1D	*GS*	070	56	38	8	123	83	53	S	156	110	6E	n
003	3	03	*ETX*	036	30	1E	*RS*	071	57	39	9	124	84	54	T	157	111	6F	o
004	4	04	*EOT*	037	31	1F	*US*	072	58	3A	:	125	85	55	U	160	112	70	p
005	5	05	*ENQ*	040	32	20	*SPACE*	073	59	3B	;	126	86	56	V	161	113	71	q
006	6	06	*ACK*	041	33	21	!	074	60	3C	<	127	87	57	W	162	114	72	r
007	7	07	*BEL*	042	34	22	"	075	61	3D	=	130	88	58	X	163	115	73	s
010	8	08	*BS*	043	35	23	#	076	62	3E	>	131	89	59	Y	164	116	74	t
011	9	09	*HT*	044	36	24	$	077	63	3F	?	132	90	5A	Z	165	117	75	u
012	10	0A	*LF*	045	37	25	%	100	64	40	@	133	91	5B	[	166	118	76	v
013	11	0B	*VT*	046	38	26	&	101	65	41	A	134	92	5C	\	167	119	77	w
014	12	0C	*FF*	047	39	27	'	102	66	42	B	135	93	5D	]	170	120	78	x
015	13	0D	*CR*	050	40	28	(	103	67	43	C	136	94	5E	^	171	121	79	y
016	14	0E	*SO*	051	41	29	)	104	68	44	D	137	95	5F	_	172	122	7A	z
017	15	0F	*SI*	052	42	2A	*	105	69	45	E	140	96	60	`	173	123	7B	{
020	16	10	*DLE*	053	43	2B	+	106	70	46	F	141	97	61	a	174	124	7C	|
021	17	11	*DC1*	054	44	2C	,	107	71	47	G	142	98	62	b	175	125	7D	}
022	18	12	*DC2*	055	45	2D	-	110	72	48	H	143	99	63	c	176	126	7E	~
023	19	13	*DC3*	056	46	2E	.	111	73	49	I	144	100	64	d	177	127	7F	DEL
024	20	14	*DC4*	057	47	2F	/	112	74	4A	J	145	101	65	e				
025	21	15	*NAK*	060	48	30	0	113	75	4B	K	146	102	66	f				
026	22	16	*SYN*	061	49	31	1	114	76	4C	L	147	103	67	g				
027	23	17	*ETB*	062	50	32	2	115	77	4D	M	150	104	68	h				
030	24	18	*CAN*	063	51	33	3	116	78	4E	N	151	105	69	i				
031	25	19	*EM*	064	52	34	4	117	79	4F	O	152	106	6A	j				
032	26	1A	*SUB*	065	53	35	5	120	80	50	P	153	107	6B	k				

3.3 Commandes.

Le deuxième point commun à tout ordinateur inventé à ce jour est la *commande*. A l'aide de simples mots saisis séquentiellement, vous indiquez à un ordinateur ce qu'il doit faire. Les ordinateurs modernes semblent en avoir fini avec les commandes parce qu'ils présentent de belles interfaces graphiques et permettent que des actions soient réalisées avec une souris. Cependant, fondamentalement, des commandes sont exécutées de manière masquée. La saisie de commandes reste le seul moyen d'exercer le contrôle sur un ordinateur. Vous ne connaissez réellement rien de l'ordinateur tant que vous n'acquérez pas les commandes. L'utilisation d'un ordinateur revient donc à saisir des mots puis à presser 

jusqu'à ce que l'ordinateur affiche une réponse à l'écran. Pour la plupart, les commandes interagissent avec un ou plusieurs fichiers.

3.4 Connexion et changement de mot de passe.

Allumez votre station de travail LINUX. Après quelques minutes d'initialisation, vous verrez un *écran de connexion* (*login*) avec une *invite* (*prompt*). Celle-ci est constituée d'un ou plusieurs mots et vous permet d'effectuer une saisie. En l'occurrence, l'invite comprend le nom de l'ordinateur (chaque ordinateur a un nom constitué typiquement de huit caractères minuscules) et le mot `login` suivi de deux points (`:`). Les machines LINUX présentent un écran graphique au démarrage (par défaut, le plus souvent) de sorte que vous pourriez avoir une invite graphique dont le comportement est le même qu'en mode console. A présent, vous devriez entrer votre *identifiant* (ou *login name*) qui est une séquence de huit caractères en minuscules que vous a attribué votre administrateur. Ensuite vous pouvez presser la touche 

.

Une invite de *mot de passe* (*password prompt*) s'affiche, derrière laquelle vous devez saisir votre mot de passe. Eventuellement, l'identifiant et le mot de passe peuvent être identiques. Notez que votre mot de passe n'apparaît pas à l'écran : à mesure que vous le tapez, il demeure invisible. Après cette saisie, enfoncez la touche "Entrée" ou "Return" à nouveau. L'écran peut afficher un message et vous présenter à nouveau une autre invite de connexion. Cela signifie que vous avez tapé quelque chose d'incorrect. Faites un nouvel essai. Désormais, vous devriez effectuer seul un enfoncement de la touche "Entrée" de manière analogue au retour d'une machine à écrire. Vous devez aussi savoir que les erreurs humaines sont fréquentes et que chaque fois que quelque chose d'erroné est saisi, l'ordinateur affiche un message d'erreur. Il est rare que quelqu'un comprenne l'ensemble des principes de fonctionnement des ordinateurs du premier coup et que les commandes fonctionnent au premier coup d'essai.

Maintenant que vous êtes connecté, vous vous trouvez devant une invite de *shell* –un shell est un programme dans lequel un utilisateur effectue des saisies de commandes. Le shell est donc l'endroit où l'administrateur système [computer manager] passe le plus clair de son temps ; toutefois, ce cadre de travail ne doit pas forcément être luxueux comme vous le constatez.

Le premier exercice consistera à changer votre mot de passe. Tapez la commande `passwd`. L'ordinateur vous demande le mot de passe qui vous est actuel-

lement attribué.[3] Introduisez ce mot de passe de manière à ce que le shell vous invite à donner le nouveau mot de passe. Confirmez ce dernier. Notez que ce mot de passe doit comprendre des lettres, des nombres et certains signes de ponctuation. Vous verrez plus tard pourquoi il est important d'assurer une bonne sécurité sur le mot de passe. Le nouveau mot de passe prend cours immédiatemment en remplaçant le mot précédent avec lequel vous vous êtes connecté. La commande "passwd" peut aussi retourner des messages qu'éventuellement vous ne comprenez pas encore. Essayez de déterminer la connotation négative ou positive de ceux-ci.

Lorsque vous utilisez un ordinateur, essayez de vous imaginer les différents endroits au sein de l'ordinateur où vous positionnent les commandes plutôt que de considérer seulement l'encodage de ces commandes. Ainsi, après que vous ayez saisi la commande `passwd`, vous n'étiez plus dans le shell. C'est seulement lorsque vous avez quitté la commande `passwd` que vous êtes revenu dans le shell.

3.5 Afficher des fichiers.

Derrière votre invite, tapez `ls`. `ls` est l'abréviation en deux lettres (comme c'est souvent le cas de nombreuses commandes UNIX) pour *list*. `ls` affiche tous vos fichiers courants (et les répertoires qui sont des dossiers regroupant des fichiers). Vous pourriez trouver qu'`ls` ne fait pas grand-chose. Alors, retournez dans le shell en appuyant sur la touche "entrée". Comme il n'y a pas encore beaucoup de fichiers, cela semble bien vide. Notez que, pour la plupart, les commandes UNIX *ne* retournent *pas* de messages. La commande `passwd` est une exception. Si vous aviez des fichiers, ceux-ci seraient disposés en lignes ou en colonnes sans autre renseignement.

3.6 Touches d'édition de la ligne de commande.

Les touches suivantes sont utiles pour l'édition en ligne de commandes. Notez qu'UNIX est le fruit d'une évolution longue et non-linéaire. Aussi les touches 

et 

peuvent ne pas fonctionner correctement. Les combinaisons des touches suivantes sont très fréquentes dans les applications LINUX :

Ctrl-a se déplacer au début d'une ligne : 

,

Ctrl-e se déplacer à la fin d'une ligne : 

,

Ctrl-h supprime en arrière : 
,

Crtl-d supprime en avant : 

,

Ctrl-f se déplacer en avant d'un caractère : 
,

Crtl-b se déplacer en arrière d'un caractère : 
,

Alt-f se déplacer en avant d'un mot,

Alt-b se déplacer en arrière d'un mot,

[3] Si vous ne voulez pas poursuivre, faites Ctrl+C. Le shell vous rend un nouvelle invite.

Alt-Ctrl-f supprime un mot à droite du curseur,[4]

Alt-Ctrl-b supprime un mot à gauche du curseur,

Crtl-p commande précédente,

Ctrl-n commande suivante,

Observez que les préfixes *Alt* pour la touche  , *Crtl* pour  et *Shift* pour  signifient qu'il faut maintenir la touche enfoncée pendant que vous enfoncez puis relâchez la touche de la lettre. Ces touches sont des *modificateurs*. Notez aussi que la touche *Ctrl* est toujours insensible à la casse. Donc Ctrl-d c'est-à-dire  -  ou Crtl-D c'est-à-dire la séquence de touche  -  -  ont la même action. Le modificateur Alt,  -?, est un raccourci pour presser/relâcher  avant d'entrer une combinaison de touches. La combinaison Esc-*f* équivaut à Alt-*f*. UNIX se distingue d'autres systèmes d'exploitation en ce qui concerne l'usage d'*Esc*. Le modificateur *Alt* n'est pas insensible à la casse bien que certaines applications s'en servent avec une insensibilité à la casse. La touche *Alt* est aussi appelée touche *Méta*. Toutes ces touches sont parfois désignées par leur abréviation : par exemple *C-a* pour *Ctrl-a*, *M-f* pour *Méta-f* (ou *Alt-f*). Le modificateur *Ctrl* est parfois désigné par un accent circonflexe : par exemple, `^C` pour *Ctrl-C*.

Votre ligne de commande conserve l'historique des commandes saisies. *Ctrl-p* et *Ctrl-n* vous déplacent respectivement en avant et en arrière dans la pile des commandes entrées précédemment. Les nouveaux utilisateurs semblent très heureux de taper et retaper les commandes de bout en bout. N'oubliez pas d'utiliser l'historique pour réutiliser les commandes.

Ctrl-s provoque la suspension de la session en cours. Le clavier ne répond plus.

Ctrl-q réactive le clavier.

Ctrl-r active une recherche dans votre historique de commande. Le fait de presser *Ctrl-r* au milieu d'une recherche permet de trouver la correspondance suivante. *Ctrl-s* inverse l'effet (bien que certaines distributions confondent avec la suspension de session).

La commande *Tab* est extrêmement efficace pour compléter le nom d'une commande partiellement saisie. Cela est vrai pour les noms de fichiers ou de répertoires. Ainsi, il n'est pas nécessaire d'effectuer une saisie complète.

Vous pouvez presser *Tab* et, pour que les autres touches arrêtent de provoquer une émission sonore irritante, modifiez le fichier `/etc/inputrc` et ajoutez-y la ligne :[5]

```
set bell-style none
```

Déconnectez-vous pour vous reconnecter (nous en verrons davantage plus loin).

[4] NdT : il se peut que cette combinaison soit remplacée par **Alt-d**. Notez aussi que le fait de presser **Atl-c** avec le curseur sur une lettre minuscule la convertit en majuscule.

[5] Ndt : la terminologie anglo-saxonne utilise l'expression *edit a file* pour *modifier un fichier*.

3.7 Touches spéciales de la console.

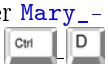
Il existe plusieurs touches spéciales interprétées directement dans une console LINUX ou une interface en mode texte. La combinaison *Ctrl-Alt-Del* initie l'arrêt et subséquemment la relance automatique du système d'exploitation. Il s'agit d'une méthode de redémarrage très brutale qui peut altérer le système de fichiers de votre système.

Les combinaisons *Ctrl-PgUp* et *Ctrl-PgDn* font défiler l'écran vers le haut et vers le bas, respectivement ; ce qui est utile pour lire du texte qui a défilé trop rapidement.

Vous pouvez utiliser *Alt-F2* pour accéder à une console indépendante et vous connecter à nouveau dans une session indépendante de la première. Six consoles virtuelles (d'*Alt-F1* à *Alt-F6*) sont ainsi accessibles. On les nomme *terminaux virtuels*. Si vous êtes en mode graphique, vous presserez *Ctrl-Atl-Fn* (où *n* est un nombre d'1 à 6), les touches *Alt-Fn* étant utilisées par certaines applications. La convention est que la septième console soit toujours graphique. *Ctrl-Alt-F7* vous ramène toujours dans une console graphique.

3.8 Création de fichiers.

Il y a différentes manières de créer un fichier. Tapez : `cat > Mary_Jones.-lettre` et saisissez quelques lignes de texte. Vous utiliserez ce fichier par après. La commande `cat` est utilisée pour écrire depuis le clavier dans le fichier `Mary_-Jones.lettre`. A la fin de la dernière ligne, pressez 

et ensuite 

pour clore la saisie. A présent, si vous exécutez `ls`, vous apercevrez `Mary_Jones-.lettre` parmi d'autres fichiers. Tapez `cat Mary_Jones.lettre` *sans* `>` et vous verrez que la commande `cat` affiche le contenu du fichier donné en argument à la commande, vous permettant ainsi de consulter votre lettre.

3.9 Caractères permis pour les noms de fichiers.

Bien que les noms de fichiers puissent contenir presque tous les caractères, les standards dictent que seuls les caractères suivants peuvent être utilisés pour nommer des fichiers ou des répertoires :

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 . _ -

Par conséquent, il ne faut jamais utiliser de caractères de ponctuation (hormis le point, le caractère de surlignement ou le tiret), de crochets ou de caractères de contrôle dans les noms de fichiers ou de répertoires. N'utilisez pas le caractère d'espacement ou *Tab* et, ne commencez jamais le nom d'un fichier par -.

3.10 Répertoires.

Nous avons mentionné qu'un système peut contenir typiquement 10.000 fichiers. Etant donné qu'il serait ardu de s'y retrouver dans 10.000 fichiers affichés suite à un `ls`, ces derniers sont regroupés selon leur type dans différents

"dossiers" de manière à être isolés d'autres fichiers. Par exemple, votre lettre pourrait être regroupées avec d'autres lettres dans un dossier idoine. Sur un ordinateur, un dossier est appelé un répertoire. Ceci est la troisième caractéristique commune des ordinateurs : tout fichier peut faire partie d'un répertoire. Pour comprendre comment fonctionnent les répertoires, tapez la commande `mkdir lettres`, le terme `mkdir` signifant *make directory*. A présent, tapez `ls`. Ceci devrait vous retourner une liste avec aussi bien `Mary_Jones.lettre` que `lettres`. Le fichier `lettres` ne contient encore rien sauf le nom d'un répertoire dans lequel d'autres fichiers pourront être placés. Pour se déplacer dans le répertoire `lettres`, il faut taper la commande `cd lettres` où `cd` signifie *change directory*. Etant donné que ce répertoire vient d'être créé, il ne contient pas de fichiers et cela peut se vérifier en tapant `ls`. Vous pouvez créer un fichier en exécutant la commande `cat` comme vous l'avez fait précédemment (essayez). Pour revenir dans le répertoire original, vous utiliserez la commande `cd ..` où `..` a la signification spéciale de vous faire sortir du répertoire courant. Tapez à nouveau `ls` pour vérifier que vous êtes remonté d'un cran dans la hiérarchie de vos répertoires.

Cependant, il est ennuyeux que vous ne puissiez voir la différence entre fichier et répertoire. La différenciation se fait en utilisant la commande `ls -l` où `-l` est une option signifiant *format long*. Si vous appliquez cette commande vous observerez de nombreux détails qu'éventuellement vous ne comprenez pas encore. Les trois informations qui d'emblée attireront votre attention sont : (i) tout-à-fait à droite, le nom de fichier ; (ii) la taille mentionnée dans la cinquième colonne (le nombre d'octets que contient le fichier) et (iii) le type de fichier, information se trouvant tout-à-fait à gauche. Parmi la chaîne de caractères, vous trouverez à l'extrême gauche soit `-` ou `d`. Un `-` signifie que vous avez à faire à un fichier tandis qu'un `d` montre qu'il s'agit d'un répertoire (*directory*). La commande `ls -l Mary_Jones.lettre` affichera seulement le fichier `Mary_Jones.lettre` ; ceci est utile pour connaître la taille d'un fichier particulier.

En fait, il n'y a pas de limitation sur le nombre de répertoires qu'on peut emboîter les uns dans les autres. De suite, nous allons entrevoir la disposition de tous les répertoires sur votre ordinateur.

Tapez la commande `cd /` où `/` a la signification spéciale de désigner le répertoire qui englobe tous les autres et qu'on nomme la *racine* (*root*). Maintenant tapez `ls -l`. Si la liste est trop longue et déborde au-delà de votre écran, utilisez plutôt la commande `ls -l | less` (pressez alors PgUp ou PgDn pour naviguer dans la page, puis `q` pour quitter). A présent, vous pouvez vous déplacer dans le système à l'aide de la commande `cd` en n'oubliant pas que `cd ..` vous fait "remonter d'un cran" et que `cd /` vous replace dans le répertoire racine.

A tout instant, vous pouvez utiliser la commande `pwd` (*present working directory*) pour déterminer le répertoire dans lequel vous êtes.

Quand vous aurez fini, déconnectez-vous avec la commande `logout` (si vous êtes dans une console virtuelle) ou avec `exit` puis "quitter la session" dans le menu graphique de base (si vous êtes en mode graphique).

Chapitre 4

Matériel PC.

Ce chapitre décrit de manière simplifiée les constituants matériels d'un PC. Les lecteurs qui ont construit le leur ou qui ont configuré divers périphériques sur MS-Windows peuvent passer cette partie qui n'est incluse que pour être aussi complet que possible. En réalité, le titre aurait dû être "*Organisation d'un micro-ordinateur*" ou, comment votre machine est électroniquement structurée.

4.1 Carte-mère.

A l'intérieur de votre machine, vous trouverez une carte de grande taille, en un seul bloc, sur laquelle il y a de nombreux circuits. C'est la *carte-mère* (voir la Figure 4.1). Elle est connectée à une alimentation électrique quelque peu bourdonnante. Elle est aussi connectée au clavier et à d'autres *périphériques*. [Il n'y a rien qui ne soit la carte-mère, l'alimentation ou un périphérique]

La carte-mère contient quelques micro-puces de grande taille et certaines –plus nombreuses– de petite taille. Voici une liste des composants les plus importants :[1]

RAM : *Random Access Memory* ou *mémoire-vive*. Cette mémoire est une séquence linéaire unique d'octets qui sont effacés lorsque l'ordinateur n'est pas alimenté. Elle contient une séquence d'*instructions* simples codées sur un à plusieurs octets de longueur : par exemple, additionner ce nombre à celui-là ; déplacer ce nombre vers ce périphérique ; aller dans une autre partie de la RAM pour obtenir d'autres instructions ; copier cette partie de la RAM vers une autre partie. Si votre machine présente "64 megs" (ou 64 mega-octets ou encore 64 Mo), elle possède $64 \times 1024 \times 1024$ octets de RAM. Les localisations dans cet espace sont des *adresses mémoires*, de sorte que dire "adresse mémoire 1000" signifie "le 1000ème octet en mémoire".

ROM Une petite partie de la RAM n'est pas effacée lors des arrêts de la machine. Elle est nommée ROM ou *Read Only Memory*. Elle est fixée d'origine par le constructeur et n'est usuellement pas modifiée durant la vie de l'ordinateur (pas même son nom). Elle correspond à une partie de la RAM proche de la fin du premier méga-octet de mémoire si bien que cette

[1] NdT : les noms en anglais ont été maintenus mais le texte propose une traduction française.

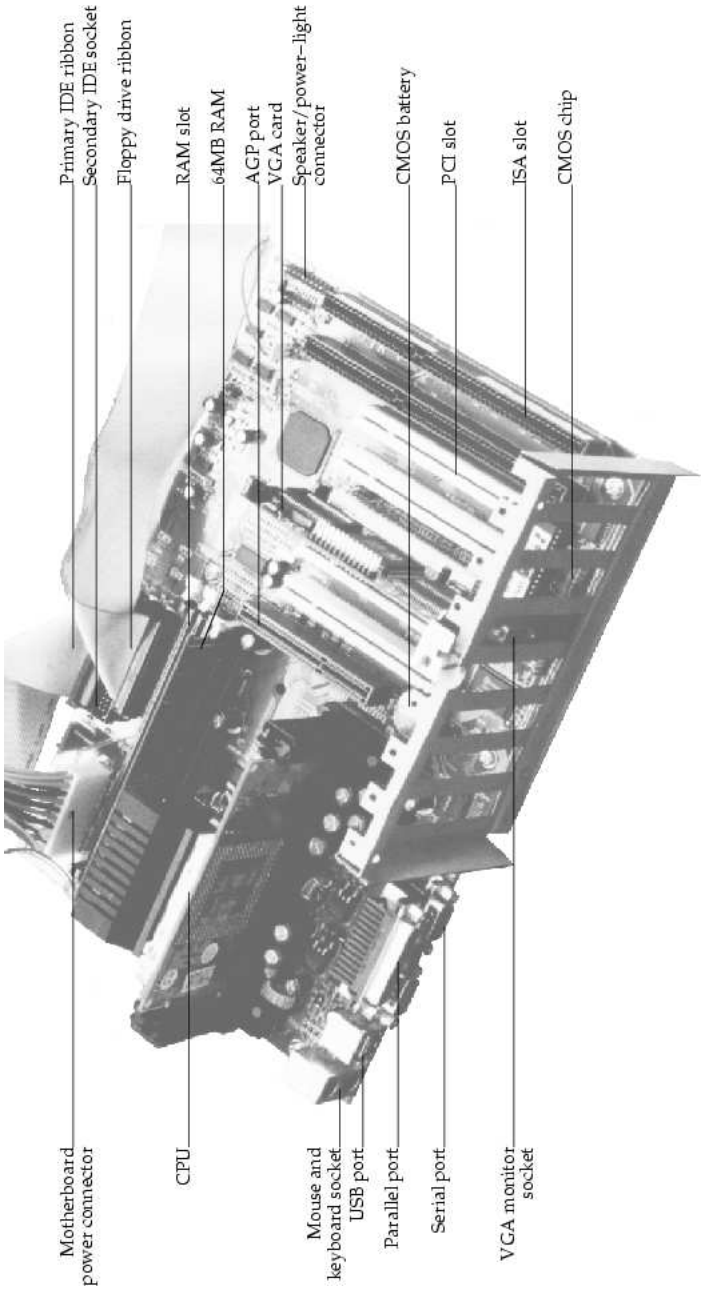


FIG. 4.1 – Carte-mère partiellement assemblée.

partie de la RAM n'est pas physiquement utilisable. La ROM contient des instructions pour démarrer le PC et accéder à certains périphériques.

CPU *Unité centrale de traitement* (*Central Processing Unit*). C'est le dispositif couramment appelé 80486, Pentium ou AMD. Au démarrage, cette unité "saute" l'adresse mémoire 1040475 (0xFE05B) et commence des instructions de lecture. La première instruction à laquelle elle accède lui enjoint de rechercher plus d'informations sur le disque ou de retourner à l'écran le message "`Boot failure`" si elle ne trouve rien d'exploitable. La CPU requiert une horloge. Cette dernière opère à une vitesse élevée de plusieurs millions ou, maintenant, de plusieurs milliards de Hz (Hertz ou s^{-1}) d'où l'attribut conféré à une machine : 2 GHz, par exemple. En première approximation, ce nombre est proportionnel au nombre d'instructions effectuées par seconde.

PORTS E/S (I/O Ports) Ce terme désigne les ports d'entrée et de sortie (ou *I/O : Input/Output*). Les ports sont un bloc de la RAM, en parallèle à celle-ci. Il y a 65.536 ports E/S. Donc, la partie E/S est petite par rapport à la RAM. Les ports E/S servent à dialoguer avec les périphériques. Quand la CPU écrit un octet au port 632 (0x278), elle émet un octet à travers le port parallèle de votre machine. La plupart des ports E/S ne sont pas utilisés. Cependant, il n'y a pas de puce qui leur soit dédiée.

Slots ISA ISA[2] est un type de connexion pour le branchement des périphériques comme les cartes modem ou les cartes son. Chaque carte s'attend à communiquer via un port E/S (ou plusieurs ports E/S consécutifs). La détermination du port qu'une carte utilisera peut être réalisée par le constructeur, par vous à l'aide de cavaliers [il s'agit de petites pièces en plastique manipulables avec les doigts ou une pince] ou de contacteurs fixés sur la carte. Il existe encore une autre possibilité : la CPU peut configurer les ports en utilisant le système *Plug-and-Pray* [ceci signifie que vous branchez le périphérique et faites signe à votre divinité préférée pour obtenir une assistance spirituelle. Des personnes se plaignent vraiment de ce que ceci pourrait être pris au sérieux –non, il s'agit d'une blague : le vrai terme est *Plug 'n Play*] ou *PnP*. Une carte doit parfois envoyer un signal à la CPU pour signifier qu'elle est prête à envoyer ou à recevoir des octets au travers d'un port E/S. Les cartes font cela grâce à une série de 1 à 16 connecteurs situés dans le slot ISA. Ces signaux sont appelés *lignes de demande d'interruption*, IRQ[3] (ou simplement *interruptions*) notées de 0 à 15. A l'instar des ports E/S, les IRQ peuvent être sélectionnées par cavaliers ou non. Si vous débranchez une vieille carte ISA, vous pourrez voir le fil de cuivre qui court depuis les cavaliers IRQ jusqu'au bord du connecteur. Enfin, les cartes ISA peuvent aussi accéder à de la mémoire directement par un des 8 canaux DMA (canaux d'accès directs à la mémoire)[4] qui peuvent éventuellement être sélectionnés par des cavaliers.

[2] *"eye-sah"* en anglais. Le terme *slot* signifie "connecteur". Il s'agit d'un emplacement sur la carte-mère permettant de connecter des cartes de périphériques ou d'extension.

[3] IRQ est l'acronyme d'*Interrupt ReQuest line* ou demande d'interruption. Il s'agit d'un signal électronique permettant de modifier le fonctionnement du processeur pour qu'il exécute une portion de code différente de celle en cours d'exécution. A la fin de l'interruption, l'exécution du code interrompu reprend là où elle a été suspendue.

[4] *DMA channels* signifie *Direct Memory Access Channels* ou accès direct à la mémoire. Il s'agit d'une méthode de tranfert de données évitant l'utilisation du processeur et/ou d'une zone

Cependant, toutes les cartes n'utilisent pas de canaux DMA.
En résumé, la CPU et le périphérique que vous considérez doivent coopérer à l'aide de 3 instruments : les ports E/S, l'IRQ et les canaux DMA. Si deux cartes quelconques sont en conflits en utilisant soit le même port E/S, le même nombre IRQ ou le même canal DMA, elles ne peuvent pas fonctionner (au pire, votre machine plante) [...].

Slots ISA ("à 8 bits") Les anciennes cartes-mères présentaient des connexions ISA plus courtes que celles d'aujourd'hui. Notez que les vôtres ont une double connection (appelée ISA "16 bits") avec un espacement. La plus grande peut encore correspondre à une carte ISA à 8 bits plus ancienne : c'est le cas de nombreuses cartes modems.

Slots PCI Les connecteurs PCI[5] sont analogues aux ISA mais constituent une nouvelle norme afin d'autoriser des périphériques de hautes performances comme les cartes-réseau et les cartes graphiques. Les cartes PCI utilisent aussi une ligne IRQ, un port E/S et éventuellement un canal DMA. Cependant, elles sont configurées automatiquement par la CPU comme une partie du PCI standard (donc elles portent rarement des cavaliers).

Slots AGP Ces connecteurs sont caractérisées par d'encore plus grandes performances et sont utiles pour les *processeurs graphiques accélérés* (AGP pour *Accelerated Graphics Processors*) c'est-à-dire les cartes graphiques 3D pour les jeux. Ils sont également auto-configurés.

Ports série Une connexion par port-série peut aller directement de la carte-mère à un connecteur sur le capot de votre machine. Usuellement, il y a deux ports série. Ils peuvent piloter un modem externe ainsi que certains types de souris ou d'imprimantes. Le port série est un système simple et bon marché pour connecter une machine dans le cas où des transferts de données lents sont acceptables (moins de 10 Ko par secondes). Les ports série ont leur propre carte ISA construite sur la carte-mère. Cette carte utilise le port E/S 0x3F8-0x3FF et la ligne IRQ 4 pour le premier port série (appelé COM1 sous MS-DOS/Windows) ainsi que le port E/S 0x2F8-0x2FF et la ligne IRQ 3 pour COM2. Une discussion sur la technologie du port-série est présentée en section 4.4.

Ports parallèle En principe, seule votre imprimante pourrait être connectée sur un port en parallèle. Cependant, ces ports sont plutôt rapides et capables de tranférer 50 Ko par seconde). Par conséquent, beaucoup de périphériques à port en parallèle sont disponibles (par exemple, le périphérique CD-ROM). Les câbles des ports en parallèle ne peuvent être que de quelques mètres de long ; au-delà on observe des erreurs de transmission. Le port en parallèle utilise le port E/S 0x378-0x37A et IRQ 7. Si votre ordinateur a 2 ports en parallèle, alors, le second utilise le port E/S 0x278-0x27F mais pas de ligne IRQ.

Ports USB L'*Universal Serial Bus* permet à tout type de matériel d'être branché sur un autre. L'idée est qu'un jour les périphériques-série et parallèle

E/S classique. Le processeur continue l'exécution de programmes lui incombant directement et les périphériques ayant leur propre canal DMA ne sont plus obligés de faire la queue pour obtenir du processeur qu'il leur attribue un port d'E/S. Ceci était surtout utile avec des processeurs lents. La technologie DMA tombe en désuétude de nos jours.

[5] "pee-see-eye" avec la prononciation anglaise.

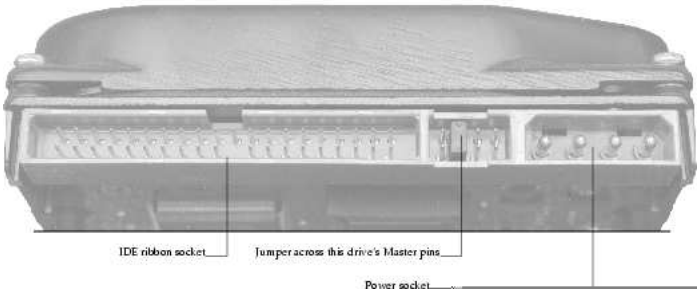


FIG. 4.2 – Connexion terminale d'un pilote IDE typique avec le connecteur IDE à gauche, les cavaliers dans l'encoche centrale et la douille de connexion d'alimentation (à droite).

seront abandonnés au profit de l'USB à partir duquel tous les périphériques seront branchés en éventail. Nous n'irons pas plus loin sur ce sujet.

Nappe IDE La nappe IDE se branche sur le pilote matériel de votre disque dur (ou `C :` sur MS-DOS ou MS-Windows) et aussi sur votre pilote de CD-ROM (appelé parfois IDE-CDROM). Le câble IDE est réellement connecté à sa carte interne PCI qui se trouve sur la carte-mère. Il existe deux connecteurs IDE utilisant les ports 0xF000-0xF007 et 0xF008-0xF00F ainsi que les lignes IRQ 14 et 15, respectivement. La plupart des CD-ROM de type IDE sont aussi de type ATAPI. ATAPI est un standard (similaire à SCSI, voir 4.1) qui permet à beaucoup d'autres périphériques d'être branchés sur le câble de la nappe IDE. On les note ATAPI-(ceci ou cela).

Nappe SCSI Une autre nappe peut être présente, qui sort d'une carte (appelée adaptateur SCSI ou carte SCSI) ou de la carte-mère. Les PCs domestiques ont rarement du matériel SCSI, coûteux et réservés aux serveurs. Les câbles SCSI ont une densité de fils plus élevées que celle des câbles IDE. Ils se terminent par un pilote matériel, un pilote de bandes, un CD-ROM, ou autres. Il n'est pas possible à ces câbles d'être "juste branchés" : ils doivent être connectés bout-à-bout, le dernier périphérique portant une *terminaison SCSI* qui ferme le circuit électrique. Pour plus d'information, consultez la section 43.6.10.

4.2 IDE maître et esclave.

Deux lecteurs matériels peuvent être connectés sur la même nappe IDE. La nappe seule ne peut distinguer qui est qui de sorte que le lecteur lui-même possède des cavaliers permettant de lever l'ambiguité (voir la Figure 4.2).

Les cavaliers permettent un arrangement de types MA (*Master*), SL (*Slave*), CS (*Cable Select*) ou *Master-only/Single-drive/*. L'arrangement MA signifie que votre lecteur est le premier sur la nappe IDE. L'arrangement SL signifie que votre lecteur est le second sur cette même nappe tandis que, pour CS, votre machine doit effectuer le choix (seuls certains PCs fonctionnent ainsi). L'arrangement *Master-only/Single-drive* veut dire qu'il n'y a pas de second lecteur sur la nappe IDE.

Parfois, il y a une seconde nappe ce qui, au total, vous donne 4 possibilités. La première nappe est connue comme IDE1 (étiquetée comme telle sur la carte-mère) ou *nappe primaire*, tandis que la seconde est appelée IDE2 ou *nappe secondaire*. Les 4 périphériques sont alors nommés : *premier maître*, *premier esclave*, *second maître secondaire* et *second esclave*. Leur étiquetage sous LINUX est discuté à la section 19.4.

4.3 CMOS.

CMOS [*Complementary Metal Oxide Semiconductor*] est une petite application se trouvant dans la ROM. Elle est aussi connue sous le nom de configuration *BIOS* de la ROM. Vous pouvez la démarrer au lieu de votre OS (système d'exploitation ou *Operating System*) en appuyant, après avoir allumé la machine, sur les touches 

ou 

, le plus souvent. En principe, le message `Press <touche> to enter set-up` apparaîtra. En pratiquant ainsi, vous accéderez au programme CMOS où il vous est loisible de changer la configuration de votre machine. Les programmes CMOS diffèrent selon les constructeurs.

A l'intérieur du CMOS, vous pouvez activer ou désactiver des dispositifs ou périphériques pré-installés (tels que votre souris ou vos ports en série) ; configurer l'horloge matérielle de votre ordinateur (pour avoir l'heure et la date justes) ; ou encore sélectionner la séquence de démarrage (en choisissant le périphérique de démarrage : le disque dur, un CD-Rom, tout ce dont dont vous aurez besoin pour installer LINUX). Les termes "démarrer", "initialiser" ou "amorcer" sont désignés en anglais par *To boot* [...]. Vous pouvez aussi y configurer votre disque dur. Vous sélectionnerez `Hardrive autodetection` que vous soyez en train d'installer une nouvelle machine ou que vous ajoutiez/retiriez un (ou des) disque(s) [le terme *autodetection* se rapporte à un système, qui bien qu'étant incomplet, s'autoconfigure. Dans ce cas, le CMOS sonde le périphérique pour déterminer ses capacités. Les très vieux CMOS exigent que vous entriez les données de périphériques manuellement.] Les différents CMOS présentent différentes méthodes. Aussi, feuilletez les menus de votre BIOS pour déterminer ce dont il est capable.

Le CMOS est important parce qu'il permet de configurer certains périphériques inclus sur la carte-mère. Les CMOS modernes vous permettent de fixer le nombre des ports E/S et des lignes IRQ pour particulariser les périphériques que vous utilisez. Par exemple, vous pouvez permuter COM1 et COM2 ou utiliser un port E/S non-standard pour votre port parallèle. Lorsqu'il est nécessaire de configurer un périphérique sous LINUX, il est judicieux d'éteindre la machine pour apprendre du CMOS ce qu'il sait de ce périphérique. Pour davantage d'informations, consultez le chapitre 43.

4.4 Périphériques série.

Les ports série facilitent la communication à vitesse lente sur de courtes distances en utilisant des câbles à 8 brins ou moins. Les standards sont obsolètes et la communication n'est pas très tolérante vis-à-vis d'erreurs. Il y a tant de variantes sur le port série qu'il en devient presque magique qu'un périphérique puisse fonctionner correctement. Vous trouverez ici une brève explication sur

les protocoles, l'électronique et le matériel. Les documents Serial-HOWTO et Modems HOWTO renferment une analyse exhaustive (voir le chapitre 17).

Certains périphériques qui communiquent par le port série sont :

- les modems domestiques ordinaires "dial-up", [6]
- certaines connexions à l'internet permanentes se faisant *via* des systèmes de type "modems",
- les souris ou tout matériel apparenté,
- les terminaux en mode texte,
- les imprimantes,
- les caisses-enregistreuses,
- les lecteurs de cartes magnétiques,
- les unités de puissance continue (UPS ou *Uninterruptible Power Unit*) et
- les périphériques des micro-processeurs embarqués.

Un périphérique est connecté à votre ordinateur par un câble avec un connecteur portant 9 ou 25 aiguilles (à chaque bout) (voir les figures 4.3 et 4.4). Les connexions sont de type :

- DB-9

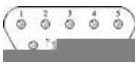


FIG. 4.3 – Connexion DB-9 et numérotation

- DB-25

FIG. 4.4 – Connexion DB-25 et numérotation

–

Seules huit des aiguilles sont réellement fonctionnelles, toutefois.

L'attribution des aiguilles à leur fonction est mentionnée dans le tableau 4.1.

La manière dont les périphériques série communiquent est directe : un flux d'octets est envoyé entre l'ordinateur et le périphérique en divisant chaque octet en 8 bits. La tension est ajustée sur une aiguille appelée *TD pin* ou *transmit pin* selon que le bit vaut 1 ou 0. Un bit de 1 correspond à une tension négative (de -15 à -5 volts) et un bit 0 à une tension positive (de +5 à +15 volts). Le *RD pin* ou *receive pin* reçoit les bits de la même manière. L'ordinateur et le périphérique série doivent être d'accord sur la vitesse de tranferts (*data rate* ou *serial port speed*) de sorte que l'ajustement et la lecture des niveaux de tensions soient synchronisés de manière idoine. La vitesse est souvent exprimée en *bps* (bits par seconde). Le tableau 4.2 montre une liste de vitesses de port série possibles.

Une souris communique avec une vitesse comprise entre 1.200 et 9.600 bps. Les modems communiquent à des vitesses de transfert de 19.200, 38.400, 57.600 ou 115.200 bps. Il est rare de trouver des périphériques série ou des ports série qui ne supportent pas les vitesses reprises dans le tableau.

[6] NdT : Ce mot désigne toute connexion non-permanente avec ou sans réseau téléphonique.

TAB. 4.1 – Correspondances des numéros d'aiguilles et de leur fonction.

Connecteur DB-9	Connecteur DB-25	Acronyme	Fonction	Direction (PC-Périph.)
3	2	TD	Transmis. données	→
2	3	RD	Reception données	←
7	4	RTS	Requête envoi	→
8	5	CTS	Prêt à l'envoi	←
6	6	DSR	Jeu de données prêt	←
4	20	DTR	Terminal données prêt	→
1	8	CD	Détection de porteurse	←
9	22	RI	Indicateur en boucle	←
5	7		Signal	

TAB. 4.2 – Vitesses typiques de transfert de données sur un port en série (unité : bps).

50	200	2.400	57.600	5760.00	2.000.000
75	300	4.800	115.200	921.600	2.500.000
110	600	9.600	230.400	1.000.000	3.000.000
134	1.200	19.200	460.800	1.152.000	3.500.000
150	1.800	38.400	500.000	1.500.000	4.000.000

Pour synchroniser le périphérique avec l'ordinateur, un *start bit* additionnel précède chaque octet et deux *stop bits* suivent chaque octet. Il peut aussi y avoir un *parity bit* indiquant s'il y a un nombre pair ou impair de 1 dans l'octet (vérification de présence d'erreurs). En théorie, il peut y avoir jusqu'à 12 bits envoyés pour chaque octet de données. Les bits additionnels sont optionnels et spécifiques aux périphériques. Les modems ordinaires communiquent avec le protocole *8N1*, c'est-à-dire 8 bits de données, pas de bit de parité et un bit d'arrêt *(stop bit)*. Une souris communique avec 8 bits et aucun bit de démarrage, d'arrêt ou de parité. Certains périphériques n'utilisent que 7 bits et en conséquence sont limités dans leur communication aux caractères ASCII (vu que la gamme de caractères s'arrête à 127).

Certains types de périphériques utilisent deux aiguilles appelées "*Request to Send* (RTS)" et *"Clear to Send* (CTS)". C'est soit l'ordinateur soit le périphérique qui applique une tension de +12 volts pour indiquer qu'il est prêt à recevoir des données. Les aiguilles appelées DTR ("*Data Terminal Ready*") et DSR ("*Data Set Ready*") travaillent à la place des deux précédentes ; elles effectuent les mêmes missions mais leurs numéros ne sont pas les mêmes. En particulier, les modems domestiques font usage des aiguilles RTS/CTS. Le mécanisme qu'elles utilisent est appelé *flux de contrôle RTS/CTS* ou *contrôle de flux matériel*. Certains périphériques simples n'utilisent pas de contrôle de flux. *Ces périphériques perdent les données qu'ils envoient au récepteur si ce dernier n'est pas prêt.*

D'autres périphériques doivent également communiquer lorsqu'ils sont prêts à recevoir des données ; toutefois, ils n'ont pas de connexions par aiguilles RTS/CTS ou DTR/DSR. Ils émettent des caractères de contrôle spéciaux envoyés parmi le flux de données pour indiquer que le flux devrait s'interrompre ou au contraire reprendre. Ceci est connu sous le nom de *contrôle de flux logiciel*. Les périphériques qui supportent optionnellement les deux méthodes de contrôle de flux devraient être configurés pour utiliser le contrôle de flux matériel. En particulier, un modem utilisé par Linux *doit* présenter un contrôle de flux matériel activé.

Les deux autres aiguilles sont l'*indicateur de boucle* (RI = *Ring Indicator*) et l'indicateur de détection de porteuse (CD = *Carrier Detect*). Elles ne sont utilisées que par des modems pour indiquer un appel entrant et la détection d'un modem pair, respectivement.

L'attribution des aiguilles et le protocole (y compris certaines spécifications à caractère électrique qui ont été omises) sont appelés protocole RS-232. Ce dernier est réalisé en utilisant une puce standard 16550 UART (*Universal Asynchronous Receiver-Transmitter*). Le protocole RS-232 est assez vite perturbé par les bruits électriques ce qui limite la longueur sur laquelle une transmission peut être effectuée ainsi que la vitesse de transmission. Un câble d'un demi-mètre de long transporte 115.200 bps sans erreur. Un câble de 15 m n'est pas fiable à des vitesses de transmission supérieures à 19.200 bps. D'autres protocoles (comme RS-423 ou RS-422) autorisent de plus grandes distances et fonctionnent avec un rapport vitesse/distance plus avantageux.

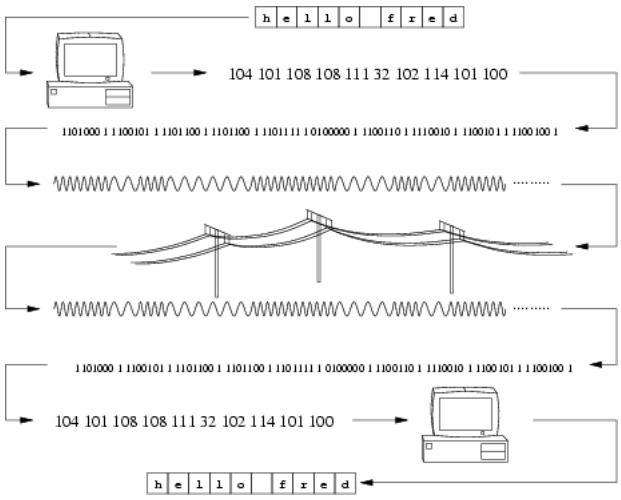


FIG. 4.5 – connexion à distance entre deux ordinateurs reliés par modems.

4.5 Modems.

Les lignes téléphoniques ayant été conçues pour transporter la voix, elles présentent des limitations lorsqu'il s'agit de transmettre des données. Il ressort que le meilleur moyen pour transférer des nombres binaires via une ligne téléphonique est d'utiliser deux fréquences : l'une basse pour 0 et l'autre plus haute pour 1. La figure 4.5 illustre schématiquement le procédé.

La conversion d'une tension en fréquence et l'opération inverse s'appellent respectivement *modulation-démodulation*, ce qui a donné le terme *modem*. Le terme *baud* désigne le nombre de fois que la fréquence peut être commutée par seconde ; le terme est interchangeable avec bps. Il existe beaucoup de nouvelles techniques modernes de modulation pour obtenir le meilleur d'une ligne téléphonique de sorte que des modems à 57.600 bps sont désormais le standard (au moment où ce texte est écrit). Les modems ont sur les données d'autres actions que la modulation : ils peuvent empaqueter les données pour réduire leur redondance (c'est la compression de bits ou *bit compression*), améliorer la détection d'erreurs et effectuer des compensations (*correction d'erreurs*). De tels protocoles de modems portent les noms V.90 (57.600 bps), V.34 (33.600 ou 28.899 bps), V.42 (14.400 bps) ou V.32 (14.400 bps ou moins). Lorsque deux modems sont en connexion, il est nécessaire qu'ils négocient un protocole "V" à utiliser. Cette négociation est basée sur leurs capacités respectives et l'état de la ligne à ce moment-là.

Un modem peut présenter deux états : le *mode commande* ou le *mode de connexion*. Un modem est connecté s'il peut entendre le *signal de transport* d'un modem pair[7] au cours d'un appel téléphonique actif (et qu'il y a probablement réception et transmission de données de la manière expliquée ici). Autrement, il est en mode commande. Dans ce mode, un modem ne peut ni moduler ni transmettre des données. En revanche, il interprète des séquences spéciales de texte qui lui sont envoyées via une ligne série. Ces séquences de

[7] au sens de semblable dans sa fonction.

texte commencent par les lettres `AT` et sont appelées `AT`*tention commands*. Les commandes `AT` sont envoyées par votre ordinateur de manière à configurer votre modem pour les conditions du moment de votre ligne téléphonique, la fonction projetée, la capacité du port série. Par exemple, il y a des commandes pour : permettre une réponse automatique en boucle, installer la méthode de contrôle de flux, composer un numéro, et raccrocher. La séquence de commandes pour configurer le modem est appelée *chaîne d'initialisation du modem*. La manière de réaliser manuellement ces commandes est discutée aux sections 33.6.3, 35.3 et 42.1. Cela deviendra intéressant lorsque vous configurerez votre *fournisseur de service internet* (ISP ou *Internet Service Provider*).

Vu que chaque modèle de modem utilise un jeu de commandes légèrement différentes, il est utile de se familiariser avec le manuel de votre modem. Les modems modernes, pour la plupart, utilisent le *jeu de commandes Hayes*, un jeu générique des commandes modems les plus utiles. Cependant, le jeu Hayes a une manière de gérer le contrôle de flux matériel que beaucoup de modems n'ont pas encore intégrée. Chaque fois que dans ce livre des exemples d'initialisation de modems seront données, une note se rapportant à cette section sera émise. Usuellement, il est suffisant de configurer votre modem avec les options par défaut du constructeur, mais souvent une seconde commande est requise pour permette le contrôle de flux matériel. Il n'y a pas de chaîne d'initialisation pour tous les modems. Les sites internet `http://www.spy.net/~dustin/modem` et `http://www.teleport.com/~curt/modems.html` sont des ressources utiles pour obtenir des spécifications sur les modems.

Chapitre 5

Les commandes de base.

Tout système UNIX est sensible à la casse. Une commande dont une seule lettre est convertie en majuscule est différente de la commande entièrement en minuscule. Ceci est vrai pour les fichiers, les répertoires, les formats de fichiers de configuration et la syntaxe de tous les langages de programmation.

5.1 La commande `ls`, fichiers cachés, options des commandes.

En plus des fichiers textes ordinaires et des répertoires, il existe d'autres types de fichiers, quoique tous contiennent une liste d'octets. Un fichier *caché* est un fichier qui n'apparaît pas lorsque la commande `ls` est exécutée pour afficher le contenu d'un répertoire. Pour voir un fichier caché, il faut que vous exécutiez la commande `ls -a` où `-a` est une option pour désigner tous les fichiers (*all files*). Une autre variante est `ls -l` qui affiche le contenu d'un répertoire dans le format *long*. Le tiret "`-`" permet d'utiliser des variantes d'une commande, appelées *options* ou *arguments* de commandes. La plupart des commandes UNIX admettent diverses options. Elles peuvent être combinées afin d'apporter de la concision [les commandes sous la licence des logiciels libres GNU sont, à ce point de vue, meilleures : elles disposent d'un plus grand nombre d'options que les commandes traditionnelles UNIX et apportent davantage de souplesse] : par exemple, `ls -a -l`, `ls -l -a` ou `ls -al` produisent une liste longue de tous les fichiers, y compris cachés.

Toutes les commandes GNU prennent `-h` ou `--help` comme arguments. Vous pouvez saisir une commande suivie de cet argument afin d'obtenir un *résumé de l'usage* de cette commande. Ceci consitue une aide brève des options que vous pourriez avoir oubliées si vous êtes déjà familier avec la commande (il ne s'agit pas d'une description exhaustive). Plus tard, nous verrons qu'il y a des pages de manuel pour cela (pages `man`).

La différence entre une fichier ordinaire et un fichier caché est que ce dernier commence avec un point ou un double point. Le fait de cacher un fichier de cette manière n'a pas pour but la sécurité mais le confort d'utilisation.

La commande `ls -l` est quelque peu déroutante pour le novice. Sa version plus explicite est `ls --format=long`. De manière analogue, l'option *all* peut être utilisée sous la forme `ls --all`, ce qui produit le même résultat que `ls -a`.

5.2 Messages d'erreurs.

Bien que la plupart des commandes ne renvoient pas de messages lors d'une exécution réussie [l'ordinateur accepte et exécute la commande], les commandes devraient afficher d'éventuels messages d'erreur de manière cohérente. Le format varie d'une commande à l'autre mais souvent, il apparaît ainsi : *nom_ de_ commande : what was attempted : error message*. Par exemple, la commande `ls -l azerty` revoie une erreur `ls : azerty : No such file or directory`. Que se passe-t-il réellement quand la commande `ls` tente de lire le fichier `azerty` ? Vu que le fichier n'existe pas, un code d'erreur 2 survient. Ce code d'erreur correspond à un cas pour lequel un fichier ou un répertoire n'est pas trouvé. Ce code d'erreur est automatiquement traduit en `No such file or directory`. Il est important de comprendre la différence entre un message d'explication qu'une commande retourne (tels que les messages rapportés par la commande `passwd` dans le chapitre précédent) et un code d'erreur traduit sous forme de texte. Le fait est qu'il y a beaucoup de raisons à l'obtention d'un code d'erreur identique (il existe une centaine de codes d'erreur). L'expérience vous montrera que les messages d'erreur ne vous disent pas ce que vous faites, mais seulement ce qui va mal. Ils ne devraient pas être considérés comme paroles d'évangile.

Le fichier `/usr/include/asm/errno.h` contient une liste complète des erreurs fondamentales. Par ailleurs, plusieurs autres fichiers d'en-tête (qui se terminent par un `.h`) peuvent définir leurs propres erreurs de code. Sous UNIX, cependant, cela constitue 99% des erreurs susceptibles d'être produites. Pour l'heure, la plupart d'entre elles n'ont pas beaucoup de signification mais elles sont répertoriées dans le tableau 5.1 à titre de référence.

TAB. 5.1 – Code d'erreurs commun de LINUX.

numéro	définition en C	messages
0		Success
1	EPERM	Operation not permitted
2	ENOENT	No such file or directory
3	ESRCH	No such process
4	EINTR	Interrupted system call
5	EIO	Input / output error
6	ENXIO	Device not configured
7	E2BIG	Argument list too long
8	ENOEXEC	Exec format error
9	EBADF	Bad file descriptor
10	ECHILD	No child processes
11	EAGAIN	Resource temporarily unavailable
11	EWOULDBLOCK	Resource temporarily unavailable
12	ENOMEM	Cannot allocate memory
13	EACCESS	Permission denied
14	EFAULT	Bad address
15	ENOTBLK	Block device required
16	EBUSY	Device or resource busy
17	EEXIST	File exists

TAB. 5.1. – Suite (1).

numéro	définition en C	Message
18	EXDEV	Invalid cross-device link
19	ENODEV	No such device
2à	ENOTDIR	Not such a directory
21	EISDIR	Is a directory
22	EINVAL	Invalid argument
23	ENFILE	Too many open files in system
24	EMFILE	Too many open files
25	ENOTTY	Inappropriate ioctl for device
26	ETXTBSY	Text file busy
27	EFBIG	File too large
28	ENOSPC	No space left on device
29	ESPIPE	Illegal seek
30	EROFS	Read-only file system
31	EMLINK	Too many links
32	EPIPE	Broken pipe
33	EDOM	Numerical argument out of domain
34	ERANGE	Numerical results out of range
35	EDEADLK	Resource deadlock avoided
35	EDEADLOCK	Resource deadlock avoided
36	ENAMETOOLONG	File name too long
37	ENOLCK	No locks available
38	ENOSYS	Function not implemented
39	ENOTEMPTY	Directory not empty
40	ELOOP	Too many levels of symbolic links
	EWOULDBLOCK	*same as* EAGAIN
42	ENOMSG	No message of desired type
43	EIDRM	Identifier removed
44	ECHRNG	Channel number out of range
45	EL2NSYNC	Level 2 not synchronized
46	EL3HLT	Level 3 halted
47	EL3RST	Level 3 reset
48	ELNRNG	Link number out of range
49	EUNATCH	Protocol driver not attached
50	ENOCSI	No CSI structure available
51	EL2HLT	Level 2 halted
52	EBADE	Invalid exchange
53	EBADR	Invalid request descriptor
54	EXFULL	Exchange full
55	ENOANO	No anode
56	EBADRQC	Invalid request code
57	EBADSLT	Invalid slot
	EDEADLOCK	*same as* EDEADLK
59	EBFONT	Bad font file format
60	ENOSTR	Device not a stream

TAB. 5.1 – Suite (2).

numéro	définition en C	Message
61	ENODATA	No data available
62	ETIME	Timer expired
63	ENOSR	Out of stream resources
64	ENONET	Machine is not on the network
65	ENOPKG	Package not installed
66	EREMOTE	Object is remote
67	ENOLINK	Link has been severed
68	EADV	Advertise error
69	ESRMNT	Srmount error
70	ECOMM	Communication error on send
71	EPROTO	Protocol error
72	EMULTIHOP	Multihop attempted
73	EDOTDOT	RFS specific error
74	EBADMSG	Bad message
75	EOVERFLOW	Value too large for defined data type
76	ENOTUNIQ	Name not unique on network
77	EBADFD	File descriptor in bad state
78	EREMCHG	Remote address changed
79	ELIBACC	Cannot access a needed shared library
80	ELIBBAD	Accessing a corrupted shared library
81	ELIBSCN	.lib section in an a.out corrupted
82	ELIBMAX	Attempting to link in too many shared libraries
83	ELIBEXEC	Cannot exec a shared library directly
84	EILSEQ	Invalid or incomplete multioctet or wide character
85	ERESTART	Interrupted system call should be restarted
86	ESRTPIPE	Streams pipe error
87	EUSERS	Too many users
88	ENOTSOCK	Socket operation on non-socket
89	EDESTADDRREQ	Destination address required
90	EMSGSIZE	Message too long
91	EPROTOTYPE	Protocol wrong type for socket
92	ENOPRO-TTOOPT	Protocol not available
93	EPROTO-NOSUPPORT	Protocol not supported
94	ESOCKT-NOSUPPORT	Socket type not supported
95	EOPNOTSUPP	Operation not supported
96	EPFNOSUPPORT	Protocol family not supported
97	EAFNOSUPPORT	Address family not supported by protocol
98	EADDRINUSE	Address already in use
99	EADDRNOTAVAIL	Cannot assign requested address
100	ENETDOWN	Network is down

TAB. 5.1 – Suite (3).

numéro	Définition en C	message
101	ENETUNREACH	Network is unreachable
102	ENETRESET	Network dropped connexion on reset
103	ECONNABORTED	Software caused connexion aborted
104	ECONNRESET	connexion reset by peer
105	ENOBUFS	No buffer space available
106	EISCONN	Transport endpoint is already connected
107	ENOTCONN	Transport endpoint is not connected
108	ESHUTDOWN	Cannot send after transport endpoint shutdown
109	ETOOMANYREFS	Too many references : cannot splice
110	ETIMEDOUT	connexion time out
111	ECONNREFUSED	connexion refused
112	EHOSTDOWN	Host is down
113	EHOSTUNREACH	No route to host
114	EALREADY	Operation already in progress
115	EAINPROGRESS	Operation now in progress
116	ESTALE	Stale NFS file handle
117	EUCLEAN	Structure needs cleaning
118	ENOTNAM	Not a XENIX names type file
119	ENAVAIL	No XENIX semaphore available
120	EISNAM	Is a named type file
121	EREMOTEIO	Remote I/O error
122	EDQUOT	Disk quota exceeded
123	ENOMEDIUM	No medium found
124	EMEDIUMTYPE	Wrong medium file

5.3 Caractères de remplacement, noms, extensions et motifs d'englobements.[1]

La liste produite par `ls` peut être très longue s'il y a un grand nombre de fichiers dans un répertoire. Pour l'heure, disons que nous sommes intéressés seulement par les fichiers se terminant par les lettres `ttre`. Pour afficher exclusivement ces fichiers, vous pouvez utiliser `ls *ttre`. Le signe * correspond à un nombre quelconque de caractères à priori quelconques. Ainsi, par exemple, les fichiers `Tina.lettre`, `Mary_Jones.lettre` et le fichier `mettre` seront affichés s'ils sont présents ; tandis qu'`Harlette` ne le sera pas. Alors que * correspond à un nombre quelconque de caractères, ? remplace un seul caractère. Par exemple, la commande `ls ?ar*` listera les fichiers `Mary_Jones.lettre` et `Harlette`.

5.3.1 Noms de fichiers.

Lorsqu'on nomme un fichier, il est toujours de bon aloi de choisir un nom qui regroupe des fichiers de même type. Vous réalisez cela en ajoutant une extension au nom de fichier qui décrit le type de fichier. Nous avons déjà fait çà

[1] NdT : le terme "englobement" est un néologisme. Il correspond à *glob expressions*, en anglais.

en créant le fichier `Mary_Jones.lettre` plutôt que `Mary_Jones`. Si vous gardez cette convention, il vous sera plus aisé d'afficher tous les fichiers de type "lettres" en entrant `ls -*.lettre`. Le nom de fichier `Mary_Jones.lettre` est décrit comme étant composé de deux parties : le *nom*, `Mary_Jones` et l'*extension* `.lettre`.

Vous trouverez ci-dessous certaines extensions classiques d'UNIX :

.a Archives. Le terme `lib*.a` désigne une bibliothèque statique.

.alias Catalogue d'alias de polices de caractères du système X-Window.

.avi Format vidéo.

.au Format audio (fichier son générique de Sun Microsystems).

.awk Fichiers source de programmation `awk`.

.bib Fichiers source de bibliographie LaTeX `bibtex`.

.bmp Format d'image Bitmap Microsoft.

.bz2 Fichiers compressés avec le programme de compression `bzip2`.

.cc, .cxx, .cpp Codes source de programmation C++.

.cf, .cfg Scripts ou fichiers de configuration.

.cgi Scripts exécutables qui produisent un affichage sous forme de pages web.

.conf, .config Fichiers de configuration.

.csh Scripts shell de type `csh.`

.c Codes source de programmation **C**.

.db Fichiers de base de données.

.dir Répertoire de polices de caratères du système X-Window.

.deb Paquets Debian pour la distribution Debian.

.diff résultats du programme `diff` indiquant la différence des fichiers, des arborescences de fichiers ou de sources.

.dvi Fichiers indépendants du périphérique (*Device-independent File*). Sortie formatée des fichiers LaTeX `.tex`.

.el Sources de programmes Lisp.

.g3 Fichiers d'images au format de fax G3.

.gif, .giff Fichiers d'images GIF

.gz Fichiers compressés avec l'utilitaire de compression `gzip`.

.htm, html, .shtm, .html Langage à balises de type Hypertext (*HyperText Markup Language*). Type de page web.

.h Fichiers d'en-tête des programmes **C**/C++.

.i Sources SWIG, ou résultat produit par le préprocesseur **C**.

.in Fichiers d'entrée `configure`.

.info Pages d'info lue avec la commande `info`.

.jpg, .jpeg Fichiers image JPEG.

.lj Fichiers LaserJet. Fichier approprié à une imprimante HP LaserJet.

.log Fichiers de journalisation d'un service du système. La taille de ce type de fichier augmente avec les messages de divers programmes du système.

.lsm Entrée de LINUX Software Map.

.lyx Documents du traitement de texte LyX.

.man Pages de manuel (man).

.mf Fichiers source du programme de polices de caractères Meta-Font

.ogg Format de compression de fichiers audio

.pbm Format de fichier d'images PBM.

.pcf Fichiers d'images PCF – représentation intermédiaire pour les polices. Polices du système X Window.

.pcx Fichiers d'images PCX.

.pfb Fichiers de police du système X-Window.

.pdf Documents formatés de manière similaire à PostScript ou dvi.

.php Codes source de programmation PHP (utilisé pour l'élaboration de pages web dynamiques).

.pl Codes source de programmation Perl.

.ps Fichiers PostScript, pour l'impression ou la visualisation.

.py Codes source de programmation Python.

.rpm Fichiers `rpm` du gestionnaire de paquet RedHat (*RedHat Package Manager*).

.sgml Langage à balises généralisé normé (*Standard Generalized Markup Language*). Utilisé pour créer des documents convertibles en de nombreux formats différents.

.sh Scripts de shell `sh`.

.so Fichiers objet partagé. Le terme `lib*.so` désigne une bibliothèque liée dynamiquement [c'est -à-dire, le code exécutable d'un programme, partagé par plus d'un programme en vue de sauver de l'espace disque et de la mémoire].

.spd Fichiers de polices de caractères du système X-Window Speedo.

.tar Arborescence de répertoire archivée à l'aide de la commande `tar`.

.tcl Codes source de programmation Tcl/Tk.

.texi, .texinfo Sources Texinfo. Les pages d'info sont compilées à partir de ces sources.

.tex Documents LaTeX ou TeX. LaTeX est un traitement de texte et de typographie.

.tga Fichiers images TGA.

.tgz Arborescence de répertoires qui a été archivée à l'aide de `tar` puis compressée avec `gzip`. Egalement un type de paquets de la distribution Slackware.

.tiff Fichier d'images de type TIFF.

.tfm Fichier de polices métriques de LaTeX.

.ttf Polices de caractères Truetype.

.txt Fichiers texte.

.voc Format audio (format propriétaire de Soundblaster).

.wav Format audio (fichiers sons courants de MS-Windows).

.xpm Fichers d'images XPM.

.y Fichier source `yacc`.

.Z Fichiers compressés avec l'utilitaire de compression `compress`.

.zip Fichiers compressés avec le programme de compression `pkzip` (ou PKZIP.EXE pour DOS).

.1, .2 ... Page de man.

En outre, des fichiers qui n'ont pas d'extension mais une description en lettre majuscule sont usuellement écrits en anglais et sont destinés à ce que vous les lisiez. Ils sont associés à des paquets logiciels et servent de documentation. Vous les verrez assez fréquemment.

Parmi les plus utiles, vous trouverez :

AUTHORS Liste de personnes ayant contribué ou ayant écrit un paquet logiciel.

ChangeLog Liste de modifications de développeurs efectuées sur un paquet.

COPYING Droits d'auteur (usuellement GPL) associés à un paquet.

INSTALL Instructions d'installation.

README Information d'aide immédiate incluse dans le répertoire principal du paquet de logiciels.

TODO Liste de travaux à effectuer sur un logiciel.

BUGS Listes d'erreurs.

NEWS Information relative aux nouveautés et au changement pour l'installation.

THANKS Liste des contributeurs à un paquet logiciel.

VERSION Information sur la version d'un paquet.

5.3.2 Motifs d'englobement.

Il existe une méthode pour limiter l'affichage des fichiers dans une gamme de caractères. Si vous désirez seulement afficher les fichiers qui commencent par A jusqu'à M, vous utiliserez `ls -[A-M]*`. Ici, les crochets ont une signification particulière : ils correspondent à un simple caractère `?` mais seulement dans la gamme indiquée (en l'occurrence, de A à M). Vous pouvez utiliser cette notation de différentes manières. Par exemple, `[a-dJW-Y]*` correspond à tous les fichiers commençant par `a`, `b`, `c`, `d`, `J`, `W`, `X` ou `Y`; et `*[a-d]id` correspond à tous les fichiers se terminant par `aid`, `bid`, `cid` ou `did`. Par ailleurs, `*.{cpp,c,cxx}` correspond à tous les fichiers se terminant par `.cpp`, `.c` ou `.cxx`. Spécifier des noms de fichiers par cette méthode consiste à utiliser des *motifs d'englobement*. Les motifs d'englobement sont utilisés dans diverses situations, comme nous aurons l'occasion de le voir plus tard.

5.4 Syntaxe – la commande `copy`.

La commande `cp` signifie copier (*to copy*). Elle duplique un ou plusieurs fichiers. Son format d'utilisation est :

cp <fichier> <nouveau_fichier>
cp <fichier> [<fichier> | ...] <répertoire>

ou encore

cp *fichier nouveau_fichier*
cp *fichier* [*fichier* ...] *répertoire*

Les lignes ci-dessus sont appelées *résumé d'utilisation*. Les signes < et > ne doivent pas être saisis mais bel et bien remplacer `<fichier>` par le nom de votre fichier. Les résumés d'utilisation sont parfois exprimés en italique sous la forme : `cp fichier nouveau_fichier`. Dans de rares cas, ils sont écrits en lettres capitales : `cp FICHIER NOUVEAU_FICHIER`. <fichier> et <répertoire> sont des *paramètres* ou *arguments* de commande. Parfois, ils sont numériques comme dans le cas d'une commande qui concernerait `<ioport>` [...]. Il y a des conventions communes pour spécifier l'usage d'une commande. Les crochets `[` et `]` ne sont pas usuellement saisis mais ils signifient que leur contenu est optionnel. Les points de suspension veulent dire que `<fichier>` peut être mentionné de manière répétée. Ils ne sont jamais saisis en réalité. Désormais, vous êtes censé substituer vos propres paramètres en interprétant le résumé d'utilisation d'une commande. Vous pouvez constater que la seconde ligne de la commande décrite ci-dessus montre qu'un ou plusieurs fichiers peuvent être affichés avant de donner le nom d'un répertoire.

De ce qui vient d'être dit, on déduit qu'il y a 2 méthodes d'utilisation de la commande `cp`. Si le dernier nom n'est pas un répertoire, `cp` copie le fichier et le renomme sous la forme du nom que vous avez indiqué en dernière position. Si le dernier nom est un répertoire, `cp` copie tous les fichiers en argument dans le répertoire.

Le résumé d'utilisation de la commande `ls` est décrit ainsi :

```
ls [-l, --format=long] [-a, --all] <file> <file> ....
ls -al
```

où la virgule indique que l'une ou l'autre option se vaut. De la même manière, pour la commande `passwd` :

```
passwd [<nom_utilisateur>]
```

A présent , vous devriez vous exercer avec la commande `cp` de manière à déplacer vos fichiers.

5.5 Manipulation de répertoires.

La commande `cd` est utilisée pour se déplacer parmi les différents répertoires. Créez un répertoire nommé `nouveau` à l'aide de la commande `mkdir nouveau`. Vous *pourriez* créer un répertoire `alpha` dans `nouveau` en faisant `cd nouveau` puis `mkdir alpha`, mais il y a un moyen plus direct : `mkdir nouveau/alpha`. Vous pouvez alors vous déplacer directement dans `alpha` en faisant `cd nouveau/alpha`. Inversément, la commande `cd ../..` vous permet de revenir en arrière

dans l'arborescence. Ainsi, vous voyez que `/` représente le passage d'un répertoire à un autre. On parle du répertoire `alpha` comme d'un *sous-répertoire* de `nouveau`.

La commande `pwd` permet de retourner le *present working directory* –c'est-à-dire le répertoire dans lequel vous êtes en train de travailler– ou répertoire courant. Cette commande vous permet de déterminer votre position dans l'arborescence. Expérimentez cela en vous déplaçant dans le répertoire racine (ou root) avec la commande `cd /` et revenez dans `/home/<nom_utilisateur>` (avec `cd /home/<nom_utilisateur>`). Le répertoire `/home/<nom_utilisateur>` est connu sous le nom de *répertoire personnel* (ou *home directory*) et est désigné par l'abréviation `~`. En d'autres termes, exécuter : `cd /home/<nom_utilisateur>` équivaut à faire `cd ~`. Le processus par lequel un `~` est remplacé par votre répertoire personnel est appelé *développement de tilde* (ou *expansion tilde* en anglais).

Pour éliminer (c'est-à-dire, écraser ou supprimer) un fichier, utilisez la commande `rm <nom_de_fichier>`. Pour enlever un répertoire, utilisez la commande `rmdir <répertoire>`. Testez ces deux commandes et notez que vous ne pourrez supprimer un répertoire que s'il est vide. Pour supprimer un répertoire avec tout ce qu'il contient, utilisez la commande `rm -R <répertoire>`. L'option `-R` précise qu'il faut descendre dans tous les sous-répertoires de `<répertoire>` et supprimer leur contenu. Le processus par lequel une commande est exécutée en passant de sous-répertoire en sous-répertoire est appelé *récursion*. `-R` signifie *récursivement*. C'est une commande très dangereuse. Bien que sur d'autres systèmes, il soit aisé de récupérer des fichiers ou des répertoires supprimés, sur UNIX, la récupération de fichiers est, au mieux, compliquée.

La commande `cp` possède aussi l'option `-R`, ce qui permet de copier tout un répertoire. La commande `mv` est utilisée pour déplacer des fichiers ou des répertoires. Elle permet également de renommer des fichiers. Notez que vous pouvez aussi utiliser les options `-p` et `-d` avec `-R` pour préserver les attributs des fichiers et reproduire proprement les *liens symboliques* (*symlinks*; ce point discuté au chapitre 16). De préférence, utilisez `cp -dpR <répertoire> <nouveau_répertoire>` au lieu de `cp -R <répertoire> <nouveau_répertoire>`.

5.6 Chemins absolus et relatifs.

Les commandes peuvent s'appliquer avec en argument des noms de fichiers donnés sous deux formes. Si vous êtes dans le même répertoire que le fichier sur lequel vous souhaitez agir (c'est-à-dire que ce fichier est dans le répertoire où vous vous êtes déplacé), il vous est loisible de n'entrer que le nom de fichier (par exemple, `cp mon_fichier nouveau_fichier`). Autrement, vous pouvez entrer le nom complet du chemin pour accéder à ce fichier, comme dans `cp /home/linus/mon_fichier /home/linus/nouveau_fichier`. Très souvent, les administrateurs utilisent la notation suivante : `./mon_fichier` pour réaliser clairement la distinction comme dans `cp ./mon_fichier ./nouveau _fichier`. Le "préfixe" `./` rend clair le fait que sont traités 2 fichiers relatifs au répertoire courant. Les noms ne débutant pas par `/` sont dits des noms de *chemins relatifs* par opposition aux noms de *chemins absolus*.

5.7 Les pages du manuel.

Note : vous devriez consulter le chapitre 17 pour avoir une vision complète de toute la documentation du système, et aussi savoir comment imprimer les pages de manuel dans un format correct.

La commande `man [<section> | -a] <commande>`fournit une aide sur un sujet particulier et, son nom est une contraction pour *manuel*. Chaque commande du système est documentée dans les *pages de man*.[2] Au cours des dernières années est apparu un nouveau format de documentation : les pages d'*info*. Celles-ci sont considérées comme un moyen moderne de description des commandes. Toutefois, la majorité de la documentation du système est seulement disponible, à l'aide de `man`.

Les pages de man constituent une référence faisant autorité quant au fonctionnement des commandes parce que ces pages sont écrites par les développeurs des commandes, eux-mêmes. Sous UNIX, toute documentation imprimée sera considérée comme de "deuxième main". En revanche, les pages de man ne contiennent souvent pas les notions de base nécessaires pour comprendre le contexte d'utilisation d'une commande. Donc, il n'est pas possible à une personne d'apprendre UNIX à partir des seules pages de man. Cependant, une fois que vous avez acquis les bases nécessaires, les pages du manuel deviennent une indispensable source d'information et vous pourrez écarter progressivement le matériel introductif.

A l'heure actuelle, les pages de man sont divisées en sections numérotées de 1 à 9. La section 1 contient toutes les pages de man pour les commandes du système telles que celles que nous avons déjà utilisées. Les sections 2 à 7 contiennent de l'information pour les développeurs. Vous n'y recourrez pas de suite. La section 8 contient des pages relatives aux commandes d'administration. Il y a des pages supplémentaires étiquetées par des lettres ; mais en dehors de celles-ci, il n'y a pas de pages de manuel autres que celles des sections de 1 à 9. Les sections se présentent ainsi :

Section	Type d'information
.../man1	Programmes d'utilisateurs
.../man2	Appels système
.../man3	Appels de bibliothèques
.../man4	Fichiers spéciaux
.../man5	Formats de fichiers
.../man6	Jeux
.../man7	Divers
.../man8	Administration système
.../man9	Documentation du noyau

A présent, vous devriez utiliser la commande `man` pour naviguer dans les pages de man associées aux commandes que nous avons utilisées. Tapez `man cp`, `man mv`, `man rm`, `man mkdir`, `man rmdir`, `man passwd`, `man cd`, `man pwd` et bien sûr `man man`. Une grande partie de l'information contenue dans ces pages peut encore vous être incompréhensible pour l'instant. Néanmoins, naviguez parmi les pages pour en acquérir la structure et savoir sous quel format elles se présentent.

[2] Par exemple : `man 1 -a cp` affiche les explications sur l'usage de la commande `cp`.

Les pages de man sont référencées avec une notation telle que `cp(1)`, pour ce qui concerne la commande `cp` dans la section 1. On peut obtenir des informations sur `cp` en tapant `man 1 cp`. Désormais, cette notation sera utilisée.

5.8 Les pages d'`info`.

Les pages d'`info` contiennent d'excellentes références et de l'information tutorielle selon un format de type hypertexte. Saisissez `info` tel quel pour obtenir le menu général d'`info`. Vous pouvez également taper `info <commande>` pour obtenir de l'aide à propos des nombreuses commandes de base. Cependant, certains paquets logiciels n'ont pas de pages d'info et d'autres systèmes UNIX ne contiennent pas de pages d'info du tout.

`info` est un programme interactif avec des touches de navigation et de la documentation d'aide. A l'intérieur de ce programme, si vous pressez  , cela appelera un écran d'aide à partir duquel vous pourrez apprendre davantage de commandes.

5.9 Commandes fondamentales.

Vous pouvez vous entraîner à utiliser les commandes suivantes :

bc est un programme de calcul manipulant des nombres à grande précision. Il est utile pour réaliser tout type de calcul en ligne de commande. Son utilisation vous est laissée à titre d'exercice.

cal [[0-12] 1-9999] retourne un calendrier agréablement formaté, du mois courant, d'un mois sélectionné, ou des mois d'une année donnée. Essayez `cal 1` pour le plaisir et `cal 9 1752`, année où le Pape a effacé quelques jours pour compenser une erreur d'arrondi.

cat <nom de fichier> [<nom de fichier> ...] écrit le contenu des tous les fichiers listés à l'écran. `cat` peut regrouper des fichiers lorsque l'expression suivante est utilisée : `cat <fichier1> <fichier2> ... > <nouveau_fichier>`. Le fichier intitulé `<nouveau_fichier>` contiendra le résultat d'une concaténation (c'est-à-dire une mise bout-à-bout) du contenu des fichiers donnés en argument.

clear nettoye le texte du terminal courant.

date retourne à l'écran la date et l'heure. (La commande `time` exécute un programme entièrement différent).

df correspond à *disk free* et vous indique la quantité d'espace libre sur votre système. L'espace disponible est usuellement exprimé en kilooctets (1024 octets) (bien que sur certains systèmes UNIX les unités sont de 512 ou 2048 octets). La colonne la plus à droite mentionne les répertoires avec leur espace libre correspondant, dans la colonne juste à côté.

dircmp compare les répertoires. Cette commande compare les répertoires entre eux pour déterminer si des changements ont eu lieu. Vous devrez souvent déterminer par quels sous-répertoires ou fichiers se distinguent deux arborescences différentes sur deux ordinateurs différents (par exemple pour tester s'il y a des fichiers manquants). Exécutez la commande `man dircmp`

(c'est-à-dire `dircmp`(1)). (Ceci est une commande du System V et, en principe, elle n'est pas présente sur LINUX. Vous pouvez néanmoins comparer des répertoires grâce à Midnight Commander, `mc`).

du <directory> correspond à *disk usage* et renvoie la quantité d'espace occupé par un répertoire. Cette commande est appliquée récursivement dans tous les sous-répertoires du répertoire désigné en argument. `du -s <répertoire>` peut ne renvoyer qu'un résumé. Testez aussi :
`du - -max-depth=1 /var` et `du -x /` sur un système avec `/usr` et `/home` sur des partitions séparées [voir la section 19.3].

dmesg renvoie à l'écran le journal des messages affichés durant la phase d'initialisation du système. Pour l'instant, ces messages ne vous sont peut-être pas encore explicites.

echo affiche un message au terminal. Essayez `echo 'bonjour'`, `echo $[10*3+2]`, `echo '$[10*3+2]'`. La commande `echo -e` permet l'interprétation de certaines séquences incluant un *backslash*. Par exemple, `echo -e "\a"` affiche une *cloche* ou fait émettre un son *via* votre terminal. `echo -n` fait la même chose mais sans pratiquer de saut de ligne. En fait, elle ne provoque pas de saut entre la ligne de saisie et l'affichage de votre invite. `echo -e -n "\b"` applique un retour en arrière sur une lettre (*back-space*), ce qui écrase le dernier caractère imprimé.

exit permet la déconnexion de la session en mode console et ferme un terminal en mode graphique tout en vous laissant dans votre session.

expr <expression> calcule l'expression numérique `expression`. La plupart des opérations arithmétiques que vous connaissez seront effectuées. Essayez `expr 5 + 10 '*' 2`. Notez les blancs entre les paramètres et la préséance des opérations (* est exécuté avant +).

file <fichier> affiche le type de données contenues dans un fichier. `file portrait.jpg` vous dira que c'est une `donnée image JPEG, standard JFIF`. La commande `file` détecte une quantité très importante de types de fichier, sur chaque plate-forme. `file` fonctionne en vérifiant que les premiers octets d'un fichier correspondent aux séquences d'octets appelés *nombres magiques*. La liste complète de ces nombres est stockée dans /usr/share/magic. [Le terme "nombre magique" sous UNIX désigne normalement les séquences d'octets ou de nombres qui ont une signification déterminée. Ces nombres magiques sont créés pour le code source, les formats de fichiers et le système de fichiers].

free renvoie à l'écran la mémoire libre disponible. Vous noterez deux listes : l'espace de swap et la mémoire physique. Elle sont contiguës pour l'utilisateur. L'espace de swap est une extension de la mémoire, extension installée sur le disque. Bien sûr, son accès est lent par rapport celui de la mémoire vive, mais elle fournit l'illusion de plus de mémoire RAM disponible. Elle évite la possibilité d'un dépassement de mémoire RAM (ce qui est "fatal").

head [-n <lignes>] <fichier> affiche la liste des n premières lignes d'un fichier ou les 10 premières lignes par défaut si l'option `-n` n'est pas renseignée (voir aussi la commande `tail`).

hostname [<nouveau_nom>] sans options, `hostname` affiche le nom de votre machine. Autrement, elle remplace le nom de machine par `<nouveau_nom>`.

kbdrate -r <caractères-par-seconde> -d <délai-de-répétition> modifie la vitesse de répétition des touches du clavier. En majorité, les utilisateurs mettent la valeur de cette vitesse à `kbdrate -r 32 -d 250` qui est la vitesse la plus élevée des claviers de PCs.

more est un afficheur (*pager* en anglais) qui formate à l'écran un fichier par page entière (sans défilement). Exécutez la commande `ls -l /bin > bin-ls` et, ensuite `more bin-ls`. La première commande crée le fichier `bin-ls` qui contient la sortie de `ls` exécutée sur le répertoire `/bin`. Ceci donnera une liste suffisament longue pour ne pas tenir sur un seul écran car `/bin` contient de nombreuses entrées. La seconde commande permet de visualiser le fichier `bin-ls`. Utilisez la barre d'espacement pour passer d'une page à l'autre. Pour sortir, pressez Q. Vous pouvez aussi tester `ls -l /bin | more` qui réalisera la même opération, mais en une seule commande composée.

less est la version GNU de `more`, mais avec des caractéristiques spéciales. Sur votre système les deux commandes sont peut-être identiques. Avec `less`, vous pouvez utiliser les touches fléchées pour naviguer dans les pages. Vous pouvez également faire des recherches en appuyant sur ?, en tapant un mot à rechercher puis en pressant Enter. Le mot trouvé sera mis en évidence, et le texte sera parcouru à partir du premier mot trouvé. Les commandes importantes sont :

- Shift-G qui permet de se déplacer à la fin du fichier,
- Shift-?*ssss* qui recherche en arrière le texte *ssss* dans un fichier,
- /*ssss* qui recherche en avant le texte *ssss* dans un fichier, [en réalité, *ssss* constitue un expression régulière. Voir le chapitre 6],
- Shift-F qui permet de remonter en arrière dans le texte. Cette commande est utile dans le cas des journaux (*logs*).
- *nnn*-G qui permet de déplacer le curseur à la ligne *nnn*,
- Q qui permet de quitter le pageur (ou *pager* en anglais). Cette commande est abondamment utilisée dans les applications-textes de UNIX (parfois sous la forme Q-Enter),
  (Vous pouvez faire en sorte que `less` ne produise pas l'avertissement sonore quelque peu agaçant, en éditant le fichier `/etc/profile`, en ajoutant les lignes :

```
LESS=-Q
export LESS
```

Ensuite, déconnectez-vous et reconnectez-vous. Néanmoins, ceci est un aparté qui prendra son sens plus tard).

lynx <url> ouvre une URL dans une console [une URL est une adresse web nommée *Uniform Ressource Locator*],

links <url> `links` est un autre type de navigateur web en mode console.

nohup <commande> & exécute une commande en arrière-plan. La commande produit le fichier `nohup.out` dans votre répertoire home.[3] `nohup` présente une caractéristique intéressante : elle continue de travailler même après que vous soyez déconnecté. L'usage de la commande `nohup` deviendra évident plus tard.

sleep <secondes> réalise une pause de `<secondes>` secondes. Voir aussi la commande `usleep`.

sort <fichier> affiche le contenu d'un fichier avec les lignes éditées par ordre alphabétique. Créez un fichier appelé `telephone`, chaque ligne contenant une courte entrée d'un annuaire téléphonique. Ensuite tapez `sort telephone`, ou `sort telephone | less` et constatez ce qui se produit. `sort` présente beaucoup d'options intéressantes comme effectuer un affichage en ordre inverse (`sort -r`), éliminer des entrées multiples (`sort -u`), ignorer les espaces blancs en début de lignes (`sort -b`), etc. Voyez `sort`(1) pour plus de détails.

strings [-n <len>] <fichier> retourne un fichier binaire, mais en ôtant tout caractère illisible. Les groupes de caractères lisibles sont placés sur des lignes séparées. Si vous avez un fichier binaire dont vous pensez qu'il contient des informations intéressantes mais de manière complètement illisibles lorsque vous l'affichez normalement, utilisez `strings` pour en extraire les informations importantes : essayez `less /bin/cp` et ensuite `strings /bin/cp`. Par défaut, `strings` ne retourne pas les séquences de taille inférieure à 4. L'option `-n` étend cette limite.

split ... divise un fichier en plusieurs fichiers séparés. Cette commande peut être utilisée lorsqu'un fichier est trop long pour être copié directement sur une disquette. Il est alors nécessaire de le morceler en fichiers d'1 Mo, par exemple. Sa commande soeur `csplit` peut diviser un fichier selon des lignes spécifiques du texte. Ces commandes sont parfois utilisées telles quelles, mais aussi, le plus souvent, par des programmes qui manipulent des textes.

tac <fichier> [<fichier> ...] écrit le contenu de tous les fichiers, à l'écran en renversant l'ordre des lignes –c'est-à-dire en affichant la dernière ligne en premier. `tac` est `cat` écrit à l'envers et se comporte comme tel.

tail [-f] [-n <lignes>] <fichier> affiche les `<lignes>` lignes d'un fichier ou les 10 dernières lignes par défaut si l'option `/-n` n'est pas donnée. L'option `-f` signifie "surveiller le fichier pour les lignes qui ont été ajoutées à la fin du fichier" (voir aussi `head` ci-dessus),

uname imprime le nom du *système d'exploitation* UNIX utilisé. Dans ce cas : LINUX.

uniq <fichier> imprime un fichier de manière à ce que les lignes multiples soient supprimées. Le fichier doit d'abord être trié.

usleep <microsecondes> fait une pause de `<microsecondes>` microsecondes [1/1.000.000 de seconde],

wc [-c] [-w] [-l] <fichier> compte le nombre d'octets (avec `-c` pour caractère), ou de mots (avec `-w`) ou de lignes (avec `-l`) dans un fichier,

[3] Si, par exemple, vous effectuez `nohup tar cvf home.tar * &` le fichier `nohup.out` contiendra la liste des fichiers archivés. Pour la commande `tar`, voir la section 5.13.

whatis <**commande**> donne la première ligne de la page de man correspondant à `<commande>`, sauf si la page n'existe pas ; dans ce cas, le message affiché est `nothing appropriate`,

whoami retourne à l'écran votre identifiant.

5.10 Le gestionnaire de fichiers `mc`.

Ceux qui viennent du monde DOS connaissent le gestionnaire de fichiers *Norton Commander*. Le projet GNU a un clône libre appelé *Midnight Commander*, `mc`. Il est essentiel d'au moins essayer ce paquet car il permet de se déplacer de manière extrêmement rapide parmi les fichiers et les répertoires, donnant ainsi une vue d'ensemble. Son utilisation réduira considérablement les commandes pénibles qu'il faudrait saisir manuellement.

5.11 Commandes multimedia.

Si vous avez une carte son correctement configurée, vous devriez utiliser les commandes suivantes. [Il ne s'agit pas de donner l'impression que LINUX n'a pas d'applications graphiques pour réaliser toutes les fonctions mentionnées dans cette section, mais vous devez être conscient que pour chaque application graphique, il y a des commandes en mode texte qui fonctionnent très bien et qui consomment peu de ressources]. Il se peut que certains des paquets ne soient pas installés ; dans ce cas, vous pouvez revenir à ce paragraphe plus tard.

play [-v <**volume**>**]** <**fichier**> joue des formats audio linéaires via la carte-son. Les formats sont `.8svx`, `.aiff`, `.au`, `.cdr`, `.cvs`, `.dat`, `.gsm`, `.com`, `.maud`, `.ogg`, `.sf`, `.smp`, `.txw`, `.vms`, `.voc`, `.wav`, `.wve`, `.raw`, `.ub`, `.sb`, `.uw`, `.sw` ou `.ul`. En d'autres termes, presque tous les fichiers sont utilisables. Le plus souvent, il s'agira de fichiers `.wav`. Indiquez `<volume>` en pourcents.

rec <**fichier**> permet l'enregistrement sous forme de fichiers *via* un microphone (`play` et `rec` viennent du même paquet logiciel).

mpg123 <**fichier**> joue la musique de fichiers audio MPEG (niveau 1, 2 ou 3). Les options utiles sont `-b 1024` (pour augmenter la taille du tampon afin de prévenir les sauts) et `--2to1` (pour sous-échantillonner d'un facteur 2 afin de réduire la charge de la CPU). Les fichiers MPEG contiennent du son et/ou de la vidéo, de manière très compacte en utilisant des techniques de traitement digital du signal que l'industrie du logiciel commercial semble penser très sophistiquées.

cdplay joue un fichier musical à partir d'un CD-audio. `cdp` est la version interactive

aumix permet d'ajuster le volume, le gain, le volume d'enregistrement, etc. de votre carte son. Vous pouvez l'utiliser intéractivement ou entrez `aumix -v <volume>` pour fixer directement le volume en pourcents. Notez que ceci est un programme *mixeur* dédié et qu'il est considéré comme une application séparée de celles permettant de jouer de la musique. De préférence, évitez de régler le volume à partir d'une application musicale en cours, même si

cette dernière affirme que vous pouvez le faire ; `aumix` permet un meilleur contrôle.

mikmod - -interpolate -hq - -renice Y <**fichier**> joue des fichiers *Mod*. Ces fichiers sont un type spécial de fichiers audio qui stockent seulement la durée et la hauteur des notes qui constituent une chanson, avec des échantillons de chaque instrument de musique nécessaire à la chanson. Ceci rend une très haute-qualité audio à partir d'un fichier d'une taille extraordinairement petite. `mikmod` supporte les formats audio 669, AMF, DSM, FAR, GDM, IMF, IT, MED, MOD, MTM, S3M, STM, STX, ULT, UNI et XM -ce qui constitue probablement les types existant actuellement. En fait, un grand nombre de fichiers est disponible sur internet au format de fichier Mod. Les formats les plus communs sont `.it`, `.mod`, `.s3m` et `.xm`. [Les fichiers originaux .mod sont le produit des ordinateurs Commodore-Amiga et ont seulement 4 pistes. Actuellement, les fichiers Mod à 16 pistes (ou plus) sont comparables à tous les fichiers musicaux enregistrés].

5.12 Commandes de terminaison.

Vous utiliserez Ctrl-C pour arrêter une application ou une commande en cours d'exécution. Vous devez appuyer sur ces touches dans le même terminal que celui où a été lancée l'application (ou la commande). Si cette méthode ne fonctionne pas, la section processus (section 10.5) explique comment *identifier* une application en cours qu'il devient nécessaire de quitter.

5.13 Fichiers compressés.

Les fichiers contiennent de nombreuses données et on peut aisément imaginer que celles-ci puissent être représentées par un plus petit nombre d'octets. Prenez le cas d'une de vos lettres. Le mot "le" est probablement répété plusieurs fois. Vous utilisez très probablement les minuscules à de nombreuses reprises. De loin, le fichier de votre lettre n'est pas constitué d'un jeu d'octets aléatoires : il utilise des espaces et d'autres caractères de manière plus abondante que d'autres. [Un texte en français, par exemple, contient en moyenne seulement 1,3 bits de données réelles par octet (il y a 8 bits dans un octet)]. De ce fait, le fichier de votre lettre peut être compressé afin d'occuper moins d'espace. La compression implique la représentation de mêmes données en utilisant un petit nombre identique d'octets de manière telle que les données originales puissent être reconstituées exactement. Il faut donc trouver des motifs dans les données. La commande utile pour compresser un fichier est `gzip <fichier>`, qui signifie *GNU zip*. Exécutez `gzip` sur un fichier de votre répertoire home et appliquez `ls` pour visualiser ce qui s'est produit. A présent, utilisez `less` pour voir le fichier compressé. Pour décompresser le fichier, utilisez `gzip -d <fichier>` ou `gunzip <fichier>`. A nouveau, utilisez `less` pour visualiser le fichier. De nombreux fichiers sur votre système se trouvent sous forme compressée. Par exemple, les pages de man sont souvent compressées et décompressées automatiquement lors de leur appel.

Précédemment, vous avez utilisé la commande `cat` pour lire le contenu d'un fichier. Vous pouvez employer la commande `zcat` pour un même résultat avec

un fichier compressé. Gzippez un fichier et ensuite lisez son contenu en saisissant `zcat <fichier>`. Vous verrez le contenu du fichier à l'écran. En général, lorsque les commandes et les fichiers possèdent un z, elles ont à voir avec la compression, la lettre `z` étant apparentée à *zip*. Vous pouvez utiliser `zcat <fichier> | less` pour lire le contenu d'un fichier de manière conviviale. Vous pouvez aussi utiliser la commande `zless <fichier>` comme équivalent de `zcat <fichier> | less`. Notez que votre `less` peut déjà posséder la fonctionnalité `zless`.

Une autre commande fait partie de l'arsenal : `bzip2`. C'est un programme de compression analogue à `gzip`, excepté qu'il est un peu plus lent et qu'il compresse avec une efficacité de 20 à 30 % supérieure à gzip. Il est utile pour compresser des fichiers provenant d'internet afin de réduire le volume de transfert. Les fichiers compressés avec `bzip2` ont l'extension `bz2`. Notez que l'amélioration de la compression dépend beaucoup du type de données à compresser. Parfois, le taux de compression est faible quoique se faisant au prix d'une certaine lenteur ; parfois, la compression est tout-à-fait avantageuse. Pour les fichiers devant être compressés/décompressés souvent, évitez `bzip2`.

5.14 Recherche de fichiers.

Vous pouvez faire usage de la commande `find` pour rechercher des fichiers. Passez dans le répertoire racine (root ou encore /) et tapez `find`. Cette commande sonde tous les fichiers en descendant récursivement dans tous les sous-répertoires [la commande va dans les répertoires, les sous-répertoires et leurs sous-répertoires de rang inférieur tout en répétant `find`]. En d'autres mots, `find`, exécuté à partir du répertoire racine, affiche tous les fichiers du système. Comme la commande travaillera durant un long moment, vous pourrez l'interrompre en appuyant sur les touches Ctrl-C.

Maintenant, revenez dans votre répertoire home et exécutez `find` à nouveau. Vous verrez tous les fichiers de votre répertoire personnel. Vous pouvez spécifier différentes options à `find` :

find -type d affiche seulement les répertoires mais pas leurs fichiers,

find -type f affiche seulement les fichiers et non les répertoires qui les contiennent même si la commande descendra dans chacun d'eux,

find -name <fichier> trouve exclusivement les fichiers qui ont pour nom `<fichier>`. Par exemple, testez `find -name '*.c'` qui détectera tous les fichiers ayant l'extension `.c` (utiliser l'expression `find -name *.c` sans les caractères ' ' ne fonctionnera pas). `find -name Mary_Jones.lettre` détectera le fichier portant le nom `Mary_Jones.lettre`,

find -taille [[+|-]] <taille> trouve les fichiers qui ont une taille supérieure à (pour +) ou inférieure à (pour -) `<taille>` kilooctets ou à la taille spécifiée s'il n'y a pas de signe,

find <répertoire> [<repétoire> ...] commence la recherche dans chaque répertoire indiqué,

Il existe trop d'options pour ce petit tour d'horizon. Aussi, est-il utile de se reporter à `find`(1) pour davantage de détails (c'est-à-dire que vous devriez exécuter la commande `man 1 find`). Regardez aussi l'option `-exec` qui force `find`

à exécuter une commande pour chaque fichier trouvé, par exemple :

```
find /usr -type f -exec ls '-al' '{}' "\;"
```

La commande `find` présente l'inconvénient de lire de manière active les répertoires pour y trouver les fichiers. Ce processus est lent, notamment lorsque vous commencez depuis le répertoire racine. `locate <fichier>` est une commande alternative. Celle-ci recherche les fichiers dans une base de données établie au préalable. Donc, la recherche est rapide. En contre-partie, il est nécessaire d'utiliser `updatedb` afin de mettre à jour la base de données des fichiers utilisés par `locate`. Sur certains systèmes, `upadatedb` est exécuté automatiquement tous les jours à 4H00.

Testez ces commandes (`updatedb` prendra quelques minutes) :

```
updatedb
locate rpm
locate deb
locate passwd
locate HOWTO
locate README
```

5.15 Recherche *dans* les fichiers.

Très souvent, vous devrez chercher dans un certain nombre de fichiers afin d'y trouver un mot ou une partie de phrase donnée. Cela pourrait être le cas de fichiers contenant une liste de numéros de téléphone avec des noms de personnes et des adresses. La commande `grep` effectue une recherche ligne-par-ligne dans un fichier et affiche les lignes qui contiennent un mot que vous lui avez indiqué. La syntaxe de `grep` se présente ainsi :

```
grep [options] <motif> <fichier> [<fichier> ...]
```

[Les mots "mot", "chaîne", "motif" (*word, string, pattern* en anglais) sont utilisés comme des synonymes dans ce contexte. Ils signifient une courte énumération de lettres et/ou de nombres pour lesquels vous essayez de trouver une correspondance. Un motif peut également désigner une chaîne de caractères avec des motifs de remplacement, comme nous le verrons plus tard].

Exécutez la commande `grep` avec pour motif le mot "le" de manière à retourner à l'écran les lignes contenant ce motif "le" : `grep 'le' Mary_Jones.lettre`. A présent, essayez `grep 'le' *.lettre`. Voyons quelques options :

grep -n <motif> <fichier> montre le nombre de lignes pour lequel le motif a été trouvé dans le fichier.

grep -<nombre> <motif> <fichier> affiche le `<nombre>` de lignes qui précèdent et suivent chaque ligne où le motif a été détecté dans le fichier.

grep -A <nombre> <motif> <fichier> affiche le `<nombre>` de lignes qui suivent chaque ligne où le motif a été détecté dans le fichier.

grep -B <nombre> <motif> <fichier> affiche le `<nombre>` de lignes qui précèdent chaque ligne où le motif a été détecté dans le fichier.

grep -v <**motif**> <**fichier**> imprime seulement les lignes qui ne contiennent pas le motif indiqué comme argument. [Vous pouvez vous dire que l'option `-v` ne fait plus le même type de recherche que `grep` est sensé faire, c'est-à-dire rechercher des chaînes de caractères. En fait, les commandes UNIX possèdent une telle versatilité dans leur fonctionnalité qu'elles recouvrent d'autres commandes. Avec UNIX, on n'arrête jamais d'apprendre de nouvelles et astucieuses méthodes pour réaliser des choses cachées dans les coins obscurs des pages de man].

grep -i <**motif**> <**fichier**> agit de la même manière que `grep` mais sans sensibilité à la casse.

5.16 Copie vers des disquettes formatées MS-DOS et Windows.

Un paquet logiciel nommé `mtools` permet de lire et d'écrire sur des disquettes MS-DOS/Windows. Les commandes de ce paquet ne sont pas des standards UNIX, mais elles sont présentes sur la majorité des distributions LINUX. Les commandes supportent les disquettes avec des "noms de fichiers longs" de type Windows. Placez une disquette dans le périphérique `A :` et essayez ceci :

```
mdir A :
touch mon_fichier
mcopy mon_fichier A :
mdir A :
```

Notez qu'il *n'*y a *pas* de systèmes comme le périphérique `A :` sur LINUX. Seul le paquet logiciel `mtools` interprète cette nomenclature de périphériques, ce qui aide les utilisateurs de Windows à se retrouver en terrain de connaissance lors de leur passage à LINUX. La liste complète des commandes est :

floppyd	mcopy	mformat	mmount	mshowfat
mattrib	mdel	minfo	mmove	mtoolstest
mbadblocks	mdeltree	mkmanifest	mpartition	mtype
mcat	mdir	mlabel	mrd	mzip
mcd	mdu	mmd	mren	xcopy

Exécutez `info tools` et vous obtiendrez plus de détails. En général, toute commande MS-DOS saisie en minuscule et précédée de `m`, donne la commande LINUX correspondante.

5.17 Archivages et sauvegardes.

Ne débuter jamais un travail avant que vous n'ayez une méthode sûre de sauvegarde de vos données.

Une des activités essentielles d'un administrateur système consiste à réaliser des sauvegardes (*backups*). Il est essentiel de ne jamais sous-estimer la volatilité des informations enregistrées sur un ordinateur. [La volatilité des données est leur capacité à disparaître ou à devenir ingérables]. Une sauvegarde est un duplicata de vos fichiers qui peut être utilisé en remplacement de données qui, en tout ou en partie, ont été détruites sur un ordinateur. L'idée est que toutes les données se trouvant dans un répertoire sont enregistrées dans un endroit différent, souvent sous forme compressée, de manière à être retrouvées en cas d'urgence. [De manière désormais usuelle, le terme "répertoire" désigne aussi bien un répertoire que les sous-répertoires, les fichiers des sous-répertoires et ainsi de suite...]. Quand nous voulons enregistrer divers fichiers, il est utile de les archiver en un seul fichier sur lequel des opérations peuvent éventuellement être réalisées. Lorsque de très nombreux fichiers sont ainsi regroupés, le fichier résultant est une *archive*. Usuellement, les archives ont l'extension `.tar`, qui est la constraction de *tape archive*.

Pour *créer* l'archive d'un répertoire, utilisez la commande `tar` :

```
tar -c -f <nom_de_fichier.tar> <répertoire>
```

Créez un répertoire avec de nouveaux fichiers, et exécutez la commande `tar` pour une sauvegarde. Un fichier `<fichier.tar>` sera créé. Prenez bien note de tous messages d'erreur que `tar` rapporterait. Listez le fichier et vérifier si sa taille est appropriée par rapport à la taille du répertoire intial.[4] Vous pouvez aussi utiliser l'option *verify* (voir la page de man) de la commande `tar` pour vérifier l'intégrité de `<fichier.tar>`. Maintenant, supprimez le répertoire, et restaurez-le avec l'option *extract* de la commande `tar` :

```
tar -x -f <nom_de_fichier.tar>
```

Votre répertoire devrait être recréé avec ses fichiers intacts. Il est intéressant d'appliquer l'option `-v`. Cette dernière affiche tous les fichiers qui ont été ajoutés ou extraits de l'archive lors de son traitement (archivage/désarchivage). Elle est très utile pour contrôler l'archivage.

Il va de soi que vous pouvez attribuer n'importe quel nom à votre archive. Toutefois, il est d'usage de la nommer `<repertoire.tar>` ce qui vous permet d'y voir plus clair grâce à l'extension. Une autre option intéressante est `-p` qui préserve les attributs de fichiers. Une fois votre archive obtenue vous voudrez probablement la compresser avec `gzip`, ce qui donnera le fichier `<repertoire.tar.gz>`, parfois nommé `<repertoire.tgz>` par souci de contraction.

Un nouveau type d'utilitaire pour l'archivage est `cpio`. `cpio` est plus performant que `tar`, mais est considéré plus difficile à manier. Le principe d'utilisation de `cpio` est similaire à `tar` et vous est laissé à titre d'exercice.

[4] Pour cela, exécutez (voir section 5.9) : `du -s <répertoire>`

5.18 Le PATH où les commandes sont recherchées.

Toute commande saisie derrière l'invite (*shell prompt*) se trouve dans un répertoire. Sous UNIX, les commandes (ou *programmes exécutables*) sont enregistrées parmi quatre répertoires . L'emplacement est déterminé par le type de la commande plutôt que par le paquet logiciel auquel elle appartient. Par exemple, dans le cas d'un traitement de texte, l'exécutable peut être stocké dans un répertoire avec d'autres exécutables totalement différents alors que ses fichiers de polices de caractères sont dans un répertoire avec les polices d'autres paquets logiciels.

Le shell a une technique de recherche des exécutables invoqués à la ligne de commande. Si vous tapez, par exemple, `/bin/cp`, le shell essaye d'exécuter le programme `cp` à partir du répertoire `/bin`. Si vous tapez seulement `cp`, le shell essaye de trouver la commande `cp` dans chaque sous-répertoire de votre `PATH`. Pour visualiser ce qu'est votre `PATH`, essayez :

```
echo $PATH
```

Vous verrez une ligne de plusieurs répertoires séparés par des caractères `:` (double-point).[5] Notez que le répertoire courant `.` n'est pas affiché. Il est fondamental que le répertoire courant ne soit pas dans le `PATH` pour des raisons de sécurité. Ceci fait que pour exécuter une commande dans le répertoire courant, il est nécessaire de faire : `./<commande>`.

Pour ajouter un nouveau répertoire à votre `PATH` (imaginons, par exemple, `/opt/gnome/bin`), faites :

```
PATH=" $PATH :/opt/gnome/bin"
export PATH
```

LINUX permet de réaliser l'opération en une seule étape :

```
export PATH=" $PATH :/opt/gnome/bin"
```

Il y a une commande supplémentaire, `which`, permettant de vérifier qu'une commande est localisable via le `PATH`. Parfois, il y a deux commandes de même nom dans différents répertoires contenus dans le `PATH`. [Ceci est plus fréquent sur Solaris que sur LINUX]. L'exécution de `which <commande>` à l'invite permet de localiser la commande que votre shell exécutera. Essayez :

```
which ls
which cp mv rm
which which
which cranzgots
```

`which` est aussi utile dans les scripts de shell pour révéler si une commande existe, et par conséquent, pour vérifier si un paquet logiciel est installé. Par exemple, `which netscape`.

[5] `/bin : /usr/bin : /usr/local/bin`

5.19 L'option --

S'il advient qu'un nom de fichier débute par -, alors il sera impossible de l'utiliser en tant qu'argument d'une commande. Pour contourner cette problème, la plupart des commandes présentent l'option - -. Cette option indique à la commande qu'il n'y a pas d'autre option qui suit. Tout autre indication sera traité comme un nom de fichier littéral. Par exemple :

```
touch -- -nom_de_fichier_stupide
rm -- -nom_de_fichier_stupide
```

Chapitre 6

Les expressions rationnelles.

Une expression rationnelle est une séquence de caractères formant un motif-type utilisé pour la recherche de chaînes de caractères dans un texte [mots, phrases, ou toute séquence de caractères]. Une expression rationnelle permet donc la recherche de motif. Pour avoir une idée de la manière dont une expression rationnelle fonctionne, imaginons le cas d'une liste de noms et de numéros de téléphone. Si vous voulez trouver un numéro de téléphone contenant un 3 en seconde position et se terminant par 8, une expression rationnelle constitue un moyen simple de faire cette recherche. Vous pourriez aussi prendre le cas d'un courriel à 50 personnes, où il faudrait remplacer le mot suivant "Chèr(e)" par le nom de la personne à contacter afin de personnaliser le message. Les expressions rationnelles autorisent ce type de recherche et de remplacement.

6.1 Vue d'ensemble.

De nombreux utilitaires emploient les expressions rationnelles pour obtenir une plus grande souplesse lors de la manipulation de textes. La commande `grep` est un exemple d'utilitaire. Au chapitre 5, nous avons utiliser `grep` pour localiser de simples séquences de lettres dans un texte. A présent, nous allons effectuer des recherches grâce aux expressions rationnelles.

Dans le chapitre précédent, nous avons vu que le caractère `?` pouvait être utilisé comme un motif de remplacement (ou *joker*) dans la mesure où il remplace n'importe quel caractère. Ce motif de remplacement fonctionne avec les noms de fichiers. Dans le cas des expressions rationnelles, le motif de remplacement est `.`. Aussi, pouvez-vous utiliser la commande `grep .3....8 <fichier>` pour trouver les numéros de téléphone à 7 chiffres, recherchés dans l'exemple décrit dasn l'introduction au présent chapitre.

Les expressions rationnelles sont employées pour effectuer une recherche ligne-par-ligne. Par exemple, si les 7 caractères sont répartis en deux lignes (c'est-à-dire qu'il y a un saut de ligne au milieu), `grep` ne pourra pas trouver le numéro correspondant à l'expression rationnelle qui lui est passée. En général, un programme qui utilise des expressions rationnelles opère ses recherches une ligne à la fois.

Ci-dessous, se trouvent quelques expressions rationnelles qui constituent une base essentielle. Nous utilisons les commandes `grep` et `egrep` pour illustrer l'uti-

lisation des expressions rationnelles (rappelez-vous que l'option `-w` vaut pour les mots, seulement). Les expressions sont insérées entre des guillemets simples ' ' pour des raisons qui seront expliquées plus tard.[1]

grep -w 't[a-i]e' détecte les mots `tee`, `the` et `tie`. Les crochets ont une signification spéciale. Ils stipulent que la recherche se fera sur un caractère allant de `a` à `i`.

grep -w 't[i-z]e' détecte les mots `tie` et `toe`.

grep -w 'cr[a-m]*t' détecte les mots `craft`, `credit` et `criquet`. Le signe * désigne n'importe quel nombre de caractères parmi ceux cités entre crochets. En l'occurrence, il s'agit de la gamme de caractères allant de `a` à `m`.

grep -w 'kr.*n' détecte les mots `kremlin` et `krypton`. Le signe . signifie "tout caractère", et le signe * signifie "un nombre quelconque de fois".

egrep -w '(th | sh).*rt' détecte les mots `shirt`, `short` et `thwart`. Le signe | signifie qu'il faut rechercher les correspondances sur `th` ou `sh`. `egrep` se comporte comme `grep` mais pour des *extended regular expressions* qui permettent donc l'usage de |. [Le caractère | est un opérateur logique OU (en anglais : OR) qui permet la sélection de ce qui se trouve à gauche ou à droite de |. Ceci est également vrai dans plusieurs langages de programmation]. Notez d'une part, que les crochets indiquent une recherche sur un caractère parmi plusieurs et d'autre part, que les parenthèses sont utilisées avec l'opérateur logique | qui permet la recherche sur un mot parmi plusieurs.

grep -w 'thr[aeiou]*t' extrait les lignes contenant les mots `thread` et `throat`. Une liste de caractères possibles peut se retrouver entre crochets.

grep -w 'thr[^a-f]*t' sélectionne les lignes contenant les mots `throughput` et `throat`. Le signe ^ après le crochet gauche indique que la recherche porte sur tous les caractères sauf ceux qui vont de `a` à `f`. Par exemple, le mot `thrift` ne sera pas détecté puisqu'il contient un `f`.

Toutes les expressions rationnelles décrites ci-dessus permettent de rechercher des mots (en raison de l'option `-w`). Si l'option `-w` n'est pas présente, elles trouveront des correspondances sur des parties de mot, ce qui contribuera à augmenter le nombre de correspondances. Notez également que le signe * signifie "tout nombre de caractère", y compris "pas de caractère". Par exemple, `t[a-i]*e` pourrait détecter la séquence `te`, soit un `t` et un `e` sans aucun caractère entre eux.

Le plus souvent, vous utiliserez des expressions rationnelles qui effectuent des recherches sur les lignes entières. Parfois, vous préférerez détecter une ligne qui commence ou se termine par une chaîne donnée. Le caractère ^ spécifie le début d'une ligne et `$` la fin. Par exemple, `^The` détecte les lignes débutant par `The` et `hack$` les lignes se terminant par `hack`. Par ailleurs, `'^*The.*hack$'` extrait les lignes de texte qui commencent par `The` et se finissent par `hack`, même si un caratère blanc est présent au début ou à la fin de la ligne. Parce que les expressions rationnelles utilisent des caractères spéciaux (`. \ [ ] * + ?`), ces derniers ne peuvent être utilisés directement. Cette restriction vous

[1] NdT : la recherche de mots anglais a été conservée comme dans le document original.

limite sévèrement pour la recherches de caratères tel que . , disons. Pour détecter correctement un caractère . , il convient d'utiliser la séquence \. forçant l'interprétation sur . qui, en conséquence ne sera plus considéré comme un caratère de remplacement. Donc, l'expression rationnelle monfichier.txt pourrait correspondre à monfichiertxt et à monfichier.txt. Cependant, l'expression rationnelle monfichier\.txt ne correspond qu'à monfichier.txt.

Vous pouvez utiliser la plupart des caractères spéciaux en les faisant précéder du signe *backslash* \ . Par exemple, vous utiliserez \[pour détecter le caractère réel [, ainsi que \$ pour le caractère $, l'expression \\ pour \, \+ pour + et l'expression \? pour le caractère réel ? (l'usage de ? et + est expliqué ci-dessous).

6.2 La commande fgrep.

fgrep est une alternative à grep. La différence est que grep (la commande la plus utilisée) fonctionne avec des expressions rationnelles alors que fgrep recherche des chaînes littérales. En d'autres termes, vous pouvez utiliser fgrep lorsque vous désirez rechercher des chaînes ordinaires qui ne forment pas des expressions rationnelles. Cela évite l'usage du caractère d'échappement \.

6.3 Notation \{\} des expressions rationnelles.

x* permet de rechercher le caractère x de 0 à une infinité d'occurrence. Vous pouvez spécifier d'autres gammes de nombres, par exemple avec x\{3,5\} qui recherche au moins 3 mais pas plus de 5 x, c'est-à-dire, xxx, xxxx et xxxxx.

Si vous recherchez xxxx exactement, vous pouvez employer x\{4\}. Notez que x\{7,\} recherchera 7 x au moins. Ici, la borne supérieure n'est pas mentionnée.

Comme dans tous exemples précités, le caractère x peut remplacer par une gamme de caractères (comme, par exemple, [a-k]) :

grep -w 'th[a-t]\{2,3\}t' détecte les mots theft, thirst, threat, thrift et throat.

grep -w 'th[a-t]\{4,5\}t' détecte les mots theorist, thicket et thinnest.

6.4 Expressions rationnelles étendues + ? \ < \ > ( ) | – notation avec egrep.

Une version améliorée des expressions rationnelles permet quelques facilités qui provoqueraient des conflits avec grep mais pas avec egrep :

\+ est analogue à \{1,\}. Il agit comme * mais détecte *un caractère ou plus* au lieu de *zéro caractère ou plus*.

? est analogue à "-1". Il détecte *zéro* ou *un* caractère.

\< \> peut entourer plusieurs chaînes séparées par l'opérateur | . (Seulement pour egrep).

\(\) peut entourer plusieurs chaînes séparées par \|. (Seulement pour grep).

Les exemples suivants devraient rendre ces deux dernières notations plus claires :

grep 'trot' détecte les mots `electrotherapist`, `betroth` et ainsi de suite mais,

grep '\<trot\>' détecte seulement `trot`.

egrep -w '(this|that|c[aeiou])*t' détecte les mots `this`, `that`, `cot`, `coat`, `cat` et `cut`.

6.5 Sous-expressions d'expressions rationnelles.

Les sous-expressions sont étudiées au chapitre 9.

Chapitre 7

Modifications de fichiers “texte”.

La création et la modification d'un fichier texte ordinaire est parfois traduite (improprement) sous le nom d'*édition de texte* (*text editing*). Un traitement de texte est un type particulier d'éditeur un peu plus simple que les éditeurs UNIX.

7.1 `vi`.

L'éditeur important qu'il faut apprendre à manipuler est `vi`. Vous découvrirez rapidement pourquoi et vous verrez des éditeurs un peu plus conviviaux.

Tapez simplement :

```
vi <fichier>
```

pour modifier un fichier. Vous pouvez utiliser un éditeur `vi` amélioré (`vi` improved = `vim`) :

```
vim <fichier>
```

Pour sortir de `vi`, pressez la touche  , la séquence `:q!` et finalement  .

`vi` possède un tutoriel qui vous permet de vous débrouiller en 20 minutes. Si, au bout d'un moment, vous êtes saturé, passez ce tutoriel. Vous poursuivrez votre apprentissage petit-à-petit. Pour lire le tutoriel, entrez :

```
vimtutor
```

ce qui édite le fichier :

- `/usr/doc/vim-common-5.7/tutor`
- `/usr/share/vim/vim56/tutor/tutor`
- `ou /usr/share/doc/vim-common-5.7/tutor/tutor`,

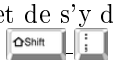
selon votre distribution. [Ceci vous permet d'apprécier la nature des différences qu'il y a entre diverses distributions de LINUX]. Vous verrez alors le texte suivant dans la partie supérieure de votre écran :

```
===========================================
=          Welcome to the VIM tutor - version 1.4          =
===========================================

Vim is a powerful editor that has many commands, too many to explain
in a tutor such as this. This tutor is designed to describe enough of the
commands that you will be able to use easily Vim as an all-purpose editor.
The approximate time required to complete the tutor is 25-30 minutes.
```

Vous êtes supposé modifier le fichier `tutor` comme exercice pour les 6 leçons qui viennent. Copiez-le d'abord dans votre répertoire personnel (home).

Le tableau 7.1 est un guide rapide pour `vi`. Il contient un extrait des quelques centaines de commandes disponibles, mais cela est suffisant pour réaliser des opérations de base. Prenez note des commandes suivantes :

- `vi` possède plusieurs *modes*. Si vous pressez la touche  , vous entrerez dans le mode *insertion*. Vous pourrez alors saisir du texte (comme vous le feriez dans un éditeur MS-DOS). Vous pourrez vous déplacer à l'aide du curseur et effacer du texte. Pour sortir de ce mode *insertion*, pressez la touche  . En conséquence, vous ne pourrez plus ajouter de texte, mais il sera possible d'en supprimer et de s'y déplacer.
- L'action consistant à presser  (c'est-à-dire la touche portant le signe :) vous place dans le mode *commande* où vous pouvez effectuer des opérations comme importer, sauver le fichier courant, rechercher des motifs, et traiter le texte. Typiquement, vous tapez `:` ensuite, vous entrez une commande et pressez  .
- Le terme *tampon* est utilisé ci-dessous. Il désigne un presse-papier caché.
- Une fonctionnalité utile est d'entrer `:set ruler` avant de faire quoique ce soit. Ceci permet d'afficher à l'écran le numéro de la colonne et de la ligne où votre curseur se trouve.

TAB. 7.1 – Commandes usuelles de `vi`.

Touches	Fonctions
Ouvrir - sauvegarder - quitter :	
:w	sauvegarder les modifications
:wq	sauvegarder et quitter
:q!	quitter sans sauvegarder
:new	ouvrir une nouvelle fenêtre vi ou vim
:w nom_de_fichier	sauvegarde avec pour nom de fichier "nom_de_fichier"
:5,20w nom_de_fichier	idem ; de la ligne 5 à la ligne 20

TABL. 7.1 – Suite (1).

:5,$ nom_de_fichier	idem mais en prenant de la ligne 5 à la fin du fichier
:e	éditer un autre fichier sans sortir de vi
:e !	passer à un autre fichier en perdant le précédant.
:split nom_de_fichier	ouvrir une autre fenêtre contenant nom_de_fichier
:qa	quitter toutes les fenêtres
:wqa	sauvegarder et fermer toutes les fenêtres
Ctrl-W j	déplace le curseur dans la fenêtre du dessous.
Ctrl-W k	déplace le curseur dans la fenêtre du dessus.
Crtl-W -	diminue la taille de la fenêtre
Crtl-W +	augmente la taille de la fenêtre
Mode insertion :	
i	insertion de nouveaux caractères avant le curseur
a	insertion de nouveaux caractères après le curseur
O	insertion à la ligne avant le curseur
o	insertion à la ligne après le curseur
:r nom_de_fichier	insère le fichier nom_de_fichier
Suppression - correction :	***Opérations effectuées en mode commande***
x	efface le caractère sous le curseur
X	effectue un backspace et une copie du mot supprimé vers le tampon.
dd	supprime la ligne où est le curseur et la copie dans le tampon
$n$dd	supprime n lignes vers le bas à partir curseur et les copie dans le tampon
P	ré-écrit la ligne fraîchement supprimée avant la ligne courante
p	ré-écrit la ligne fraîchement supprimée après la ligne courante
u	supprime la dernière modification
dw	supprime le mot sur lequel le curseur se trouve (ou le remet)
dg	supprime le texte entre la position du curseur et la fin du fichier
d$	supprime le texte entre la position du curseur et la fin de la ligne.
de	supprime, de la position du curseur à la fin du mot
db	supprime, de la position du curseur au début du mot

TABL. 7.1 – Suite (2).

Annuler - Refaire	
u	annuler (= undo)
Crtl-R	refaire
Remplacement de texte :	
R	met le curseur sur la première lettre du mot à remplacer
r	remplace le caractère sous le curseur
~	bascule de minuscule vers majuscule, depuis la lettre sous le curseur (et vice-versa). Faire alt ~ puis appuyer sur la touche "espace"
cw	supprime le mot sur le curseur (on peut écrire ensuite)
p	copie le texte du tampon à la position du curseur
Déplacement :	
h	curseur vers la gauche (identique à ←)
j	curseur vers le bas
k	curseur vers le haut
l	curseur vers la droite
w	curseur au mot suivant
b	curseur au début du mot courant
0 (zéro)	curseur au début de la ligne courante
$	curseur à la fin de la ligne courante
f	vers l'avant
b	vers l'arrière
G	en fin de fichier
1G	au début de fichier
10G	à la ligne 10 par rapport au début
Recherche et remplacement	
/mot (suivi d'une pression sur la touche enter)	recherche les occurrences de "mot" vers l'avant
:? (suivi d'une pression sur la touche "enter")	idem mais vers l'arrière
:1,%s/mot_à_remplacer/mot_de_remplacement	de la ligne un à la fin, remplace mot_à_remplacer par mot_de_remplacement %s peut être substitué par %10, si on veut effectuer l'opération de la ligne 1 à la ligne 10
:1,%s/mot_à_remplacer/mot_de_remplacement/c	idem mais en demandant une confirmation
:1,%s/mot_à_remplacer/mot_de_remplacement/g	idem mais sans confirmation

TABL. 7.1 – Suite (3)

Copier-coller	
v	passage en mode "mise en évidence" (*highlighting*)
y	copie le texte mis en évidence dans le tampon
d	supprime le texte mis en évidence et le copie dans le tampon
p	copie le contenu du tampon à l'endroit où le curseur est positionné
Presser v et déplacer le curseur de quelques mots ou de quelques lignes et ensuite :	Sélectionne une partie du texte et ...
:s/mot_à_remplacer/ mot_de_remplacement	... y remplace mot_à_remplacer par mot_de_remplacement
Assistance	
:help	accès à l'aide

7.2 Mise en évidence syntaxique.

Il est possible d'utiliser la *mise en évidence syntaxique* dans vos textes (*syntax highlighting*) ce qui signifie qu'un script de `bash` (terme explicité plus tard) ressemblera à ceci :

```
#!/bin/sh

for file in * ; do
    VAR="cat $file"
    echo $VAR | tr 'a-z' 'A-Z'
done
```

au lieu de

```
#!/bin/sh

for file in * ; do
    VAR="cat $file"
    echo $VAR | tr 'a-z' 'A-Z'
done
```

Cette mise en évidence syntaxique contribue à éviter des erreurs de programmation en colorant des motifs de manière déterminée. Vous pouvez enclencher la mise en évidence syntaxique dans `vi` en faisant `:syntax on` (mais cela ne fonctionnera pas avec `vi`). Utilisez la mise en évidence lorsque c'est possible : les bons éditeurs modernes le permettent.

7.3 Editeurs.

Bien qu'UNIX ait des capacités graphiques très performantes, la plupart des actes d'administration de bas-niveau s'effectuent à partir de fichiers en mode texte. Le traitement des textes est également mieux réalisé avec des outils tenant en compte la typographie qui se basent sur la création de texte ordinaire. [Ceci en dépit de tous les améliorations des traitements de texte WYSIWIG (what you see is what you get). Ainsi, ce document de traduction lui-même a été réalisé avec LyX mais aussi avec LaTeX et l'éditeur `vim`].

Historiquement, l'éditeur de texte standard était `ed`. `ed` permet à l'utilisateur de ne voir qu'une ligne de texte à la fois (ce qui est primitif du point de vue des

standards actuels). Aujourd'hui, `ed` est utilisé dans sa version dynamique, `sed`. Depuis longtemps, `ed` a été remplacé par `vi`. Un éditeur est l'outil avec lequel vous passerez le plus clair de votre temps, que vous fassiez du traitement de texte, de la création de pages web, de la programmation ou de l'administration. C'est votre première application interactive.

7.3.1 Cooledit.

(Lisez ceci si vous voulez un éditeur ouvrant un fichier et vous permettant de faire des saisies comme sur Windows).

Le meilleur éditeur pour le travail quotidien est `cooledit` [Pour citer Paul Sheer : "en tant qu'auteur de `cooledit`, je suis problalement partial"], disponible sur la *page web de Cooledit* `http ://cooledit.sourceforge.net`. Cooledit est un éditeur graphique (fonctionnant donc sous **X**). Il constitue aussi un Environnement de Développement Intégré (ou *Integrated Development Environnement* – IDE) pour tout ce que vous désirez faire. Ceux d'entre vous qui veulent acheter un IDE ne doivent pas aller plus loin car Cooledit est gratuit.

Les personnes venant du monde Windows trouveront qu'il est aisé et très puissant. Il ne requiert pas d'apprentissage. Tapez juste Cooledit et commencez votre saisie. Son alter ego en mode texte s'appelle `mcedit`; ilest associé au paquet logiciel GNU Midnight Commander, `mc`. La version en mode texte est moins performante que celle des autres éditeurs comme `emacs` et `jed` mais elle est adéquate si vous ne désirez pas investir beaucoup de temps en mode texte.

Cooledit a des menus déroulant et des touches intuitives. Il n'est pas nécessaire de lire de la documentation avant d'en tirer parti.

7.3.2 `vi` et `vim`.

Aujourd'hui, `vi` est considéré comme le standard. C'est le seul éditeur qui est installé par défaut sur tous les systèmes UNIX. `vim` est une version améliorée et augmentée en termes de fonctionnalité par rapport à `vi`. Il est fondamental d'acquérir les bases de `vi` même si ce dernier n'est pas votre éditeur utilisé au quotidien. Le fait est qu'un jour ou l'autre, tout administrateur est confronté à son usage.

En revanche, les nouveaux utilisateurs trouveront cet éditeur peu intuitif et laborieux parce qu'il requiert un certain temps d'apprentissage et la mémorisation de nombreuses commandes.

A sa décharge, il faut signaler que de nombreuses personnes n'utilisent que `vi` et qu'il s'agit probablement (avec `emacs`) de l'éditeur qui fait *tout*. C'est aussi un des rares éditeurs qui présentent une version parfaitement opérationnelle et cohérente sur tous les systèmes UNIX et non-UNIX. `vim` fonctionne sur AmigaOS, AtariMiNT, BeOS, DOS, MacOS, OS/2, RiscOS, VMS, et les Windows (95/98/NT4/NT5/2000) ainsi que sur toutes les variantes d'UNIX.

7.3.3 Emacs.

Emacs signifie Editor MACroS. C'est le monstre parmi les éditeurs et il peut absolument faire tout ce qu'on peut imaginer réaliser avec un logiciel. Il est devenu *de facto* un standard à l'instar de `vi`.

Emacs est bien plus qu'un simple éditeur de texte. C'est un système complet d'utilisation d'un ordinateur pour le développement, les communications, la gestion de fichiers, et de multiples autres fonctions que vous n'imaginez peut-être pas encore. Il en existe même une version pour le système X Window, qui permet de naviguer sur l'Internet.

7.3.4 Autres éditeurs.

Parmi les autres éditeurs, vous trouverez `joe`, `jed`, `gedit`, `nedit`, `pico`, `nano` et davantage encore qui émulent l'aspect et le comportement des environnements DOS, Windows et Apple Mac, ou qui apportent de meilleures interfaces en utilisant Gtk/Gnome ou Qt/KDE. La liste s'allonge régulièrement. En bref, ne pensez pas que l'éditeur ou les éditeurs que votre distribution a gravé(s) sur vos CDs est (ou sont) ce qui convient le mieux. Il en va d'ailleurs de même pour les applications.

Chapitre 8

Les scripts du shell.

Ce chapitre introduit les notions de *programmation d'un ordinateur* (*computer programming*). Jusque-là, vous avez entré les commandes une à une. La programmation consiste simplement à organiser l'exécution d'un ensemble de commandes, pour obtenir une nouvelle et puissante fonction.

8.1 Introduction.

Pour exécuter plusieurs commandes séquentiellement, créez d'abord un fichier avec une extension `.sh`, ce fichier contiendra vos commandes. L'extension `.sh` n'est pas absolument nécessaire mais elle sert de rappel indiquant que le fichier contient un texte spécial nommé *script de shell* (ou *shell script*). Désormais, le mot *script* sera employé pour décrire toute séquence de commandes dans un fichier texte. A présent, faites :

```
chmod 0755 monficher.sh
```

ce qui permet au fichier d'être exécuté (les droits sur les fichiers sont décrits au chapitre 12).

Modifiez le fichier en utilisant votre éditeur de texte préféré. La première ligne devrait se présenter comme ceci (sans qu'il n'y ait d'espace blanc [Les espaces blancs sont des tabulations, des espaces à proprement parler ou de nouvelles lignes]) :

```
#!/bin/sh
```

Cette ligne impose au programme qui la suit d'être un script de *shell*. Ce qui veut dire que sont acceptées des commandes du même type que celles entrées en ligne de commandes derrière votre invite (ou *prompt*). A présent, entrez les commandes suivantes pour vous faire la main :

```
echo “Bonjour tout le monde”
echo “Quel est votre nom? (saisissez votre nom ici et puis pressez la
touche enter)”
read NM
echo “Bonjour $NM”
```

Maintenant, vous pouvez sortir de votre éditeur et taper `./monfichier.sh`. Ceci exécutera votre script [ce qui provoque la lecture puis l'action des lignes de commandes en séquence. C'est ce qu'on appelle *exécuter* un programme]. Notez que taper `./monfichier.sh` n'est pas différent que de saisir une commande à l'invite du shell. Votre fichier `monfichier.sh` est en fait devenu une nouvelle commande LINUX.

Observez bien ce que fait la commande `read`. Elle crée une “boîte” nommée `NM` et y place le texte saisi au clavier. Ensuite, chaque fois que le shell rencontre `NM`, son contenu est écrit à la place de `NM`, pour autant que cette variable soit précédé du signe $. Nous disons que `NM` est une *variable* parce que son contenu peut être modifié.

Vous pouvez utiliser les scripts de shell pour faire du calcul. Essayez ceci :

```
echo “Je vais calculer le produit de X par Y”
echo “Entrez X”
read X
echo “Entrez Y”
read Y
echo “X*Y = $X*$Y = $[X*Y]”
```

Les crochets `[` et `]` signifient que ce qu'ils renferment doit être évalué [substitué, modifié, réduit à une forme plus simple] comme une expression numérique [une séquence de nombres avec les signes +, *, -, / entre ces nombres]. D'ailleurs, vous pouvez à tout moment faire un calcul dans le shell en saisissant derrière l'invite :

```
echo $[3*6+2*8+9]
```

Vérifiez que votre shell permet l'usage de cette notation `[ ]`. Certains UNIX utilisent la commande `expr` pour réaliser ces opérations.

8.2 Boucles : instructions `while` et `until`.

Le shell lit chaque ligne séquentiellement de haut en bas : c'est le flux de programmation. Si, maintenant, vous désirez qu'une commande soit exécutée plus d'une fois, il faut altérer le flux de programmation. La commande `while` exécute une séquence contrôlée de commandes de manière répétée. En voici un exemple (`-le` veut dire *less than or equal* : plus petit ou égal) :

```
N=1
while test "$N" -le "10"
do
     echo "Number $N"
     N=$[N+1]
done
```

L'initialisation `N=1` crée une variable `N` et lui attribue une valeur `1`. La commande `while` exécute toutes les commandes entre `do` et `done`, répétitivement juqu'à ce que la condition de `test` ne soit plus vérifiée (en l'occurrence, jusqu'à ce que `N` ne soit plus inférieur ou égal à `10`). L'expression `-le` signifie *less than or equal to*. Voir `test` (1) (c'est-à-dire `man (1) test`) pour rencontrer d'autres types de test que vous pouvez appliquer à des variables. Observez aussi la manière dont `N` est substitué avec la nouvelle valeur qui augnmente d'une unité à chaque répétition de la boucle `while`.

Vous devriez constater que chaque ligne est une commande ; les commandes sont séparées par un *passage à la ligne*. Vous pouvez aussi avoir plus d'une commande par ligne pour autant qu'elles soient séparées par un point-virgule comme dans l'exemple suivant :

```
N=1 ; while test "$N" -le "10"; do echo "Number $N"; N=$[N+1] ; done
```

(Essayez de décompter à partir de 10 avec (*greater than or equal to*)). Il est aisé de constater que les scripts de shell sont puissants parce que tout type de commande peut être exécuté avec des conditions et des boucles.

La déclaration `until` est identique à `while` sauf que sa logique est inverse. La même fonctionnalité est rencontrée avec `-gt` (*greater than*) :

```
N=1 ; until test "$N" -gt "10 ; do echo "Number $N"; N=$[N+1] ; done
```

8.3 Boucles : l'instruction `for`.

La commande `for` permet également l'exécution de commandes à de multiples reprises. Elle fonctionne de la manière suivante :

```
for i in vache mouton coq cochon
do
     echo "$i est un animal de ferme"
done
echo -e "mais\nLes GNUs ne sont pas des animaux de ferme"
```

La commande `for` prend chaque chaîne de caractères après `in` et exécute les lignes entre `do` et `done` avec `i` remplacé par chaque chaîne. Les chaînes peuvent décrire quoique ce soit (y compris des nombres) bien que, souvent, il s'agisse de nom de fichiers.

La commande `if` exécute certaines commandes pourvu qu'une condition soit remplie (`-gt` vaut pour *greater than* ; `-lt` pour *lesser than*). La commande exé-

cute toutes les lignes entre `if` et `fi` (c'est-à-dire "if" épelé à l'envers).

```
X=10
Y=5
if test "$X" -gt "$Y"; then
      echo "$X est plus grand que $Y"
fi
```

La commande `if` peut être utilisée pleinement comme dans :

```
X=10
Y=5
if test "$X" -gt "$Y"; then
      echo "$X est plus grand que $Y"
elif test "$X" -lt "$Y"; then
      echo "$X est plus petit que $Y"
else
      echo "$X est égal à $Y"
fi
```

A présent, créons un script qui interprète ses arguments.

Appelons-le `backup-lots.sh` :

```
#!/bin/sh

for i 0 1 2 3 4 5 6 7 8 9; do
      cp $1 $1.BAK-$i
done
```

Ne l'exécutez pas immédiatement après en avoir modifié les droits (`chmod 0755 backup-lots.sh`). Créez d'abord un fichier vide `data` avec la commande `touch` (c'est-à-dire `touch data`). Maintenant, l'exécution de `./ backup-lots.sh data` revient à copier le fichier `data` 10 fois avec 10 extensions différentes. Vous constatez que la variable `$1` a une signification particulière : elle désigne le premier argument mentionné derrière la commande. Réalisons un script plus astucieux où nous allons tester l'existence d'un argument et également, l'existence de fichiers de sauvegarde (utilisation de la forme `test -e`) :

```
#!/bin/sh

if test "$1" = "" ; then
      echo "Usage : backup-lots.sh <filename>"
      exit
fi
for i in 0 1 2 3 4 5 6 7 8 9; do
      NEW_FILE=$1.BAK-$i
      if test -e $NEW_FILE; then
              echo "backup-losts.sh **warning** $NEW_FILE
              echo "                already exists - skipping"
      else
              cp $1 $NEW_FILE
```

```
      if
done
```

8.4 Arrêt de boucles et relance.

Une boucle qui requiert une terminaison prématurée peut comprendre une instruction `break` :

```
#!/bin/sh

for i 0 1 2 3 4 5 6 7 8 9; do
      NEW_FILE=$1.BAK-$i
      if test -e $NEW_FILE; then
         echo "backup-lots.sh : **error** $NEW_FILE
         echo "                already exists - exiting"
         break
      else
         cp $1 $NEW_FILE
      fi
done
```

Ceci permet de sortir du programme en allant sur la ligne après le `done`. Si deux boucles sont imbriquées, la commande `break 2` provoquera l'arrêt de l'exécution sur les `2` boucles ; et ainsi de suite pour les valeurs supérieures à `2`.

L'instruction `continue` est aussi utile pour clôturer l'itération d'une boucle. Si l'instruction `continue` est rencontrée, l'exécution est poursuivie à partir du début de la boucle suivante, donc en ignorant le corps des boucles restant à effectuer :

```
#!/bin/sh
for i 0 1 2 3 4 5 6 7 8 9; do
      NEW_FILE=$1.BAK-$i
      if test -e $NEW_FILE; then
              echo "backup-lots.sh : **warning** $NEW_FILE
              echo "                already exists - skipping"
              continue
      fi
              cp $1 $NEW_FILE
done
```

Notez bien que `break` et `continue` fonctionnent dans des boucles `for`, `while` et `until`.

8.5 Boucles sur des motifs d'englobement.

Nous savons que le shell peut interpréter des noms de fichiers *via* les motifs d'englobement. Ainsi, nous savons que `ls -l *.txt` affichee tous les fichiers se terminant par `txt`. Allons plus loin :

```
#!/bin/sh

for i in *.txt ; do
      echo "found a file :" $i
done
```

Le motif `*.txt` est appliqué à toute la recherche mais seulement dans le *répertoire courant*. Si vous incluez un chemin absolu, la recherche se fera dans le répertoire désigné :

```
#!/bin/sh

for i in /usr/doc/*/*.txt ; do
      echo "found a file :" $i
done
```

Cet exemple démontre la capacité du shell à rechercher des fichiers ainsi qu'à utiliser les motifs d'englobement et des chemins absolus.

8.6 L'instruction `case`.

L'instruction `case` peut réduire un programme potentiellement complexe. Voici un exemple d'application :

```
#!/bin/sh

case $1 in
        --test|-t)
                echo "you used the --test option"
                exit 0
       ;;
        --help|-h)
                echo "Usage :"
                echo "      myprog.sh [--test|--help|--version]"
                exit 0
       ;;
        --version|-v)
                echo "myprog.sh version 0.0.1"
                exit 0
       ;;
        -*)
                echo "No such option $1"
                echo "Usage :"
                echo "      myprog.sh [--test | --help|--version]"
                exit 1
       ;;
esac

echo "You type \"$1\" "on the command-line"
```

Vous noterez que nous essayons de traiter le premier argument d'un programme. Etant donné qu'il y a plusieurs options, l'usage de `if` conduirait à un programme

plutôt long. L'instruction `case` permet de spécifier plusieurs blocs d'instructions en fonction de la valeur d'une variable. Notez le signe `;;` qui sépare chaque bloc d'instructions. Les chaînes avant le motif `)` sont des correspondances de motifs d'englobement. La première correspondance réussie entraîne l'exécution de son bloc. Le symbole `|` permet d'entrer plusieurs motifs d'englobement.

8.7 Usage des fonctions : le mot-clé `function`.

Jusqu'ici, nos programmes s'exécutaient principalement du début à la fin. Souvent, certaines parties du code sont répétées. C'est une mauvaise pratique de programmation que de répéter explicitement des instructions présentant la même fonctionnalité. Les définitions de fonction fournissent une méthode pour grouper des instructions en un seul bloc. Une fonction regroupe donc une liste de commandes sous le même nom. Elle est exécutable à chaque appel. Prenons l'exemple qui suit :

```
#!/bin/sh

function usage ()
{
        echo "Usage :"
        echo "       myprog.sh [--test|--help|--version]"
}

case $1 in
        --test|-t)
                echo "you used the --test option"
                exit 0
        ;;
        --help|-h)
                usage
        ;;
        --version|-v)
                echo "myprog.sh version 0.0.2"
                exit 0
        ;;
        -*)
                echo "Error : no such option $1"
                usage
                exit 1
        ;;
esac

echo "You typed \" $1\ "on the commande-line"
```

Lorsqu'`usage` apparaît, le terme est substitué par les lignes se trouvant entre `{` et `}`. Les avantages de cette approche sont évidents : si vous souhaitez changer la description de la fonction "usage", il ne vous faudra faire les modifications qu'à un seul endroit du script. Les bons programmes utilisent si abondamment cette technique qu'ils exploitent rarement plus de 50 lignes de code sans faire un appel à l'une ou l'autre fonction.

8.8 Utiliser proprement les arguments de la ligne de commandes : le mot-clé `shift`.

La plupart des programmes que nous avons vu peuvent prendre de nombreux arguments en ligne de commandes, parfois sans que l'ordre n'ait d'importance. Nous allons voir comment nos propres scripts peuvent utiliser cette fonctionnalité. Les arguments en ligne de commandes sont accessibles via `$1`, `$2`, etc. Le script :

```
#!/bin/bash

echo "The first argument is : $1, second argument is : $2, third
argument is : $3"
```

peut être exécuté avec :

```
myfile.sh chien chat hibou
```

et il affiche :

```
The first argument is : chien, second argument is : chat, third
argument is : hibou
```

Nous aurions bien besoin d'une boucle sur chaque argument. Un script comme :

```
for i in $1 $2 $3 $4 ; do
      <instructions>
done
```

n'apporte pas beaucoup de souplesse. Le mot-clé `shift` est destiné à faciliter les choses. Il modifie la position des arguments en déplaçant `$1` à la valeur de `$2`, tandis que `$2` prend la valeur de `$3` et ainsi de suite (`!=` teste que "`$1`" n'est pas égal à "", c'est-à-dire que `$1` n'est pas vide). Essayez :

```
while test "$1" != "" ; do
      echo $1
      shift
done
```

et exécutez le programme avec de nombreux arguments.

A présent, vous pouvez introduire toute sorte de conditions entre `do` et `done` pour traiter les arguments tour-à-tour :

```
#!/bin/sh

function usage ()
{
        echo "Usage :"
        echo "      myprog.sh [--test|--help|--version] [--echo
<text>]"
```

```
}

while test "$1" != ""; do
        case $1 in
                --echo|-e)
                        echo "$2"
                        shift
                ;;
                --test|-t)
                        echo "you used the - -test option"
                ;;
                --help|-h)
                        usage
                        exit 0
                ;;
                --version|-v)
                        echo "myprog.sh version 0.0.3"
                        exit 0
                ;;
                -*)
                        echo "Error : no such option $1"
                        usage
                        exit 1
                ;;
        esac
        shift
done
```

myprog.sh peut, dès lors, être utilisé avec de multiples arguments.

8.9 Les arguments en ligne de commande : $@ et $0.

Alors que $1, $2, $3, ... s'appliquent aux arguments individuels passés aux programmes, $@ s'applique à tous les arguments. Ce comportement est utile pour passer tous les arguments restant à une seconde commande. Par exemple :

```
if test "$1" = "--special"; do
      shift
      myprog2.sh "$@"
fi
```

$0 désigne le nom du programme lui-même et non "tout argument" de la ligne de commande. En fait, $0 est la commande utilisée pour invoquer le programme. Dans les cas ci-dessus, il s'agit de ./myprog.sh. Notez que $0 est insensible à l'opération shift.

8.10 Notation avec guillemets 'simples droits'.

Les guillements simples ' ' protègent le texte qu'ils entourent des actions du shell. En d'autres termes, vous pouvez placer n'importe quel caractère spécial entre des guillements simples (de type accent aigü) et le shell les traitera de manière littérale sans les interpréter. Par exemple, si vous voulez que le signe $ soit considéré comme un littéral dans l'expression `costs $1000`, vous utiliserez `echo 'costs $1000'` au lieu d'`echo " costs $1000"`.

8.11 Notation avec guillements "doubles".

Les guillements doubles "" ont exactement la signification inverse de celle des guillemets simples . Ils autorisent le shell à interpréter les caractères qu'ils renferment. Ils sont utilisés car ils permettent de grouper les mots et les caractères séparés par un espace blanc en un seul mot apparent (autrement le shell considérerait chaque partie séparée par un espace blanc comme un terme). Essayez :

```
for i in "henry john mary sue"; do
      echo "$i   is    a    person"
done
```

et comparez avec :

```
for i in henry john mary sue; do
      echo "$i   is    a    person"
done
```

8.12 Substitution avec guillemets `inversés `.

Les guillemets simples mais inversés ` ` ont une signification particulière pour le shell. Quand une commande est à l'intérieur des guillements inversés, cela signifie que la commande sera exécutée et sa *sortie* substituée par ce que les guillemets inversés contiennent. Considérez la commande `cat`. En premier lieu, créez un petit fichier intitulé `to_be_catted` contenant le seul mot `daisy`. Ensuite, créez le script suivant :

```
X=`cat to_be_catted`
echo $X
```

L'action de `cat` sur le fichier `to_be_catted` retourne `daisy` dans les guillemets de sorte que `X` contient ce mot. C'est un outil puissant. Considerez la commande `expr` :

```
X=`expr 100 + 50 '*' 3`
echo $X
```

Donc, nous pouvons utiliser `expr` et les guillemets inversés pour exécuter des

opérations mathématiques sur des entiers dans les scripts de shell.[1] Ci-dessous se trouve une fonction permettant de réaliser le factoriel d'un nombre. Notez la manière dont le signe `*` est inséré dans les guillemets simples. Ceux-ci empêchent le shell de considérer ce signe comme un motif de remplacement (comme cela est utilisé dans la recherche de fichiers) :

```
function factorial ()
{
   N=$1
   A=1
   while test $N -gt 0 ; do
      A=‘expr $A ’*’ $N‘
      N=‘expr $N -1‘
   done
   echo $A
}
```

Vous observerez que les crochets utilisés précédemment peuvent être employés presque tout le temps avec `expr`. (Cependant, la notation `$[ ]` est une extension des shells de GNU et ne constitue pas une norme pour toutes les variantes d'UNIX). Nous pouvons maintenant exécuter le code et observer le résultat. Si nous voulons attribuer le résultat à une variable, nous utiliserons `X=‘factorial 20‘`.

Prenez note du fait qu'une autre notation donne le même effet que les guillemets inversés : `$(`*command*`)` qui est identique à ‘*command*‘. Dans ce livre, nous utiliserons la notation classique “en guillemets”.

[1] NdT : pour les calculs impliquant des nombres décimaux, il convient d'utiliser `bc`.

Chapitre 9

Flux et sed comme éditeur de flux.

Une des forces de Linux, c'est sa capacité à utiliser les tubes (*pipes* en anglais). Le non-usage de ces derniers constitue d'ailleurs un des handicaps des systèmes non-Unix. Les tubes utilisés en ligne de commande –comme cela est expliqué dans ce chapitre– constituent un procédé assez direct et simple. Cependant, les tubes employés dans les programmes en **C** simplifient énormément la programmation. Sans eux, de grandes quantités de code complexe et difficile à déboguer devraient être écrits pour réaliser des tâches simples. L'auteur espère que ce chapitre donnera au lecteur une idée de la raison qui fait d'Unix un système de référence si fréquent et si stable.

9.1 Introduction.

Les commandes `grep`, `echo`, `df`, etc. affichent leur résultat à l'écran. En fait, ce qui se produit à un niveau plus bas, c'est qu'elles écrivent les caractères qui seront affichés, dans un *flux de données* (ou *tube*) appelé tube *stdout*. Le shell lit ces données, caractère par caractère, et les affiche à l'écran. Le mot *tube* signifie exactement ceci : un programme met des données au début d'un canal tandis qu'un autre programme lit ces données à l'autre bout. Les tubes permettent donc à deux programmes séparés de communiquer entre eux. Dans les cas cités ci-avant, les programmes communiquent simplement avec le shell qui affiche leur sortie.

9.2 Tutoriel.

Créez un fichier texte avec plusieurs lignes contenant le mot `GNU` et une ligne contenant à la fois `GNU` et `Linux`. Exécutez la commande : `grep GNU mon_fichier.txt`. Comme d'habitude, le résultat est envoyé à stdout. A présent, essayez : `grep GNU mon_fichier.txt > gnu_lignes.txt`. La sortie de la commande `grep` a été l'objet d'une *redirection* vers un fichier. La partie `> gnu_lignes.txt` indique au shell de créer un nouveau fichier `gnu_lignes.txt`

et y copie toute sortie de stdout au lieu d'effectuer un affichage à l'écran. Si un fichier existe déjà, il sera *tronqué* [son contenu sera écrasé].

A présent, supposons que vous désirez ajouter le résultat à la suite du contenu du fichier. L'utilisation de `>>` au lieu de `>` ne tronque pas le fichier et, en outre, permet de copier le résultat de `grep` à la suite du contenu de ce fichier. Essayez :

```
echo "davantage de données" >> gnu_lignes.txt
```

et visualisez `gnu_lignes.txt`.

9.3 Tubes et opérateur |

La vraie force des tubes se manifeste lorsqu'un programme peut lire la sortie d'un autre programme. Revenons à la commande `grep` qui lit à partir de stdin lorsqu'on ne lui confère pas d'arguments. Exécutez `grep` avec un argument en ligne de commandes (le shell met le curseur à la ligne) ; ensuite, saisissez quelques lignes comme ceci :

```
[root@cericon] grep GNU
Une ligne sans ce mot
Une autre ligne sans ce mot
Une ligne avec GNU
Une ligne avec GNU
J'ai compris
```

Pour revenir au shell, pressez Ctrl-C. Le comportement par défaut de `grep` consiste à lire depuis stdin si aucun argument n'est donné. Comme vous pouvez le constater, le fonctionnement normal de la commande est d'afficher le mot `GNU` lorsqu'il se trouve dans une ligne. Aussi, les lignes qui contiennent `GNU` sont-elles affichées une seconde fois (puisque vous les introduisez, que la commande lit le texte et vérifie la présence de `GNU`).

Maintenant, essayez `grep GNU mon_fichier.txt | grep Linux`. Le premier `grep` sort toutes les lignes ayant le mot `GNU` vers stdout. Le signe `|` indique que toute la sortie de stdout doit être écrite en tant que stdin (ainsi que nous l'avons fait ci-dessus) vers la commande suivante, qui est également un `grep`. Cette seconde commande `grep` crible les données pour les lignes contenant le mot `Linux`. La commande `grep` est souvent utilisée comme filtre [un système qui trie les données] et peut être employée à de nombreuses reprises.

```
grep L mon_fichier.txt | grep i | grep n | grep u | grep x
```

Le caractère `<` redirige le contenu d'un fichier à la place de stdin. Dit autrement, le contenu d'un fichier remplace ce qui normalement provient du clavier. Essayez :

```
grep GNU < gnu_lignes.txt
```

9.4 Exemple d'un tube complexe.

Dans le chapitre 6, nous avons appliqué `grep` à un dictionnaire anglais pour aborder les expressions rationnelles. Voyons comment faire un dictionnaire de mots (votre dictionnaire de base peut se trouver dans `/var/share/` ou `/usr/lib/spell`) :

```
cat /usr/lib/aspell/fr-60-only.rms | strings | tr 'A-Z' 'a-z' \
| grep '^[a-z]' | sort -u > mon_dico
```

[Un *backslash* comme dernier caractère d'une ligne indique que la ligne se poursuit. Vous pouvez omettre le \ mais vous devrez aussi omettre la nouvelle ligne. Ceci est connu sous le nom de *continuité de ligne* (ou *line continuation*)].

Le fichier `fr-60-only.rms` contient un dictionnaire UNIX usuellement utilisé pour les corrections orthographiques. En triant un peu, vous créerez un dictionnaire qui permet d'aisément résoudre des mots croisés. Tout d'abord, nous employons la commande `string`, vue à la section 5.9, pour extraire les octets lisibles du texte. En l'occurrence, nous avons tiré parti de son mode alternatif : elle lit à partir de stdin lorsqu'aucun fichier ne lui est passé en argument. La commande `tr` (abréviation pour *translate* (traduire)) convertit les majuscules en minuscules. La commande `grep` filtre alors les lignes ne débutant pas par une lettre. Finalement, la commande `sort` trie les mots par ordre alphabétique. L'option `-u` signifie `u`*nique*, et précise que les lignes multiples doivent être éliminées. Essayez `less mon_dico`.

9.5 Redirections avec `>&`.

A présent, testez la commande `ls nofile.txt > A`. Nous nous attendons à ce que `ls` retourne un message d'erreur si `nofile.txt` n'existe pas. Le message d'erreur est en effet affiché mais il n'est pas écrit dans le fichier `A`. La raison en est que `ls` a écrit son message dans stderr tandis que `>` a seulement redirigé stdout. Pour faire en sorte qu'à la fois la sortie vers stdout et celle vers stderr soient versées dans le même fichier, il faut utiliser un *opérateur de redirection*. Pour le shell, sdtout est désigné par `1` et stderr par `2`, de sorte qu'une commande doit être suivie de la redirection `2>&1` pour forcer stderr à accompagner la sortie de stdout. Les mots stderr et stdout sont, en fait, utilisés dans le langage **C** où les nombres *1* et *2* sont connus comme des *numéros de fichiers* ou des *descripteurs de fichiers*. Essayez ceci :

```
touch fichier_existant
rm -f fichier_non_existant
ls fichier_existant fichier_non_existant
```

`ls` produira deux lignes : l'une contenant le terme `fichier_existant` et l'autre, un message d'erreur pour indiquer que le fichier `fichier_non_existant` n'existe pas. Le message d'erreur a été passé à stderr (ou descripteur de fichier `2`) et la ligne d'après a été passée à stdout (descripteur de fichier `1`).

A présent, exécutez les commandes suivantes :

```
ls fichier_existant fichier_non_existant 2>A
cat A
```

D'une part, la ligne associée à `fichier_existant` est retournée à l'écran et d'autre part, `A` contient le message d'erreur. Essayons la commande suivante :

```
ls fichier_existant fichier_non_existant 1>A
cat A
```

La notation `1>` est la même que `>A` car, lorsque le descripteur de fichier n'est pas indiqué, le shell suppose qu'il s'agit d'`1`. Notez qu'`A` contient stdout et que le message d'erreur est retourné à l'écran.

Testons :

```
ls fichier_existant fichier_non_existant 1>A 2>&1
cat A
```

A présent, le fichier `A` contient le message d'erreur et la sortie normale. `>&` est appelé un *opérateur de redirection*. L'expression x>&y indique au shell d'écrire le tube x dans le tube y. La redirection est indiquée de la droite vers la gauche sur la ligne de commande. Donc, dans l'exemple ci-dessus, `2>&1` est opéré avant `1>A`. Donc, stderr est mélangé à stdout et le résultat est redirigé vers `A`.

Finalement :

```
ls fichier_existant fichier_non_existant 2>A 1>&2
cat A
```

Nous observons que ceci a le même effet que précédemment, excepté que la séquence est inverse : stdout est d'abord redirigé vers stderr (`1>&2`) puis stdout est redirigé vers `A`.

Pour apprécier ce qui se passe en permutant les redirections entre elles, faisons :

```
ls fichier_existant fichier_non_existant 2>&1 1>A
cat A
```

ce qui signifie que stdout est redirigé vers `A` puis, que stder est redirigé vers stdout. Cette commande ne mixe pas stderr et stdout puisque la redirection vers `A` se produit en premier lieu.

9.6 Utilisation de `sed` pour éditer les flux.

`ed` a été utilisé comme l'éditeur standard (standard *ed*itor) d'UNIX. Sa maîtrise est difficile mais l'éditeur est compact et programmable. `sed` est la contraction de *stream editor* et il constitue la seule résurgence d'`ed` qui soit encore utilisée de nos jours. `sed` permet d'éditer des fichiers de manière non-interactive. A l'instar de `grep` qui peut chercher des mots pour filtrer des lignes de texte, `sed` peut opérer des recherches et remplacements, insérer et supprimer des lignes dans des fichiers de textes. `sed` est un de ces programmes qui n'ont pas de pages

de man. Faites `info sed` pour accéder à des pages `info` explicitant `sed` à l'aide d'exemples.

L'usage le plus courant de `sed` est le remplacement de mots dans un flux avec d'autres mots ou expressions littérales. `sed` lit depuis sdtin et écrit vers stdout. Son comportement est similaire à `grep` ; il lit une ligne à la fois et ensuite écrit la ligne modifiée par les opérations d'édition qui lui ont été imposées. Les remplacements sont, en général, réalisés de cette manière :

```
cat <fichier> | sed -e 's/<recherche-expreg>/<remplace-texte>/<option>' > \
<fichier_résultat>
```

Ici, `<recherche-expreg>` désigne une expression rationnelle, `<remplace-texte>` est le texte qui substitue le motif détecté, `<option>` peut ne pas être mentionné ou valoir `g`. Dans ce dernier cas, cela signifie que le remplacement aura lieu pour chaque occurrence du motif détecté (usuellement, `sed` ne remplace que la première occurrence de l'expression rationnelle de chaque ligne). Il existe d'autres `<options>` ; référez-vous à la page `info sed`. Testons maintenant ceci :

```
sed -e 's/e/E/g'
```

Appuyez sur la touche "entrée" et après le passage à la ligne, introduisez quelques lignes de texte.

9.7 Sous-expressions d'expressions rationnelles.

Cette section explique comment faire la tâche apparemment complexe de permuter des parties de textes à l'intérieur d'un texte. Prenons, par exemple, la sortie de `ls` : supposons que vous vouliez isoler automatiquement la colonne "taille". `sed` peut effectuer ce type d'opération si vous utilisez la notation spéciale `\ ( \)` pour regrouper les parties d'une expression rationnelle. Avant de s'attaquer à ce problème, considérons deux exemples d'utilisation de sed avec des sous-expressions rationnelles qui nous viendrons à point nommé pour traiter le cas de `ls`. Premier exemple : la structure des sous-expressions rationnelles. Analysons :

```
sed -e 's/\(<[^ ]*\>\)\([ ]*\)\(<[^  ]*\>\)/\3\2\1/g'
```

Notons que cette commande contient, pour `<recherche-expreg>`, le motif `\(<[^ ]*\>\)\([ ]*\)\(<[^ ]*\>\)` qui est consituté de trois sous-expressions rationnelles : la `1` est `\(<[^ ]*\>\)` ; la `2` est `\([ ]*\)` et la `3` est `\(<[^ ]*\>\)`. Le motif `<remplace-texte>` est constitué du numéro des trois sous-expressions rationnelles `\3\2\1`. Voyons la signification de cette commande.

La première sous expression `\(<[^ ]*\>\)` signifie `\<. *\>`, c'est-à-dire n'importe quel caractère (.) un nombre de fois indéfini (*) (il s'agit d'un mot au sens large). La seconde sous-expression `\([ ]*\)` signifie `\[ ]*`, c'est à dire un nombre arbitraire d'espaces (il s'agit d'une suite de blancs en nombre quelconque). La troisième sous-expression est identique à la première. Nous demandons ainsi à `sed` de rechercher un mot suivi de blancs, eux-mêmes suivis d'un autre mot.

Deuxième exemple : testons ce qui se produit quand vous exécutez ceci :[1]

```
sed -e 's/\(<[^ ]*\>\)\([ ]*\)\(<[^ ]*\>\)/\3\2\1/g'
GNU Linux est génial
Linux GNU génial est
```

A présent, revenons à notre problème de `ls` (notez que ceci n'est vraiment qu'un exemple, car pour déterminer la taille des fichiers vous devriez utiliser la commande `du`). Voyons comment nous pourrions obtenir la taille en octets de tous les fichiers de notre répertoire :[2]

```
expr 0 `ls -l | grep '^-' | \
    sed 's/^\([^ ]*[ ]*\)\{4,4\}\([0-9]*\).*$/ + \2/'`
```

Nous savons que, dans le cas des fichiers ordinaires, la sortie de `ls -l` commence par `-` . Nous utilisons `grep` pour éliminer les lignes qui ne commencent pas par - (ce que représente l'expression `ls -l | grep '^-'`). Si nous faisions cela, nous verrions que la sortie (qui maintenant ne concerne plus que les fichiers) est partagée en quatre colonnes d'information qui ne nous intéressent pas, avant que nous ne trouvions la colonne avec les tailles de fichiers.

Pour effectuer le remplacement ultérieurement, il faut donc rechercher une chaîne de caractères suivie d'un nombre quelconque d'espaces à l'aide de la première sous-expression rationnelle `\([^ ]*[ ]*\)` et ce, quatre fois exactement : d'où le complément `\{4,4\}`. Après quoi, nous trouverons une série de chiffres allant de 0 à 9 ayant une occurrence quelconque d'où la seconde sous-expression rationnelle `\([0-9]*\)`. Ensuite viennent une série de caractères quelconques (.) en nombre indéterminé (`*`) à la fin (`$`) de la recherche. Tous ces éléments compris entre `sed 's/` et `+` constituent donc le motif `<recherche-expreg>` que nous avons défini dans nos premiers exemples. Notez que nous avons fait l'impasse sur la notation `\< \>` pour les mots complets (voir la section 6.4). A la différence de grep, `sed` essaye de trouver les correspondances sur le nombre maximum de caractères autorisés, ce qui revient au même que de préciser que la recherche doit se faire sur des noms complets.

Vous pouvez essayer d'imaginer ce que retourne la commande `ls -l | grep '^-' | sed 's/^\([^ ]*[ ]*\)\{4,4\}\([0-9]*\).*$/ + \2/'` (ceci est un exemple) :

```
+ 438
+ 1525
+ 76
+ 92146
```

Donc, vous avez remplacé chaque ligne par la sous-expression `\2` et vous l'avez faite précédée d'un signe `+`. Les guillemets inversés vous retournent la sortie vers `expr` qui effectue la somme en ignorant tous les caractères de nouvelle ligne alors même que chaque nouvelle somme est l'objet d'une nouvelle ligne. Le

[1] On s'attend à ce que les trois premiers motifs, c'est-à-dire "GNU", " " ,"Linux" soient traités de sorte que la permutation retourne Linux GNU. Puis, `sed` opère sur le reste du texte en prenant les trois motifs suivants : "est", " ", "génial", qu'il permute.

[2] NdT : faites bien attention à entourer d'espaces l'opérateur +.

seul problème (mineur) est que la première ligne débute par un signe + qu'aucun nombre ne précède. Ceci devrait amener `expr` à émettre un message d'erreur. Pour contourner ce problème, il suffit d'introduire `0` suite à la commande `expr` de manière à commencer le calcul ainsi : `0 + ...` .

9.8 Insertion et effacement de lignes.

`sed` peut réaliser des opérations qui améliorent la rédaction de scripts qui modifient des fichiers de configuration. Par exemple :

```
sed -e '7a\
une ligne supplémentaire.\
Encore une autre.\
Une de plus.'
```

ajoute trois lignes *à la suite* de la ligne 7, tandis que :

```
sed -e '7i\
une ligne supplémentaire.\
Encore une autre.\
Une de plus.'
```

insère trois lignes *avant* la ligne 7. Par ailleurs,

```
sed -e '3,5D'
```

supprime les lignes de 3 à 5.

Dans la terminologie de `sed`, les nombres mentionnés ici sont des adresses qui peuvent être détectées à l'aide d'expressions rationnelles. Démontrons cela :

```
sed -e '/Cher Henry/, /Meilleures salutations, Jeanne/D'
```

Cette dernière instruction supprime toutes les lignes comprises entre une ligne contenant l'expression rationnelle `Cher Henry` et celle contenant, par ailleurs, `Meilleures salutations, Jeanne` (ou la fin du fichier si la dernière expression n'est pas mentionnée).

Ce comportement s'applique également aux insertions :

```
sed -e '/Love Jane/i\
Love Carol\
Love Beth'
```

Notez que le symbole `$` indique la dernière ligne :

```
sed -e '$i\
The new second last line\
The new last line.'
```

et finalement, que le symbole de négation, `!`, est utilisé pour détecter toutes les lignes qui ne sont pas spécifiées. Par exemple :

```
sed -e '7,11 !D'
```

efface toutes les lignes *mais pas* le texte compris entre les 7ème et 11ème lignes.

Chapitre 10

Processus et variables d'environnement.

A partir de ce chapitre, nous allons décrire ce qui se trouve sous le capot de votre système Linux.

10.1 Introduction.

Sur Unix, quand vous exécutez un programme (tel que toutes les commandes du shell que vous avez déjà testées), les vraies instructions de l'ordinateur sont lues depuis un fichier situé sur le disque, dans le répertoire `/bin` et, placées en RAM. Le programme est alors exécuté dans la mémoire et devient un *processus*. Un processus est une commande, un programme ou un script de shell qui a été lancé (ou *exécuté*) en mémoire. Quand le processus est terminé, il est éliminé de la mémoire. Il y a environ 50 processus fonctionnant simultanément à tout instant sur un système où un utilisateur s'est connectée. La CPU "passe" d'un processus à l'autre pour partager son temps d'exécution [c'est-à-dire le temps consacré pour réaliser les instructions d'un programme particulier. Notez que ceci est différent de ce qui se passe sur Windows ou DOS, où un programme –de lui-même– doit permettre aux autres un partage de la CPU : sous Unix, les processus n'ont rien à dire à ce propos]. Chaque processus se voit attribuer un nombre appelé le *PID* (process ID ou *process IDentifier*). A côté de la mémoire réellement occupée par l'exécutable, le processus en soi consomme de la mémoire pour ses opérations.

De la même manière qu'un fichier a pour propriétaire un utilisateur ou un groupe, un processus est affecté à un propriétaire –usuellement la personne qui exécute le programme. Chaque fois qu'un processus essaye d'accéder à un fichier, ses droits de propriété sont comparés à ceux du fichier de manière à décider si l'accès est autorisé. Du fait que tous les périphériques sont des fichiers, un processus ne peut faire *quelque chose* que via un fichier et donc, les restrictions sur les droits des fichiers sont le seul type de restrictions requises sur Unix [il y a des exceptions, toutefois]. C'est comme cela que la gestion des accès par Unix et la sécurité fonctionnent.

Le centre des opérations de gestion est le noyau (*kernel*). Le noyau est l'entité qui permet l'accès au matériel, l'exécution, l'allocation d'ID de processus, le

partage du temps de la CPU et la gestion des droits.

10.2 ps – liste de processus en cours.

Connectez-vous à un terminal et tapez la commande `ps`. Vous deviez obtenir une sortie analogue à :

```
PID  TTY  STAT  TIME  COMMAND
5995   2  S     0 :00   /bin/login -- mon_nom
5999   2  S     0 :00   -bash
6030   2  R     0 :00   ps
```

`ps` sans options montre trois processus en cours. Il n'y a que trois processus visibles pour vous, en tant qu'utilisateur, bien qu'il y ait d'autres processus en cours ne vous appartenant pas. Le premier est le programme par lequel vous vous êtes connecté en introduisant votre identifiant et votre mot de passe. Cela a entraîné l'exécution d'un autre processus appelé `bash`, le "Bourne Again shell" dans lequel vous avez saisi la commande `ps` [le shell de Bourne est le shell original de UNIX]. Finalement, vous avez exécuté `ps` qui s'est trouvé lui-même parmi les processus en cours. Ce processus disparaît immédiatement après l'affichage.[1]

10.3 Contrôle des tâches.

Le shell présente de nombreuses fonctionnalités pour le contrôle et l'exécution de processus –appelé contrôle des tâches (ou *jobs*). Créez un petit script intitulé `proc.sh` :

```
#!/bin/sh
echo "proc.sh : en cours"
sleep 1000
```

Exécutez ce script avec `chmod 0755 proc.sh` et `./proc.sh`. Le shell se fige en attendant que le processus s'accomplisse. Maintenant, pressez ^Z. Ceci entraînera l'arrêt du processus (c'est-à-dire une mise en pause sans la disparition en mémoire). Exécutez `ps` à nouveau. Vous constaterez que votre script sera listé. Cependant, il n'est plus en cours d'exécution puisqu'il a été arrêté. Tapez `bg` (pour *background*) ; le script sera maintenant réactivé mais en arrière-plan. Vous pouvez exécuter d'autres processus pendant ce temps. Saisissez `fg` et votre script revient en avant-plan. Pressez ^C pour interrompre le processus associé à votre script.

10.4 Création de processus en arrière-plan.

Créons, à présent, un programme qui fait quelque chose d'un peu plus intéressant :

[1] NdT : Considérez aussi la commande `pstree` (voir `pstree(1)`).

```
#!/bin/sh

echo "proc.sh : en cours"
while true ; do
        echo -e '\a'
        sleep 2
done
```

A présent, faisons ^Z, `bg`, `fg` et ^C comme précédemment. Si vous souhaitez mettre un processus en arrière-plan dès son exécution, vous utiliserez :

```
./proc.sh &
```

La section "**CONTROLE DES JOBS**" (**JOB CONTROL**) de la page de man de bash (`bash`(1)) ressemble à ceci :[2]

CONTROLE DES TACHES[3]

Le contrôle des tâches est la capacité à arrêter de manière sélective (*suspend*) l'exécution des processus et à en continuer l'exécution (resume) ultérieurement. Un utilisateur emploie typiquement cette fonctionnalité via une interface interactive fournie par le pilote de terminal du système et le **bash**.

L'interpréteur (shell) associe un *job* à chaque tube [Qu'est-ce que cela signifie ? Simplement que chaque fois qu'une commande, un programme ou un script est exécuté en arrière-plan, un nombre unique, appelé numéro de tâche, lui est attribué]. L'interpréteur maintient une table des tâches en cours d'exécution, cette table pouvant être affichée avec la commande `jobs`. Quand **bash** lance une tâche de manière asynchrone (c'est-à-dire en *arrière-plan*), il imprime une ligne du genre :

[1] 25647

indiquant que la tâche est associée à un numéro valant 1 et que le PID du dernier processus dans le tube est 25647. Tous les processus dans un tube simplefont partie du même processus. Le **bash** utilise la notion de tâche (ou *job)* comme une abstraction pour aider à les contrôler.

Pour faciliter l'implantation d'une interface utilisateur pour le contrôle des tâches, le système introduit la notion d'*ID de groupe de processus associés au terminal courant*. Les membres de ce groupe de processus (ceux dont le GID est égal au process group ID du terminal courant) reçoivent des signaux produits par le clavier tels que **SIGINT**. Ces processus sont dits en *avant-plan*. Les processus en *arrière-plan* sont ceux dont le GID de processus diffère de celui du terminal ; de tels processus sont protégés des signaux produits par le clavier. Seuls les processus en avant-plan peuvent lire à partir du terminal et y écrire. Les processus en arrière-plan qui tentent de lire à partir du terminal ou d'y écrire reçoivent un signal **SIGTTIN** (ou **SIGTTOU**) par le pilote de périphérique du terminal qui, sauf interception du signal (saisie), suspend le processus.

Si le système d'exploitation sur lequel **bash** est en cours, permet le contrôle des tâches, **bash** vous permettra l'accès. La saisie du caractère *suspend* (soit ^Z ou Ctrl-Z) pendant l'exécution du processus, provoque l'arrêt du processus et vous ramène dans le **bash**. La frappe du caractère de *suspension différée* (typiquement ^Y ou Crtl-Y) arrêtera le processus lorsqu'il tentera de lire depuis le terminal. Le contrôle reviendra à **bash**. Vous pourrez alors manipuler l'état de la tâche en utilisant la commande **bg** de manière à la relancer en arrière-plan, la commande **fg** pour mettre la tâche en avant-plan ou la commande **kill** pour tuer le processus. Un ^Z prend effet immédiatement et a pour effet corollaire de provoquer une sortie immédiate.

Il y a plusieurs méthodes pour désigner une tâche dans un shell. Le caractère % introduit un nom de tâche. Le numéro **n** de tâche peut être mentionné comme **%n**. Une tâche peut

[2] Merci à Brian Fox et à Chet Raney pour ces données (de la part de P. Sheer).

[3] NdT : concernant la traduction française, le texte est très largement inspiré de la page de man bash dont la traduction a été réalisée par Christophe Blaess en 1997 (cette traduction a été réactualisée par Thierry Vigneaud en 1999-2001).

aussi être décrite en employant le préfixe du nom utilisé pour le démarrer ou en employant une sous-chaîne qui apparaît en ligne de commande. Par exemple, **%ce** se réfère à une tâche **ce** arrêtée. Si un préfixe correspond à plusieurs tâches, **bash** rapporte une erreur. Utiliser **%?ce**, en revanche, permet de désigner toute tâche contenant la chaîne **ce** dans la ligne de commande. Si la sous-chaîne correspond à plusieurs tâches, **bash** renvoie une erreur. Les symboles **%%** et **%+** se rapportent au "job" de shell courant, qui est la dernière tâche arrêtée alors qu'elle était en avant-plan. La tâche précédente peut être mentionné en utilisant **%-**. Dans les affichages relatifs aux tâches (c'est-à-dire la sortie de la commande **jobs**), la tâche en cours est toujours indiqué avec un +, et celle qui la précède avec un **-**.

Il suffit de nommer une tâche pour la ramener en avant-plan : **%1** est un synonyme pour "**fg %1**", ramenant la tâche 1 de l'arrière-plan vers l'avant-plan. De manière analogue, "**%1 &**" transfère la tâche 1 en arrière-plan, ce qui équivaut à "**bg %1**".

Le shell est immédiatement informé de tout changement d'état d'une tâche. Normalement, le **bash** attend jusqu'à ce qu'il soit prêt à afficher une invite avant de faire un rapport des modifications de status de tâche afin qu'il n'interrompe pas toute autre affichage. Si l'option **-b** de la commande **set** est activée, le **bash** affiche les modifications immédiatement. (Voir aussi la description de la variable **notify** dans les **variables du shell**, ci-dessus).

Si vous essayez de quitter le **bash** alors que des tâches sont suspendues, le shell affiche un message d'avertissement. Vous pouvez alors utiliser la commande **jobs** pour vérifier l'état des tâches. Si vous pratiquez de cette manière ou que vous essayez de sortir de suite, vous ne serez plus averti et les tâches suspendues seront définitivement terminées.

10.5 Tuer un processus avec `kill`. Envoyer des signaux.

Pour terminer un processus, utilisez la commande `kill` :[4]

```
kill <PID>
```

Cette commande `kill` envoie en réalité un *signal* de terminaison au processus `<PID>`. L'envoi d'un signal veut simplement dire que le processus doit exécuter une des 30 fonctions prédéfinies.[5] Dans certains cas, les développeurs ne se sont pas préoccupés de définir une fonction (appelée *catching*) pour un numéro de signal particulier. Dans ce cas, le noyau utilise le comportement par défaut pour ce signal. Le comportement par défaut est usuellement d'ignorer le signal, d'arrêter le processus ou de le terminer. Le comportement par défaut pour la terminaison d'un signal est de faire disparaître le processus.

Pour envoyer un signal spécial à un processus, vous pouvez citer le nom du signal en argument à la ligne de commande ou utiliser son équivalent numérique :

```
kill -SIGTERM 12345
```

ou bien :

```
kill -15 12345
```

qui est le signal que `kill` envoie normalement quand aucun signal particulier n'est déclaré en option.[6]

[4] NdT : pour déterminer le PID, il faut utiliser la commande `top` (voir la section 10.8).

[5] Onze d'entre elles –les plus communes– sont décrites dans la section 10.6.

[6] NdT : Pour suspendre un processus, il est possible d'utiliser `kill -19 <PID>`, et pour le relancer `kill -18 <PID>`.

Pour terminer un processus de manière inconditionnelle :

```
kill -SIGKILL 12345
```

ou

```
kill -9 12345
```

qui ne devrait être utilisé qu'en dernier recours. *Il est interdit aux processus de traiter le signal* `SIGKILL`.

Etant donné qu'il est ennuyeux de devoir rechercher régulièrement le PID d'un processus, les utilitaires GNU présentent une commande, `killall`, qui envoie un signal à tous les processus de même nom :

```
killall -<signal> <nom_de_processus>
```

Cette commande est utile quand vous êtes sûr qu'il n'y a qu'un seul processus en cours, soit parce que personne d'autre n'est connecté, soit que vous n'agissez pas comme super-utilisateur. *Notez que sur d'autres systèmes* UNIX*, la commande killall tue* tous *les processus que vous êtes autorisés à tuer. Si vous agissez comme super-utilisateur, cette action plantera la machine.*

10.6 Liste des signaux courants.

La liste complète des signaux peut être obtenue via `signal`(7) ; consultez aussi le fichier `/usr/include/asm/signal.h`.

SIGHUP (1) *Suspendre (Hang up).* Si le terminal est déconnecté par un processus, ce signal est envoyé automatiquement au processus. Le fait d'envoyer ce signal à un processus amène souvent ce dernier à relire ses fichiers de configuration. Ceci est plus utile que de redémarrer le programme. Vérifiez toujours la page de man pour déterminer si un processus adopte ce comportement.

SIGINT (2) *Interrompre (Interrupt)* à partir du clavier. Ce signal est émis si vous pressez ^C.

SIGQUIT (3) *Quitter (Quit)* à partir du clavier. Ce signal est émis si vous pressez ^D.

SIGFPE (8) *Exception de virgule flottante (floating point exception).* Ce signal est transmis automatiquement à un programme réalisant certaines opérations mathématiques non-permises.

SIGKILL (9) *Tue (kill)* un processus. Ceci est un des signaux qui ne peuvent jamais être *traité* par un processus. Si un processus reçoit ce signal, il doit impérativement quitter et ne réaliser aucune opération de nettoyage (telle que fermer les fichiers ou enlever des fichiers temporaires). Vous pouvez envoyer un signal `SIGKILL` à un processus s'il n'y a pas d'autre moyen de le terminer.

SIGUSR1 (10), SIGUSR2 (12) *Signal d'utilisateur (User signal).* Ces signaux sont utilisés par les développeurs lorsqu'ils souhaitent des fonction-

nalités supplémentaires. Par exemple, certains processus commencent à consigner des messages de débogage quand vous envoyez `SIGUSR1`.

SIGSEGV (11) *Violation de segmentation (segmentation violation).* Ce signal est émis automatiquement quand un processus essaie d'accéder à de la mémoire en dehors de l'espace des adresses allouées ; ce signal est équivalent à Fatal Exception ou General Protection Fault sous Windows. Notez que les programmes avec des bogues ou des programmes en cours de développement rapportent souvent ce type de commentaire. Un programme recevant `SIGSEGV` ne peut jamais compromettre le reste du système. Si le noyau recevait ce type de signal, cela provoquerait l'arrêt du système ; néanmoins, cela est extrêmement rare.

SIGPIPE (13) *Tube inactif (pipe died).* Un programme a essayé d'écrire dans un tube, mais la sortie de ce dernier n'était plus disponible.

SIGTERM (15) *Terminaison (terminate)* : conduit un programme à terminer proprement.

SIGCHLD (17) *Terminaison des processus-fils (child terminate).* Ce signal est envoyé à un processus-parent chaque fois qu'un des processus qu'il a engendré s'éteint.

10.7 `Nice` et `renice` : programmer les priorités.

Tous les processus se voient allouer une durée d'exécution par le noyau. Si tous les processus avait la même durée allouée, les performances se dégraderaient à mesure que le nombre de processus augmenterait. Le noyau utilise un jeu de règles heuristiques pour estimer la durée à allouer à un processus. Le noyau essaie d'être équitable –deux utilisateurs requérant le même usage de la CPU devrait obtenir la même priorité, donc la même durée.

La plupart des processus sont en attente d'une frappe de touche, d'un envoi de données via réseau ou via un périphérique, etc. Ces processus ne consomment donc pas de CPU.

En revanche, quand plusieurs processus travaillent normalement, le noyau doit décider s'il faut donner une priorité plus grande à tel ou tel autre processus. Que se passe-t-il si un processus est en train de faire certaines opérations plus consommatrices de CPU que celles d'un autre processus ? Comment réagit le noyau ? La réponse tient dans une caractéristique d'UNIX appelée *programme de priorité* (ou *niceness*). La gamme de programmation des priorités s'étend de -20 à +20. Vous pouvez fixer la priorité d'un processus par la commande `renice` :

```
renice <priorite> <PID>
renice <priorite> -u <utilisateur>
renice <priorite> -g <groupe>
```

Un exemple typique concerne le programme *SETI*. [SETI signifie Search for Extraterrestrial Intelligence. SETI est une initiative basée sur des divers fonds d'origines variées pour balayer le ciel à la recherche de signaux radios en provenance d'autres civilisations. Les données que SETI recueille doivent être traitées de manière intensive. SETI distribue une partie de ses données à qui veut exécuter le programme `seti` en arrière-plan de manière à tirer parti de l'inactivité

de millions de machine. Il y a même un économiseur d'écran devenu populaire. Malheureusement, le collègue qui partage mon bureau exécute seti à la priorité -19 au lieu de +19, de sorte que rien ne fonctionne correctement sur sa machine...]. Mettez la priorité à +19 à l'aide de la commande :

```
renice +19 <PID>
```

pour éviter autant que possible de faire planter votre machine.

Notez que les valeurs de nice présente un ordre inverse à ce qui est souvent attendu : +19 signifie qu'un processus aura peu de CPU, tandis que -19 donne à un processus beaucoup de CPU. Seul le super-utilisateur peut fixer des valeurs négatives au processus.[7]

Essentiellement, les applications multimédia ainsi que quelques autres utilitaires sont les seuls processus qui nécessitent un adoucissement négatif et la plupart de ceux-ci ont leurs propres options en ligne de commandes pour fixer des valeurs de douceur adéquates. Voyez, par exemple, cdrecord(1) et mikmod(1) –une valeur négative vous prémunira contre les sauts de lecture. [Linux disposera bientôt d'un programme de priorité de processus en temps réel. C'est une caractéristique du noyau pour réduire la latence (les temps morts entre les moments durant lesquels un processus est exécuté par la CPU et aussi la durée requise pour qu'un processus se réveille). Il y déjà des correctifs pour réaliser cet objectif].[8]

Par ailleurs, les options -u et -g sont utiles, elles fixent la priorité de tous les processus d'un utilisateur ou d'un groupe.

De plus, nous disposons de la commande nice qui démarre un programme sous une douceur donnée par rapport à la valeur de douceur courante employée par un utilisateur. Par exemple :

```
nice +<priorite> <PID>
nice -<priorite> <PID>
```

Finalement, la commande snice peut à la fois afficher et modifier la douceur courante d'un processus. Cette commande purrait ne pas fonctionner sur certaines machines :

```
snice -v <PID>
```

10.8 Consommation mémoire et CPU : top.

La commande top classe tous les processus par leur consommation de la CPU et de la mémoire. Elle affiche le "top 20" sous forme d'une table. Lancez top chaque fois que vous voulez voir le (ou les) processus qui monopolise(nt) votre machine. La commande top -q -d 2 est utile[9] pour programmer la commande top elle-même afin de lui conférer une haute priorité. Ceci

[7] NdT : cette échelle est assez logique, si on admet que le terme *niceness* désigne la douceur d'un processus. Un processus doux possède bien évidemment une douceur plus grande (jusqu'à +20) qu'un processus agressif en terme de consommation de la CPU.

[8] NdT : au moment où le noyau 2.6 sortait, cette fonctionnalité n'était pas encore incorporée.

[9] NdT : la version top utilisée par le traducteur sur Gentoo-Linux n'admet pas l'option -q

permet le rafraîchissement de l'affichage à l'écran sans décalage (important). `top -n 1 -b > top.txt` affiche tous les processus et `top -n 1 -b -p <PID>` affiche de l'information à propos d'un processus en désignant ce dernier par son PID.

`top`présente des réponses interactives très utiles lorsque certaines touches sont pressées :

f affiche une liste des champs que vous pouvez modifier interactivement. Par défaut, les seuls champs montrés sont `USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND`, ce qui constitue l'essentiel (la signification des champs est donnée ci-après),

r redéfinit la priorité (renice) d'un processus,

k tue un processus.

La page de man de `top` décrit la signification des champs. Certains de ceux-ci sont un peu déroutant et supposent une connaissance des programmes en C qui les constitutent. La principale question qui intéresse un utilisateur se résume souvent ainsi : *combien de mémoire un processus utilise-t-il ?* La réponse est donnée par la champ `RSS`, qui signifie *Resident Set Size*. `RSS` désigne la quantité de RAM qu'un processus consomme à lui seul. Les exemples suivants montrent les totaux pour tous les processus en cours sur le système (qui possède 65536 kilo-octets de RAM, en l'occurrence). Ils représentent le total des champs `SIZE`, `RSS` et `SHARE`, respectivement.

```
echo ‘echo ’0’ ; top -q -n 1 -b | sed -e ’1,/PID *USER *PRI/D’ | \
   awk ’{print “+” $5}’ | sed -e ’s/M/\\*1024/’ ‘ | bc
68016

echo ‘echo ’0’ ; top -q -n 1 -b | sed -e ’1,/PID *USER *PRI/D’ | \
   awk ’{print “+” $6}’ | sed -e ’s/M/\\*1024/’ ‘ | bc
58098

echo ‘echo ’0’ ; top -q -n 1 -b | sed -e ’1,/PID *USER *PRI/D’ | \
   awk ’{print “+” $7}’ | sed -e ’s/M/\\*1024/’ ‘ | bc
30184
```

Le champ `SIZE` représente l'usage de la mémoire que fait un processus, au total. `RSS` fait la même chose, mais ne tient pas compte de la mémoire échangée sur la partition d'échange "swap". `SHARE` est la quantité de mémoire partagée entre les processus.

Les autres champs sont décrits dans la page de man comme suit :

uptime (ou top) Cette première ligne affiche l'heure, la durée depuis laquelle le système fonctionne, le nombre d'utilisateurs connectés, et les trois charges moyennes du système. Ces trois moyennes indiquent le nombre moyen de processus ayant fonctionné durant les 1, 5 et 15 dernières minutes. Cette ligne est exactement celle de la sortie de w(1) ou uptime(1). La ligne d'uptime peut être mise à jour par la commande 1 dans top ;

processes (ou tasks) La ligne 2 représente le nombre de processus en cours depuis le dernier rafraîchissement. Elle est constituée de plusieurs champs : nombre de tâches en cours, en suspens, arrêtées ou zombies (ce dernier terme désigne un processus qui a été tué mais qui n'a pas abandonné la table des processus parce que le processus-parent n'a pas encore exécuté un **wait**(2) ; les processus zombies sont parfois affichés par la commande ps(1)). Les processus et les états montrés peuvent être rafraîchis par la commande interactive **t** ;

CPU states indique, en pourcentage, le temps CPU utilisé en mode utilisateur (user), mode système (system), redéfinition des priorités (nice) et, arrêts (idle). Les processus dont le degré de douceur a été redéfini sont les seuls à avoir des valeurs négatives ;

Mem donne les statistiques sur l'utilisation de la mémoire : la mémoire totale disponible, la mémoire libre, la mémoire utilisée, la mémoire partagée et la mémoire utilisée par les tampons. L'affichage de ces informations peut être rafraîchi par la commande intéractive **m** ;

Swap donne les statistiques sur l'espace de swap : l'espace total, celui disponible et celui utilisé. Swap et Mem constituent le résultat de la commande free(1) ;

PID désigne l'IDentificateur du processus associé à chaque tâche ;

PPID est l'IDentificateur du processus-parent de chaque tâche ;

UID est l'ID d'utilisateur (le propriétaire de la tâche) ;

USER désigne le nom d'utilisateur du propriétaire de la tâche ;

PRI définit la priorité de la tâche ;

NI représente la valeur de la douceur d'une tâche ;

SIZE est la taille du code d'une tâche, ajoutée à l'espace de pile, (unités : kilo-octet) ;

TSIZE représente la taille du code d'une tâche. Ceci donne des valeurs spéciales pour les processus du noyau ; non-fonctionnel pour les processus ELF ;[10]

DSIZE est la taille des données et de la pile. Ne fonctionne pas pour les processus ELF ;

TRS indique la taille de texte restant ;

SWAP représente la taille de la partie (d'une tâche) déplacée sur l'espace de swap ;

D indique la taille de pages marquées "dirty" ;

LIB est la taille de l'ensemble des pages de bibliothèques utilisées. Ne fonctionne pas pour les processus ELF ;

RSS désigne la quantité totale de mémoire physique utilisée par une tâche (exprimée en kilo-octets). Dans le cas des processus ELF, les pages de bibliothèques utilisées sont prises en compte ici (ce n'est pas le cas pour les processus a.out) ;

SHARE représente la quantité de mémoire utilisée par une tâche ;

STAT :l'état d'une tâche est y représentée. L'état est soit S pour sleeping, D pour uninterruptible sleep, R pour running, Z pour zombie, ou T pour stopped (encore appelé traced). Ces états sont modifiés par un traceur **<** pour les processus avec une valeur de priorité négative, **N** pour les processus avec des valeurs de priorités positives et **W** pour un processus dégagé vers l'espace de swap (ceci ne fonctionne correctement que pour les processus du noyau) ;

WCHAN : en fonction de la disponibilité de /boot/psdatabase ou de la table de liens du noyau (/boot/System.map), ceci montre les adresses ou le nom de la fonction du noyau pour lequel la tâche est suspendue ;

TIME définit la durée totale de CPU qu'une tâche a utilisé depuis son lancement. Si le mode cumulatif est activé, ceci inclut aussi la durée de CPU utilisée par les processus-fils qui ont été tués. Le mode cumultif peut être activé à l'aide de l'option S en ligne de commande. Vous pouvez opérer un rafraîchissement de la valeur avec la commande interactive S. La ligne d'en-tête sera modifiée de manière à utiliser le CTIME.

%CPU représente le partage (en fonction des tâches) de la durée d'utilisation de la CPU depuis la dernière mise-à-jour à l'écran. Cette valeur est exprimée en pourcentage de la durée d'utilisation de CPU par processeur.

%MEM représente le partage (en fonction des tâches) de la mémoire physique.

COMMAND désigne le nom (la commande) d'une tâche. Celui-ci sera tronqué s'il est trop long pour être affiché sur une seule ligne. Les tâches en mémoire ont une ligne de commande, mais les tâches dégagés sur l'espace de swap ont seulement le nom du programme entre parenthèses (par exemple : "(getty)").

[10] Format de binaires sous SVr4 (*System V*, version 4).

10.9 Environnement des processus.

Chaque processus en cours d'exécution connaît des expressions littérales du type *var=value*. Ceci signifie qu'un processus peut prendre en compte la valeur de certains paramètres ou variables qu'il peut avoir hérité de son processus-parent. La liste complète des expressions littérales telles que *var=value* constitue l'*environnement* d'un processus et chaque paramètre *var* est une *variable d'environnement*. Chaque processus a son environnement, copie de l'environnement du processus-parent.

Après que vous vous soyez connecté et que vous ayez obtenu une invite de shell, le processus que vous avez utilisé (le shell lui-même) s'est comporté comme tout autre processus, c'est-à-dire qu'il a commencé à utiliser un environnement et les variables qui constituent ce dernier. Pour obtenir une liste complète de ces variables, tapez :

```
set
```

La commande qui suit est utile pour trouver la valeur d'une variable dont vous n'êtes pas sûr du nom :

```
set | grep <expr_reg>
```

Essayez `set | grep PATH` pour connaître la variable d'environnement `PATH` que nous avons vu à la section 5.18.

Le but d'un environnement est essentiellement de fournir un moyen alternatif pour passer des paramètres à un programme (alternatif veut dire : "en plus des arguments passés en ligne de commande"). La différence est qu'un `environnement` est transmis d'un processus au suivant : par exemple, un shell peut disposer d'un jeu de variables d'environnement et exécuter un gestionnaire de fichiers, qui à son tour exécute un traitement de texte. Le traitement de texte hérite son environnement du gestionnaire de fichier qui, lui-même, hérite son environnement du shell. Si, par exemple, vous avez placé une variable `PRINTER` dans le shell, celle-ci sera transmise jusqu'au traitement de texte. Cette technique n'oblige plus à configurer séparément l'imprimante que le traitement de texte doit utiliser.

Testez :

```
X="Bonjour à tous"
echo $X
```

Vous avez défini une variable `X`. A présent, lancez un second bash :

```
bash
```

Vous venez de créer un processus-fils appartenant au processus (le premier bash) dans lequel vous étiez. Tapez :

```
echo $X
```

Vous constaterez que X n'est pas une variable d'environnement du second bash dans lequel vous vous trouvez maintenant. La raison en est que la variable X n'a pas été exportée comme variable d'environnement. De ce fait, elle n'a pas été transmise. Tapez maintenant :

```
exit
```

qui vous ramène au processus-parent (le premier bash). A présent, exécutez les commandes suivantes :

```
export X
bash
echo $X
```

Vous pouvez constater que le nouveau bash connaît X.

Nous venons de fixer une variable arbitraire pour notre convenance personnelle. bash, comme beaucoup d'autres programmes, fixe automatiquement ses propres variables d'environnement. La page de man de bash affiche ces variables. Lorsqu'on parle de désactivation (unsetting) d'une variable, cela signifie qu'il faut utiliser la commande unset <variable>. Il est possible que, pour l'instant, certaines variables vous soient encore très peu familières ; cependant, nous en avons fait la liste parce que nous en aurons besoin à l'avenir.

Ce qui suit est un extrait de la page de man de bash. Vous verrez que les variables appartiennent à deux catégories : celles qui fournissent une information spéciale et sont lues sans être fixées et celles qui configurent le comportement du shell (ou d'autres programmes) et qui peuvent être établies à tout instant.[11]

Variables du shell.[12]

– Les variables suivantes sont fixées par le shell :

PPID représente le PID du shell-parent (variable protégée en écriture) ;

PWD est associée au répertoire de travail courant tel que configurée par la commande **cd** ;

OLDPWD est associée au répertoire de travail précédent telle que définie par la commande **cd** ;

REPLY contient la ligne d'entrée lue par la commande interne **read** quand aucun argument n'est fourni ;

UID contient l'UID de l'utilisateur courant, et est initialisée au démarrage du shell ;

EUID contient l'ID de l'utilisateur effectif associé à l'utilisateur courant, initialisé au démarrage du shell ;

BASH contient le nom complet du chemin utilisé pour invoquer l'instance de **bash** en cours ;

BASH_VERSION correspond au numéro de version de l'instance du **bash** en cours ;

SHLVL (Shell Level) est incrémentée d'une unité, à chaque invocation d'une session de bash ;

RANDOM chaque fois que ce paramètre est référencé, un entier aléatoire est produit. La séquence des nombres entiers aléatoires peut être initialisée en attribuant une valeur à **RANDOM**. Si **RANDOM** est inutilisée (usage de la commande **unset**), elle perd ses propriétés spéciales, même si elle est ultérieurement réinitialisée ;

[11] Merci à Brian Fox et Chet Ranney. (de la part de P. Sheer)

[12] NdT : la traduction française est inspirée de la traduction de la page de man bash, réalisée par Christophe Blaess en 1997 (cette dernière a été réactualisée par Thierry Vigneaud en 1999-2001).

SECONDS chaque fois que ce paramètre est lu, le nombre de secondes écoulés depuis l'appel du shell est renvoyé. Si une valeur est conférée à **SECONDS**, la valeur retournée pour des attributions ultérieures sera le nombre de secondes depuis l'attribution "plus" la valeur conférée initialement. Si **SECONDS** n'est pas initialisée, elle perd ses propriétés spéciales, même si elle est ultérieurement réinitialisée ;

LINENO chaque fois que ce paramètre est référencé, le shell le remplace par un nombre décimal représentant le numéro de ligne courante (commençant à 1) dans le script ou la fonction. Lorsqu'il ne s'agit pas d'un script, il n'est pas garanti que la valeur remplacée ait un sens. Lorsqu'il s'agit d'une fonction, la valeur n'est pas le numéro de la ligne du fichier-source où la commande apparaît (cette information est perdue dès que la fonction est exécutée), mais bien une approximation du nombre de commandes simples exécutées dans le corps de la fonction en cours. Si **LINENO** est détruite (par **unset**), elle perd ses propriétés spéciales, même si elle est ultérieurement réinitialisée (c'est-à-dire recréée) ;

HISTCMD c'est le numéro (ou rang) d'une commande dans l'historique. Si **HISTCMD** est détruite, elle perd ses propriétés spéciales, même si elle est ultérieurement recréée ;

OPTARG est la valeur du dernier argument traité par la commande interne **getopts** (voir **COMMANDES INTERNES DU SHELL**, ci-dessous) ;

OPTIND est l'indice du prochain argument, que doit traité la commande interne **getopts** (voir **COMMANDES INTERNES DU SHELL**, ci-dessous) ;

HOSTTYPE est automatiquement rempli par une chaîne qui décrit de manière univoque le type de machine sur laquelle **bash** est en train d'être exécuté. La valeur par défaut dépend du système ;

OSTYPE est automatiquement rempli par une chaîne qui décrit le système d'exploitation sur laquelle **bash** est en train d'être exécuté. La valeur par défaut est fonction du système.

– Les variables suivantes sont utilisées par le shell. Dans certains cas, **bash** attribue une valeur par défaut aux variables. Ces cas sont notés ci-dessous :

IFS est le *Séparateur de Champ Interne (Internal Field Separator)* qui est utilisé pour séparer les mots après une développement (ou *expansion*) et pour découper les lignes en mots avec la commande interne **read**. La valeur par défaut est "<space><tab><newline>" (<espace><tabulation><retour_chariot>) ;

PATH définit le chemin de recherche des commandes. C'est une liste de répertoires séparée par des doubles points (:), que le shell parcourt pour rechercher les commandes (voir **COMMAND EXECUTION** ci-dessous). Le chemin par défaut dépend du système, et de la configuration choisie par l'administrateur qui installe le **bash**. Une valeur commune est "/usr/gnu/bin :/usr/local/bin :/usr/ucb :/usr/bin :/" ;

HOME définit le répertoire personnel de l'utilisateur courant ; c'est l'argument par défaut de la commande interne **cd** ;

CDPATH est le chemin de recherche pour la commande **cd**. C'est une liste séparée des double points (:), de répertoires dans lequel le shell recherche les répertoires de destination spécifiés par la commande **cd**. Une valeur typique est ". :~ :/usr" ;

ENV si ce paramètre est fixé quand le **bash** exécute un script de shell, sa valeur est considérée comme un nom de fichier contenant les commandes pour initialiser le shell, comme dans *.bashrc*. La valeur de **ENV** est soumise au remplacement de paramètres, la substitution de commandes et l'évaluation (ou expansion) arithmétique avant d'être interprétée comme un nom de chemin. **PATH** n'est pas utilisé pour rechercher le fichier obtenu.

MAIL si ce paramètre correspond à un nom de fichier et que le variable **MAILPATH** n'est pas configurée, **bash** informe l'utilisateur de l'arrivée d'un courrier dans le fichier indiqué.

MAILCHECK souvent exprimée en secondes, cette variable spécifie quand le bash doit vérifier l'arrivée de courriels. La valeur par défaut est de 60 s. Lorsqu'il est temps est écoulé, le shell vérifie l'arrivée de courriel avant de répondre par le symbole d'accueil. Si la variable est supprimée, le shell n'autorise pas le relevé du courriel.

MAILPATH consiste en une liste (séparée par des double-points) de chemins qui doivent être parcourus pour le relevé du courriel. Le message qui doit être affiché en cas de courriel peut être indiqué en séparant le nom du chemin et le message à l'aide du motif

'?'. Le motif $_ indique le nom du fichier de messages. Exemple : MAILPATH='/usr/spool/mail/bfox?" :~/shell-mail?"$_has mail"'. Bash fournit une valeur par défaut pour cette variable, mais la localisation des fichiers de courriel "boîte-à-lettres" d'un utilisateur qu'elle emploie dépendant du système (à titre d'exemple : /usr/spool/mail/**$USER**).

MAIL_WARNING si cette variable est créée, et qu'un fichier (que le bash vérifie pour le courriel) a été consulté, le message "*The mail is in mailfile has been read*" est affiché.

PS1 La valeur de ce paramètre est developpée (voir **PROMPTING** ci-dessous) pour être utilisée comme la chaîne d'accueil principale. La valeur par défaut est "**bash**"**$**" .

PS2 La valeur de ce paramètre est développée pour être utilisée comme chaîne d'accueil secondaire. La valeur par défaut est ">".

PS3 La valeur de ce paramètre est utilisée comme symbole d'accueil de la commande *select* (voir **SHELL GRAMMAR** ci-dessus).

PS4 La valeur de ce paramètre est developpée et affichée avant que chaque commande **bash** ne soit affichée durant un suivi d'exécution. Le premier caractère de PS4 est reproduit autant de fois que nécessaire pour indiquer les différents niveaux d'imbrication. Par défaut, le caractère est "+".

HISTSIZE est le nombre de commandes à mémoriser dans l'historique de commandes (voir **HISTORY** ci-dessous). La valeur par défaut est 500.

HISTFILE est le nom du fichier dans lequel l'historique des commandes est sauvegardé (voir **HISTORY** ci-dessous). La valeur par défaut est *~/.bash_history*. Si cette variable est détruite, l'historique de commandes n'est pas sauvé à la fin d'un shell interactif.

HISTFILESIZE représente le nombre maximum de lignes contenues dans l'historique. Quand une valeur est attribuée à cette variable, le fichier d'historique est tronqué, si nécessaire, pour ne pas contenir davantage que ce nombre de lignes. La valeur par défaut est 500.

OPTERR Si cette variable est posée à 1, **bash** montre les messages d'erreurs produits par la commande interne **getopts** (voir **SHELL BUILTIN COMMANDS** ci-dessous). **OPTERR** est initialisée à 1 chaque fois que le shell est invoqué ou qu'un script de shell est exécuté.

PROMPT_COMMAND Si cette variable est initialisée, elle est exécutée en tant que commande avant d'émettre chaque invite principale.

IGNOREEOF contrôle le comportement du shell à la réception d'un caractère **EOF** (end of file) comme unique entrée. Lorsque cette variable est initialisée, la valeur qu'elle contient est le nombre de caractères **EOF** consécutifs apparaissant comme caractère initial d'une ligne, que bash ignore avant de se terminer. Si la variable existe sans avoir de valeur numérique, ou tout autre valeur, la valeur par défaut sera fixée à 10. Si la variable n'existe pas, **EOF** signifie la fin de saisie dans le shell. Seuls les shells intéractifs sont concernés.

TMOUT si cette variable contient une valeur positive, celle-ci est interprétée comme le nombre de secondes qu'il faut attendre pour toute entrée après l'affichage de l'invite principale. Sil n'y a pas d'entrée avant la fin de ce délai, **Bash** se terminera.

FCEDIT est l'éditeur par défaut utilisé par la commande interne **fc**.

FIGNORE est une liste (séparée par des double-points (:)) de motifs (suffixes) à ignorer lors de la complétement de nom de fichiers (voir **READLINE** ci-dessous). Un nom de fichier dont le suffixe correspond à une des entrées dans **FIGNORE** est exclus de la liste des noms de fichiers. Une valeur-type est ".o :~".

INPUTRC est le nom de fichier pour la configuration de readline. Il prévaut sur la valeur par défaut de ~/.inputrc (voir **READLINE** ci-dessous).

notify si la valeur existe, le bash affiche un rapport dès qu'une tâche en arrière-plan est terminée. Sinon, il attendra l'affichage de la prochaine invite principale (voir aussi l'option -b de la commande interne **set**).

history_control

HISTCONTROL si cette variable contient la valeur *ignorespace*, les lignes commençant par le caractère **espacement** (**space**) ne sont pas conservées dans l'historique. Si la valeur est *ignoredups*, la ligne correspondant à la dernière ligne de l'historique ne sera pas mémorisée. La valeur *ignoreboth* combine les deux comportements. Si la variable est détruite, ou si toute autre valeur que les trois précitées lui est attribuée, toutes les lignes lues par l'interpréteur sont sauvées dans l'historique.

command_orientated_history si cette variable est créée, **bash** tente de sauver toutes les lignes des commandes multi-lignes sous une même entrée dans l'historique. Ceci facilite la réédition de commandes sur plusieurs lignes.

glob_dot_filenames Si cette variable est activée, le **bash** inclut les noms de fichiers commençant par un "." lors complètement des nom de chemin (pathname).

allow_null_glob_expansion Si cette variable est activée, le **bash** permet aux motifs des noms de chemin ne correspondant à aucun fichier d'être complétés par une chaîne nulle plutôt que par leur propre valeur (voir *Pathname expansion* ci-dessous).

histchars ce sont les deux ou trois caractères qui contrôlent le complètement de l'historique et le découpage en éléments lexicaux (voir **HISTORY EXPANSION** ci-dessous). Le premier caractère est le caractère de complètement de l'historique, c'est-à-dire le caractère qui signale le début du complètement de l'historique, normalement "!". Le second caractère ou caractère de substitution rapide (*quick substitution*) est utilisé comme abrégé pour exécuter à nouveau la commande précédemment entrée, en subsitutant une chaîne par une autre dans la commande. La valeur par défaut est "^". Le troisième caractère est optionnel et indique que le reste de la ligne est un commentaire lorsqu'il précède un nom (normallement il s'agit de"#"). Le caractère de commentaire de l'historique produit un saut vers les mots suivants d'une ligne. Il n'induit pas nécessairement l'interpéteur synatxique du shell à traiter le reste d'un ligne comme un commentaire.

nolinks si cette variable est activée, le shell ne suit pas les liens symboliques lors des exécutions de commandes qmodifiant le répertoire de travail courant. Au lieu de cela, elle permet d'utiliser la structure physique du répertoire. Par défaut, le **bash** suit la chaîne des répertoires lorsqu'il exécute des commandes modifiant le répertoire courant, telles que **cd**. Voir aussi la description de l'option -P de la commande interne **set** (**SHELL BUILTIN COMMAND** ci-dessous).

hostname_completion_file

HOSTFILE contient le nom d'un fichier dans le même format que */etc/hosts* qui devrait être consulté par le shell lorsqu'il doit compléter un nom d'hôte. Ce fichier peut être modifié interactivement ; au complétement de nom d'hôte (*hostname*) à la suite de la modification, **bash** ajoute le contenu du nouveau fichier à sa base de données existante.

noclobber si cette variable est activée, le bash n'écrase pas le contenu d'un fichier existant lorsque les opérateurs de redirection >, >**&** et <> sont utilisés. Cette variable peut être annulée lors de la création de fichiers de sortie en utilisant l'opérateur de redirection > – au lieu de > (voir aussi l'option -C de la commande interne **set**).

auto-resume cette variable contrôle la manière dont le shell interagit avec un utilisateur et le contrôle des tâches. Si cette variable existe, les commandes simples (en un seul mot) sans redirections sont traitées comme des candidats pour la reprise d'une tâche existante mais arrêtée. Il n'y a pas d'ambiguïté permise : s'il y a plus d'une tâche commençant avec la chaîne saisie, le choix se porte sur celle d'accès le plus récent. Dans ce contexte, le *nom* (*name*) de la tâche arrêtée est la ligne de commande utilisée pour son démarrage. Si la variable est posée à *exact*, la chaîne fournie doit correspondre exactement au nom d'une tâche arrêtée; si la valeur est *substring*, la chaîne doit correspondre à une sous-chaîne du nom de la tâche arrêtée. La valeur *substring* fournit une fonctionnalité analogue à l'id de job **%?** (voir le **JOB CONTROL** ci-dessous). Si la valeur est autre, la chaîne fournie doit être le préfixe du nom d'une tâche arrêtée ; ceci fournit une fonctionnalité analogue à l'id de job %.

no_exit_on_failed_exec si cette variable existe, un shell non-interactif ne se terminera pas s'il ne peut exécuter le fichier spécifié dans la commande interne **exec**. Un shell interactif ne se termine pas si **exec** échoue.

cdable_vars si cette variable est activée, un argument de la commande interne **cd** qui ne serait pas un répertoire sera considéré comme le nom d'une variable dont le contenu est le répertoire à modifier.

Chapitre 11

Courriel

En général, les utilisateurs d'un ordinateur entre en contact avec l'internet via la messagerie électronique qui permet d'envoyer et de recevoir du courriel (*electronic mail* ou *e-mail* ou encore mél, en français). Bien qu'usuellement, le courriel soit utilisé dans un environnement graphique, nous vous montrons ici comment il a été conçu pour un système multi-utilisateur. Dans une large mesure, ce qui est décrit ci-dessous constitue les fondements de tout système de messagerie.

Un message courriel est un texte en un bloc, envoyé par un utilisateur à l'attention d'un autre utilisateur et ce, à l'aide d'une commande ou d'un programme de messagerie. Une rubrique sujet définit souvent le thème du message. L'idée de base est qu'un message puisse être envoyé à quelqu'un qui n'est pas nécessairement connecté et que ce message puisse être stocké jusqu'au moment où le destinataire décide de le lire. La forme d'une adresse de courriel vous est probablement familière ; par exemple : `bruce@kangeroo.co.au`. Ceci signifie que `bruce` a un compte sur un ordinateur appelé `kangeroo.co.au`, ce qui veut souvent dire que cet utilisateur peut se connecter en tant que `bruce` sur cette machine-là. Le texte derrière le signe `@` indique toujours le nom d'une machine. De nos jours, l'internet ne respecte plus totalement ceci, mais il y a toujours une machine sur laquelle `bruce` *doit* avoir un compte sur lequel le courriel peut être envoyé. [Le système d'exploitation de cette machine est usuellement un UNIX].

Parfois, les adresses de messagerie électronique sont écrites de manière plus conviviale comme `Bruce Wallaby <bruce@kangeroo.co.au>` ou encore : `bruce@kangeroo.co.au (Bruce Wallaby)`. Si c'est le cas, les caractères qui entourent l'adresse sont purement cosmétiques ; seule l'expression `bruce@kangeroo.co.au` sera réellement exploitée.

Lorsque vous recevez un courriel (soit d'un autre utilisateur du système sur lequel vous travaillez, soit d'un utilisateur travaillant sur une autre machine), celui-ci est ajouté au fichier `/var/spool/mail/<nom_utilisateur>` appelé communément "boîte à messages" (*mail file* ou *mailbox*) ; `<nom_utilisateur>` désigne votre identifiant. Vous exécutez alors un des programmes qui interprètent votre boîte à messages, vous permettant ainsi de feuilleter le fichier en une séquence de messages, de les lire et éventuellement d'y répondre.

Un exemple réaliste d'ajout à cette boîte de messagerie se présente comme ceci :

```
From mands@inetafrica.com Mon Jun 1 21 :20 :21 1998
Return-Path : <mands@inetafrica.com>
Received : from pizza.cranzgot.co.za (root@pizza.cranzgot.co.za [192.168.2.254])
         by onion.cranzgot.co.za (8.8.7/8.8.7) with ESMTP id VAA11942
         for <psheer@icon.co.za>; Mon, 1 Jun 1998 21 :20 :20 +0200
Received : from mail450.icon.co.za (mail450.icon.ca.za [196.26.208.3])
         by pizza.cranzgot.co.za (8.8.7/8.8.7) with ESMTP id VAA19357
         for <psheer@icon.co.za>; Mon, 1 Jun 1998 21 :17 :06 +0200
Received : from smtp02.inet.africa.com (smtp02.inetafrica.com [196.26.208.3])
         by mail450.icon.co.za (8.8.8/8.8.8) with SMTP id VAA02315
         for <psheer@icon.ca.za>; Mon, 1 Jun 1998 21 :24 :21 +0200 (GMT)
Received : from default [196.31.19.216] (fullmoon)
         by smtp02.inetafrica.com with stmp (Exim 1.73 #1)
         id OygTDL-00041u-00, Mon, 1 Jun 1998 13 :57 :20 +0200
Message-ID :<357296DF.60A3@inetafrica.com>
Date : Mon, 01 Jun 1998 13 :56 :15 +0200
From : a person <mands@inetafrica.com>
Reply-To :mands@inetafrica.com
Organisation : private
X-Mailer : Mozilla 3.01 (Win95; I)
MIME-Version : 1.0
To : paul sheer <psheer@icon.co.za>
Subject : hello
Content-Type : text/plain; charset=us-ascii
Content-Transfer-Encoding : 7bit
Status : RO
X-Status : A

hey paul
ist me
how r u doing
i am well
what u been upot
hows life
hope you are well

amanda
```

Les courriels commencent par `From` suivi d'un espace. Ensuite, vient l'*en-tête* (*header*) qui indique comment le message a été routé jusqu'à atteindre votre boîte de messagerie, quel en est l'expéditeur, quel est l'adresse où les réponses aboutiront, le sujet du courriel ainsi que d'autres *champs d'en-tête* (*mail header fields*). En l'occurrence, l'en-tête est plus grand que le texte lui-même. Examinez cet en-tête attentivement.

L'en-tête est sépararé du texte par une ligne blanche. Le corps du message suit immédiatement. Dans le fichier, l'en-tête suivant commencera à nouveau par `From`. Ces "`From`" n'apparaissent jamais comme premier mot d'une ligne du corps. Si c'était le cas, la boîte de messagerie serait corrompue.

Certains logiciels de traitement du courriel enregistrent les messages dans un format différent. Cependant, le format ci-dessus est typique des systèmes UNIX (on l'appelle le format *mbox*). Le format *Maildir* est intéressant car il n'enregistre pas les messages de manière contiguë dans un seul fichier. Chaque message est placé dans un fichier séparé à l'intérieur d'un répertoire. Le nom du répertoire est alors défini comme le "fichier" de la boîte de courrier électronique. Par défaut *Maildir* utilise un répertoire `Maildir` dans le répertoire home de l'utilisateur.

11.1 Recevoir et envoyer du courriel.

La manière la plus simple d'envoyer un message électronique consiste à utiliser la commande `mail`.[1] Tapez `mail -s "bonjour" <nom_utilisateur>`. Le programme `mail` attendra que vous saisissiez votre message. Lorsque vous avez terminé, entrez un `.` tel quel sur une ligne. Le "nom d'utilisateur" sera celui d'un autre utilisateur de votre système. Si personne d'autre n'est sur votre système, envoyez le message à `root` avec `mail -s "Bonjour" root` ou `mail -s "Bonjour" root@localhost` (si le signe `@` n'est pas mentionné, alors la machine locale –`localhost`– est impliquée). La méthode pour envoyer des fichiers par courriel est discutée à la section 13.6.

Vous utiliserez la commande `mail` pour visualiser ce que contient votre boîte électronique. C'est une fonctionnalité assez sommaire en comparaison des programmes de messagerie graphique modernes mais c'est probablement le seul programme de messagerie qui peut manipuler des messageries de toute taille. Il peut arriver que votre messagerie ait une taille supérieure au giga-octet, et `mail` est la seule manière d'y supprimer des messages. Pour voir ce qui se trouve dans la boîte de messagerie, tapez `mail` et ensuite `z` pour lire la fenêtre suivante. `z-` permet de remonter d'une fenêtre. La plupart des commandes fonctionnent sous forme d'une commande suivie d'un nombre : par exemple, `delete 14` ou `reply 7`. Le numéro du message est accompagné dans la colonne de gauche d'un `N` (pour "`N`ew message").

Pour des programmes de messagerie parfaitement au point et fonctionnant en mode console (ce qu'il convient d'appeler des *clients* de messagerie), essayez `mutt` et `pine` [la licence de pine n'est pas libre].

Il existe actuellement de nombreux programmes graphiques à un stade plus ou moins évolué de développement. Notez que `balsa` est un des meilleurs programmes de messagerie existant. On notera aussi l'usage de clients graphiques comme `thunderbird`, `evolution`, `kmail`, etc.[2]

11.2 Protocole SMTP - Envoyer le courriel au port 25.

Pour envoyer du courriel, vous ne devez pas nécessairement utiliser un client de messagerie. Le client obéit seulement au protocole *SMTP* (Simple Mail Transfer Protocol) dans lequel vous faites vos saisies à partir du clavier.

Par exemple, vous pouvez envoyer un mél par telnet via le port 25 d'une machine qui a un *MTA* actif (Mail Transfer Agent –appelé aussi *démon*[3] *de messagerie* ou *serveur de mél*; en anglais : *mailer daemon* ou *mail server*). Le terme démon désigne un programme qui fonctionne silencieusement, c'est-à-dire sans l'intervention d'un utilisateur.

Ceci est, en fait, la manière dont les messages anonymes ou messages de spam sont envoyés sur internet [Le spam est un terme utilisé pour désigner tout

[1] NdT : le paquet à installer peut être `mailx`.

[2] La suite `mozilla` (client courriel, navigateur web, composeur, gestionnaire d'adresses) est désormais remplacée par `seamonkey`.

[3] NdT : ce terme est la francisation du mot *daemon* (Disk And Extension MONitor) qui désigne un processus s'exécutant en tâche de fond après avoir été invoqué de manière non-manuelle. Les processus de ce type attendent un signal ou la réalisation d'une condition.

courriel non-sollicité, c'est-à-dire des messages indésirables postés en masse à un nombre important d'adresses électroniques données]. Un démon de messagerie fonctionne dans la plupart des petites institutions à travers le monde et doit accomplir la tâche simple de recevoir les requêtes de mél et de les relayer vers d'autres serveurs de messageries. Essayez ceci, par exemple (évidemment, il faut substituer `mail@cranzgot.co.za` par le nom d'un serveur de messagerie qui est en activité) :

```
[root@cericon]# telnet mail.cranzgot.co.za 25
Trying 192.168.2.1...
Connected to 192.168.2.1.
Escape character is '^]'.
220 onion.cranzgot.co.za ESMTP Sendmail 8.9.3/8.9.3; Wed, 2 Feb 2000 14 :54 :47 +0200
HELO cericoncranzgot.co.za
250 onion.cranzgot.co.za Hello cericon.ctn.cranzgot.co.za [192.168.3.9], pleased to meet
you
MAIL FROM : psheer@icon.co.za
250 psheer@icon.co.za... Sender ok
RCPT TO : mands@inetafrica.com
250 mands@inetafrica.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject : just to say hi

hi there
heres a short message

.
250 OAA04620 Message accepted for delivery
QUIT
221 onion.cranzgot.co.za closing connexion
connexion closed by foreign host.
[root@cericon]#
```

L'information se touvant dans l'encadré permet d'envoyer le message "`hi there heres a short message`" à l'adresse `mands@inetafrica.com` (= R*e*C*i*-P*ient*). Naturellement, on peut entrer toute adresse souhaitée et il peut être difficile de déterminer qui a envoyé le message. Dans cet exemple, le `Subject :` est le seul en-tête, bien qu'il ne soit pas obligatoire d'en fournir un.

Il se peut qu'ayant essayer ceci, vous ayez reçu un message d'erreur sans appel. Ceci pourrait être causé par un MTA qui n'est pas configuré pour relayer des messages hormis ceux pour des machines de confiance, disons (c'est-à-dire des machines faisant partie d'un système organisé). Ceci évite les messages anonymes.

En revanche, si vous vous connectez au serveur même de messagerie d'un utilisateur, il devra recevoir le message, quiconque l'ait envoyé. Donc, ce qui précède est un moyen efficace de fournir un bogue **au départ** d'une adresse et d'envoyer le message presque anonymement. Par "presque", il faut entendre que le serveur de messagerie détectera la machine sur laquelle vous travaillez ainsi que le moment de la connexion. Il n'y a pas de connexion totalement anonyme du point de vue d'un serveur de messagerie correctement configuré.

La technique décrite ci-dessus est souvent le seul moyen de tester efficacement un serveur de messagerie.

Chapitre 12

Comptes d'utilisateurs et droits.

Par essence, UNIX est un système multi-utilisateur et multi-tâche. Chaque utilisateur possède un répertoire personnel `/home/<nom_utilisateur>` dans lequel ses fichiers sont enregistrés tout en restant à l'abri du regard des autres utilisateurs.

Lorsque vous utilisez la machine en tant que `superutilisateur` (`root`, ou administrateur), vous avez un accès total à chaque fichier du système. Le répertoire personnel du `superutilisateur` est `/root` et ce répertoire est monté sur la racine (`/`) [Notez que, dans la terminologie anglo-saxonne, il y a des ambiguïtés : le répertoire-parent de votre système (`/`) est appelé `root`. Le superutilisateur s'appelle `root` et son répertoire personnel est `/root`].

Sauf en ce qui concerne le superutilisateur, les autres utilisateurs ont un accès *limité* aux fichiers et aux répertoires. Utilisez toujours votre machine comme un utilisateur classique de manière à ne vous connecter en `root` (superutilisateur) que pour des travaux d'administration. Dans ce chapitre, nous montrons comment créer manuellement et automatiquement des comptes utilisateurs.

L'ensemble des utilisateurs est divisé en groupes (*groups*). Un utilisateur peut appartenir à différents groupes et il n'y a pas de limitation sur le nombre de groupes dans le système. Chaque groupe est défini par une liste d'utilisateurs. Par ailleurs, chaque utilisateur peut avoir un groupe à son nom (c'est-à-dire un groupe ayant le même nom que l'identifiant).

12.1 Droits associés aux fichiers.

Chaque fichier sur un système donné est la *propriété* d'un utilisateur particulier ; il est aussi la *propriété* d'un groupe particulier. Quand vous exécutez `ls -al`, vous pouvez voir dans la troisième colonne le nom de l'utilisateur qui possède un fichier donné. Le nom du groupe qui possède ce fichier est dans la quatrième colonne (ce nom est souvent identique à celui de l'utilisateur : ceci indique que le groupe du fichier est un groupe auquel seul l'utilisateur appartient). Pour changer les droits de propriétés d'un fichier, utilisez simplement la commande `chown` (*change ownerships*), comme suit :

```
chown <utilisateur> [ :<groupe>] <nom_de_fichier>
```

12.2 Le fichier des mots de passe : /etc/passwd.

Le fichier /etc/passwd est le seul dans tout le système où le nom d'un utilisateur est enregistré. [Des exceptions à cette règle : (i) plusieurs schémas d'authentification distribués et (ii) le paquet logiciel Samba. Cependant, pour l'instant, vous ne devez pas vous inquiétez de cela]. Une fois qu'un utilisateur est additionné à cette liste, on dit qu'il "existe" sur le système. Si vous pensiez que les comptes des utilisateurs étaient enregistrés dans un coin sombre et inaccessible de votre système, ceci devrait dissiper cette idée. Listez le fichier *password* bien connu des administrateurs, avec `less` :

```
root :x :0 :0 :root :/root :/bin/bash
bin :x :1 :1 :bin :/bin :/bin/false
daemon :x :2 :2 :daemon :/sbin :/bin/false
adm :x :3 :4 :adm :/var/adm :/bin/false
lp :x :4 :7 :lp :/var/spool/lpd :/bin/false
sync :x :5 :0 :sync :/sbin :/bin/sync
shutdown :x :6 :0 :shutdown :/sbin :/sbin/shutdown
halt :x :7 :0 :halt :/sbin :/sbin/halt
mail :x :8 :12 :mail :/var/spool/mail :/bin/false
news :x :9 :13 :news :/usr/lib/news :/bin/false
uucp :x :10 :14 :uucp :/var/spool/uucppublic :/bin/false
operator :x :11 :0 :operator :/root :/bin/bash
ftp :x :21 :21 : :/home/ftp :/bin/false
nobody :x :65534 :65534 :nobody :/ :/bin/false
alias :x :501 :501 : :/var/qmail/alias :/bin/bash
paul :x :509 :510 :Paul Sheer :/home/paul :/bin/bash
jack :x :511 :512 :Jack Robbins :/home/jack :/bin/bash
silvia :x :511 :512 :Silvia Smith :/home/silvia :/bin/bash
```

Ci-dessus se trouve un extrait d'un fichier *password* typique. Chaque utilisateur est enregistré sur une ligne différente. La plupart d'entre elles ne correspondent pas à des utilisateurs physiques mais sont associés à des programmes.

Chaque ligne comprend sept champs (*fields*) séparés par une paire de points. Par exemple, le compte appelé `jack` se décompose de la manière suivante :

jack est l'identifiant de l'utilisateur. Il devrait être composé de minuscule et de nombres. D'autres caractères sont permis, mais cela n'est pas souhaitable. En particulier, il ne devrait *jamais* y avoir deux noms d'utilisateurs ne différant que par une majuscule.

x est un signe masquant le mot de passe crypté et indiquant que ce dernier se trouve dans le fichier /etc/shadow. Le fichier de mots de passe *cryptés* (*shadow password*) est un fichier additionnel des systèmes UNIX. Il contient des informations supplémentaires à propos des utilisateurs.

511 ce nombre est l'UID ou User IDentifier de l'utilisateur [il est utilisé par les programmes pour identifier tout utilisateur. En fait, en interne, seul l'UID -et non le nom de connexion- est utilisé].

512 ce nombre est le GID ou Group IDentifier [Le commentaire décrit ci-dessus s'applique aux groupes. Ceux-ci sont discutés plus tard].

Jack Robbins est le nom complet de l'utilisateur.

/home/jack représente le répertoire personnel de l'utilisateur `jack`. La variable d'environnement `HOME` sera fixée à cette valeur lorsque l'utilisateur se connectera.

/bin/bash est le shell à démarrer lors de la connexion de l'utilisateur.

12.3 Le fichier `/etc/shadow`.

Le problème des fichiers `passwd` traditionnels est qu'ils sont lisibles par tous (*world readable*) [n'importe qui sur le système peut lire le fichier `passwd`]. Ceci permet aux programmes d'y extraire des informations au sujet d'un utilisateur, telles que son nom complet. Ceci veut donc dire que tout le monde peut lire le mot de passe crypté dans le second champ. Aussi, tout le monde peut-il copier le champ "mot de passe" d'un utilisateur et essayer des milliards de mots de passe jusqu'à ce qu'un des mots cryptés ainsi composé corresponde au mot de passe crypté de l'utilisateur. Si vous avez une centaine d'utilisateurs sur votre système, il se peut que plusieurs d'entre eux choisissent un mot du dictionnaire comme mot de passe. Les attaques par le dictionnaire consiste à tester les 80.000 mots d'anglais courant pour trouver une correspondance. Si vous pensez qu'il est astucieux d'ajouter un nombre en préfixe à un mot du dictionnaire, sachez que les algorithmes de craquage des mots de passe savent ça aussi [comme c'est d'ailleurs le cas pour d'autres astuces]. C'est pour résoudre ce problème que les mots de passe cryptés ont été inventés. Le fichier des mots de passe cryptés est exclusivement utilisé lors de l'étape d'authentification [en vérifiant que l'utilisateur est le véritable détenteur du compte]. Il n'est pas lisible par tous et aucun utilisateur normal n'a le droit de voir le champ "mot de passe chiffré". Il n'y a pas d'information dans ce fichier nécessaire aux programmes. Les champs du fichier `/etc/shadow` sont séparés par des doubles points comme c'est le cas du fichier `/etc/passwd` :

```
jack :Q,Jpl.or6u2e7 :10795 :0 :99999 :7 :-1 :-1 :134537220
```

jack est le nom d'utilisateur

Q,Jpl.or6u2e7 est le mot de passe chiffré résultant d'une transformation irréversible d'un mot de passe à 8 caractères. Un algorithme mathématique est appliqué au mot de passe de manière à produire un résultat unique pour chaque mot de passe. Pour s'en convaincre, le mot de passe `Loghimin` (plutôt simple) est converti en : `1Z1F.0VSRRucs` dans le fichier `/etc/shadow`. Le mot quasi-identique `loghimin` donne une transformation totalement différente : `CavHIpD1w.cmg`. Donc pour retrouver le mot de passe à partir du mot chiffré, il faudrait essayer chaque mot possible. Bien qu'une attaque brutale de ce type soit considérée comme coûteuse du point de vue de la charge de calcul, elle n'est pas impossible. Pour vérifier si un mot de passe est bon, il suffit de lui appliquer l'algorithme permettant de vérifier la correspondance avec le mot de passe crypté. C'est comme cela que fonctionne la commande `login`. Si vous voyez un `*` à la place du mot chiffré, cela signifie que le compte a été désactivé.

10795 c'est le nombre de jours entre le 1er janvier 1970 et la date de dernier changement du mot de passe.

0 c'est le nombre de jours avant que le mot de passe ne puisse être changé. Usuellement, la valeur est = 0. Ce champ n'est pas souvent utilisé.

99999 correspond au nombre de jours après quoi le mot de passe doit être modifié. Ce champ est rarement utilisé. Par défaut, sa valeur est 99999.

7 nombre de jours pour lequel un utilisateur est averti que son mot de passe expirera.

-1 est le nombre de jours après quoi le mot de passe expire de sorte que le compte est désactivé. -1 est utilisé pour indiquer un nombre infini de jours (ce qui veut dire que cette propriété du compte n'est pas considérée).

-1 est le nombre de jours entre le 1er janvier 1970 et la date de désactivation du compte.

134537220 est un indicateur réservé à un usage ultérieur.

12.4 La commande `groups` et `/etc/group`.

Vous désirerez peut-être donner les mêmes droits d'accès à un certain nombre d'utilisateurs de votre système LINUX. Supposons, par exemple, cinq utilisateurs qui seront autorisés à accéder à un fichier et dix autres auxquels vous donnerez le droit d'exécuter un programme. Vous pouvez faire deux groupes, l'un appelé `previl` et l'autre `wproc`. Le fichier et les répertoires appropriés seront attribués sélectivement comme ceci, par exemple :

```
chown root :previl /home/un_fichier
chown root :wproc /usr/lib/wproc
```

Les droits [expliqués au chapitre 15] imposent le type d'accès, mais le fichier ou le répertoire doit au préalable être la propriété de ce groupe avant que ne se discutent les droits d'accès.

Le fichier `/etc/group` présente aussi des champs séparés par des double-points. Une ligne typique se présente comme ceci :

```
wproc :x :524 :jack,mary,henry,arthur,sue,lester,fred,sally
```

wproc est le nom du groupe. Il devrait également y avoir un utilisateur portant ce nom.

x c'est le mot de passe du groupe. Ce champ est usuellement occupé par un `x` et n'est pas utilisé.

524 est le GID ou Group IDentifier qui doit être unique dans le fichier group.

jack,mary,henry,arthur,sue,lester,fred,sally forment la liste des utilisateurs appartenant à ce groupe. Il doit y avoir une virgule (sans espace) comme séparateur.

Vous pouvez bien sûr étudier le fichier `/etc/group` pour trouver les groupes auquel un utilisateur appartient [ce qui n'est pas la même chose –et c'est facile à voir– que de déterminer quels utilisateurs forment tel ou tel groupe]. Toutefois, quand il y a de nombreux groupes, cette méthode est fastidieuse par le balayage complet du fichier qu'elle impose. La commande `groups` affiche cette information.

12.5 Création manuelle d'un compte d'utilisateur.

Les étapes suivantes sont requises pour la création manuelle d'un compte d'utilisateur :

/etc/passwd Pour créer une entrée dans ce fichier, modifiez simplement celui-ci et copiez une ligne existante [Lors de la modification de fichiers de configuration, n'écrivez jamais une ligne si celle-ci présente un format particulier. Dupliquez toujours une entrée existante connue pour fonctionner correctement, et modifiez-la ensuite. Cette pratique évitera des erreurs]. Ajoutez toujours les utilisateurs dans le bas du fichier et essayez de préserver le style du fichier. C'est-à-dire que si vous voyez que des nombres augmentent de ligne en ligne, poursuivez avec des nombres croissants pour la nouvelle entrée ; si vous ajoutez un utilisateur normal, faites-le à la suite des lignes existantes qui se rapportent aux utilisateurs normaux. Chaque utilisateur doit avoir un UID unique et devrait aussi avoir un GID unique. Ainsi, si vous ajoutez une ligne à la fin du fichier, augmentez d'une unité les UID et GID de la dernière ligne servant de modèle.

/etc/shadow Créez une nouvelle entrée pour le mot de passe crypté (*shadow password*). A cette étape, vous n'en connaissez pas la valeur. Donc, mettez un *. Vous déterminerez le mot de passe ultérieurement avec la commande `passwd`.

/etc/group Créez une nouvelle entrée pour le groupe de l'utilisateur. Vérifiez bien que le nombre du groupe corresponde à celui indiqué dans le fichier `passwd`.

/etc/skel Ce répertoire contient un modèle de répertoire personnel pour les utilisateurs. Copiez le répertoire en entier et tout son contenu dans `/home` ; et renommez-le pour l'utilisateur. Dans le cas de notre exemple, `paul` ; vous obtiendrez ainsi un `/home/paul`.

Droits du répertoire personnel Vous devez, à présent, modifier les droits de propriétés du répertoire personnel pour qu'ils correspondent au nouvel utilisateur. La commande `chown -R paul :paul /home/paul` réalise cet objectif.

Mot de passe Utilisez la commande `passwd <nom_utilisateur>` pour attribuer un mot de passe.

12.6 Méthode automatique : `useradd` et `groupadd`.

La méthode décrite précédemment est plutôt fastidieuse. Les commandes réalisant les mises à jour automatiquement sont `useradd`, `userdel` et `usermod`. Les pages de `man` expliquent ces commandes en détail. Notez que les divers types d'UNIX présentent des commandes différentes. Certaines distributions offrent des interfaces graphiques ou web comme assistants à la création, suppression, modification des utilisateurs.

Par ailleurs, les commandes `groupadd`, `groupdel` et `groupmod` agissent similairement en ce qui concerne les groupes.

12.7 Connexion utilisateur.

Il est possible de basculer d'un utilisateur à un autre aussi bien que de voir l'état de connexion d'un utilisateur (ou le vôtre). La connexion procède selon une méthode "silencieuse" qu'il est important de bien comprendre.

12.7.1 La commande `login`.

Un utilisateur accède le plus souvent au système via le programme `login`. Ce programme analyse l'IUD et le GID à partir des fichiers `passwd` et `group`, puis il authentifie l'utilisateur.

Ce qui suit est extrait de la page de man de `login`, et détaille la méthode :

login est utilisé lorsqu'on s'engage sur un système. Il peut aussi être utilisé pour basculer d'un utilisateur à un autre à tout moment (la plupart des shells modernes possèdent cette fonctionnalité).

Si aucun argument n'est passé, **login** s'adresse à l'utilisateur.

Si l'utilisateur n'est pas *root*, et si */etc/nologin* existe, le contenu de ce fichier est affiché à l'écran et la connexion se termine. C'est typiquement le cas pour prévenir les connexions lorsque le système est en cours de démontage.

Si des restrictions spéciales d'accès sont spécifiées, à l'endroit de l'utilisateur dans */etc/usertty*, celles-ci doivent être respectées, ou alors, la tentative de connexion avortera et un message de **syslog** sera produit [syslog écrit tous les messages du système dans le fichier */var/log/messages*]. Voir la section "Special Access Restrictions".

Si l'utilisateur est *root*, la connexion peut avoir lieu sur un tty listé dans */etc/securetty* [Si ce fichier n'est pas présent, les connexions du superutilisateur seront permises n'importe où. Il est pertinent de supprimer ce fichier si votre machine est protégée par un pare-feu (firewall) et que vous souhaitez vous connectez aisément à partir d'une autre machine de votre réseau LAN. Si `/etc/securetty` est présent les connexions seront seulement utilisées pour les terminaux que ce fichier définit]. Les échecs seront consignés via **syslog**.

Après que ces conditions aient été testées, le mot de passe sera demandé et vérifié (si un mot de passe est requis pour l'utilisateur). Dix essais sont autorisés avant que la connexion ne s'interrompe, mais après les trois premiers essais, la réponse devient plus lente. Les échecs de connexion sont consignés via **syslog**. Cette fonctionnalité permet aussi de mentionner les connexions réussies du superutilisateur.

Si le fichier *.hushlogin* existe, une connexion "douce" est réalisée (elle désactive la vérification des messages électroniques et l'affichage de la date de dernière connexion et du message du jour). Autrement, si */var/log/lastlog* existe, la date de dernière connexion est affichée et la connexion courante est enregistrée.

Les tâches d'administration comme la mise en place de l'UID et du GID d'un tty sont réalisées. La variable d'environnement TERM est préservée, si elle existe (les autres variables d'environnement sont préservées si l'option **-p** est utilisée). Les variables d'environnement HOME, PATH, SHELL, TERM, MAIL et LOGNAME sont créées. Par défaut, la variable PATH concerne les utilisateurs normaux, */usr/local/bin :/bin :/usr/bin :.* [notez que le . qui représente le répertoire courant est repris dans PATH] et */sbin :/bin :/usr/sbin/usr/bin* pour le superutilisateur. Enfin, s'il n'y a pas de connexion douce, le message du jour est affiché et le fichier contenant le nom de l'utilisateur dans */usr/spool/mail* est vérifié. Si sa taille n'est pas nulle, un message est affiché.

Le shell d'utilisateur est alors démarré. Si aucun shell n'est spécifié dans **/etc/passwd**, **/bin/sh** est utilisé. S'il n'y a pas de répertoire déclaré dans */etc/passwd*, / est utilisée (une vérification est réalisée pour s'assurer que le répertoire personnel contient le fichier *.hushlogin*).

12.7.2 *set user*, la commande `su`.

Pour devenir temporairement un autre utilisateur, vous pouvez utiliser le programme `su` :

```
su paul
```

Cette commande vous invite à entrer votre mot de passe (sauf si vous êtes superutilisateur). Ceci ne fait rien de plus que de permettre à l'utilisateur courant d'obtenir les droits d'accès de `paul`. Les variables d'environnement restent identiques pour la plupart. Les variables `HOME`, `LOGNAME` et `USER` sont adaptées à l'utilisateur `paul` mais toutes les autres variables seront héritées. `su` ne permet donc pas une connexion normale.

Pour obtenir l'équivalent d'une connexion classique avec `su`, effectuez la commande suivante :

```
su - paul
```

Ainsi, les scripts d'initialisation seront exécutés. Il s'agit de ceux démarrés lors de la connexion de l'utilisateur lui-même. [Dans ce cas, le shell subséquent est démarré avec un - avant l'argument d'ordre zéro. Ceci fait que le shell lit le profil personnel de l'utilisateur. La commande `login` pratique de la même façon]. Par conséquent, après avoir exécuté `su` avec l'option `-`, vous vous connectez comme si vous aviez utilisé la commande `login`.

12.7.3 Les commandes `who`, `w`, `users` pour déterminer qui est connecté.

`who` et `w` affichent une liste des utilisateurs connectés au système ainsi que leur consommation CPU et d'autres informations statistiques. `who --help` donne :

```
Usage : who [OPTION]... [ FILE | ARG1 ARG2 ]
   -a, --all same as -b -d --login -p -r -t -T -u
   -b, --boot time of last system boot
   -d, --dead print dead processes
   -H, --heading print line of column headings
   -i, --idle add idle time as HOURS :MINUTES, . or old
       (deprecated, use -u)
   --login print system login processes
       (equivalent to SUS -l)
   -l, --lookup attempt to canonicalize hostnames via DNS
       (-l is deprecated, use --lookup)
   -m only hostname and user associated with stdin
   -p, --process print active processes spawned by init
   -q, --count all login names and number of users logged on
   -r, --runlevel print current runlevel
   -s, --short print only name, line, and time (default)
   -t, --time print last system clock change
   -T, -w, --mesg add user's message status as +, - or?
   -u, --users list users logged in
       --message same as -T
       --writable same as -T
       --help display this help and exit
       --version output version information and exit


If FILE is not specified, use /var/run/utmp. /var/log/wtmp as FILE is common.
If ARG1 ARG2 given, -m presumed : 'am i' or 'mom likes' are usual.
Report bugs to <bug-coreutils@gnu.org>.
```

Les pages `info` pour cette commande donnent davantage d'information. La durée d'inactivité (*idle time*) indique la période écoulée depuis qu'un utilisateur

n'a pas pressé de touches. Le plus souvent, c'est la commande `who - Huw` qui est utilisée.

La commande `w` est similaire. Un extrait de la page de page de `man` indique :

> w affiche l'information à propos des utilisateurs actuellement connectés sur le système, et à propos des processus en cours. L'en-tête montre dans l'ordre : la date, le temps écoulé depuis que le système est actif, le nombre d'utilisateurs connectés, et les charges moyennes du système pour la dernière minute, et les 5 et 15 dernières minutes.
>
> Les entrées suivantes sont affichées pour chaque utilisateur : l'identifiant de connexion, le nom du tty, les hôtes distants, la durée de connexion, la durée d'inactivité, JCPU, PCPU, et la ligne de commande de leurs processus.
>
> La durée JCPU est la durée utilisée par tous les processus associés au tty. Cela n'inclut pas les jobs en arrière-plan qui sont terminés, mais seulement les jobs en arrières-plan en cours.
>
> La durée PCPU est celle utilisée par les processus en cours tels que nommés dans le champ "what".

Finalement, depuis un script de shell, la commande `users` est utile pour connaître l'identifiant des utilisateurs connectés. Vous pouvez utiliser le script de shell que voici, par exemple :

```
for user in `users`; do
   <etc>
done
```

12.7.4 Commande `id` et l'UID *effectif*.

La commande `id` affiche vos UID et GID réels et effectifs. Un utilisateur normal possède un UID et un GID. Il peut aussi avoir un UID et un GID effectifs. Un processus est exécuté avec un UID réel correspondant à l'identité de l'utilisateur qui l'a lancé (de même pour le GID). Un second attribut complète cette information : l'UID effectif qui est l'identité que le noyau prend en compte de manière à vérifier les droits d'accès pour les opérations nécessitant une identification (ouverture de fichiers, appels système réservés, etc.).

12.7.5 Limites attribuées aux utilisateurs.

Il existe un fichier `/etc/security/limits.conf` qui répertorie les limites d'usage (par utilisateur) de la CPU, la consommation des processus et d'autres ressources. La documentation de ce fichier de configuration est contenue dans `/usr/[share/]doc/pam-<version>/txts/README.pam_limits`.[1]

[1] NdT : le fichier peut éventuellement se trouver dans `/usr/share/doc/-pam-<version>/modules/README.pam_limits`.

Chapitre 13

Utilisation des services Internet.

Ce chapitre résume la manière de se connecter à distance et les nombreuses méthodes destinées à transférer des fichiers et des données par l'internet.

13.1 `ssh`, pas `telnet` ni `rlogin`.

`telnet` est un programme permettant d'établir une liaison via un service réseau de type Unix. Cett ecommande est très souvent utilisée pour réaliser une connexion à distance. Essayez :

```
telnet <ordinateur_distant>
telnet localhost
```

pour vous connecter sur votre machine à distance. Il n'est pas nécessaire de disposer d'un réseau physique ; les services du réseau fonctionne toujours parce qu'un ordinateur possède toujours un lien interne.

`rlogin` est une version minimale de `telnet` qui permet seulement la connexion à distance. Vous pouvez exécuter les commandes :

```
rlogin -l <nom_utilisateur> <ordinateur_distant>
rlogin -l jack localhost
```

si le système est configuré de manière à permettre des connexions à distance.

Ces deux services sont l'apanage de vieux systèmes Unix. Pour des raisons de sécurité, `ssh` est aujourd'hui préféré pour se connecter à distance :

```
ssh [-l <nom_utilisateur>] <ordinateur_distant>
```

Bien que `rlogin` *et* `telnet` *puissent fonctionner, ces commandes ne devraient jamais être utilisées pour une connexion à travers un réseau public parce qu'alors que vous en faites la saisie, votre mot de passe peut aisément être lu par une tierce personne.*

13.2 rcp et scp.

rcp permet de faire la copie à distance (*remote copy*) de fichiers (ou d'archives). scp en est la version sécurisée associée au paquet ssh. Ces deux commandes autorisent le transfert d'une machine vers une autre tout en utilisant une notation similaire à cp.

```
rcp [-r] [<ordinateur_distant> :]<fichier> [<ordinateur_distant> :]<fichier>
scp [-l <nom_utilisateur>] [-r] [<ordinateur_distant> :]<fichier>
[<ordinateur_distant> :]<fichier>
```

Voici un exemple :

```
[psheer@cericon]# rcp /var/spool/mail/psheer divinian.cranzgot.co.za : \
/home/psheer/mail/cericon
[psheer@cericon]# scp /var/spool/mail/psheer divinian.cranzgot.co.za : \
/home/psheer/mail/cericon
The authenticity of host 'divinian.cranzgot.co.za' can't be established.
RSA key fingerprint is 43 :14 :36 :5d :bf :4f :f3 :ac :19 :08 :5d :4b :70 :4a :7e :6a.
Are you sure you want to continue connecting (yes/no) ? yes
Warning : Permanently added 'divinian.cranzgot.co.za' (RSA) to the list of known hosts.
psheer@divinian's password :
psheer    100% |*************************************|   4266 KB    01 :18
```

L'option -r permet la copie de manière récursive ; les copies peuvent être réalisées dans chaque direction ou même entre deux machines non-locales.

scp devrait toujours être utilisé au lieu de rcp pour des raisons de sécurité. Notez aussi le message d'avertissement retourné par scp lors de la première connexion. Voyez la documentation de ssh pour réaliser votre première connexion de manière sûre. Toutes les commandes du paquet ssh ont le même comportement.

13.3 rsh.

rsh (*shell distant* ou *remote shell*) est un utilitaire très intéressant pour exécuter une commande sur une machine distante. En voici quelques exemples :

```
[psheer@cericon]# rsh divinian.cranzgot.co.za hostname
divinian.cranzgot.co.za
[psheer@cericon]# rsh divinian.cranzgot.co.za \
tar -czf - /home/psheer | dd of=/dev/fd0 bs=1024
tar : Removing leading '/' from members names
20+0 records in
20+0 records out
[psheer@cericon]# cat /var/spool/mail | rsh divinian.cranzgot.co.za \
sh -c 'cat >> /home/psheer/mail/cericon'
```

La première commande retourne le nom de l'hôte sur la machine distante. La seconde sauvegarde mon répertoire personnel situé sur la machine *distante* vers la disquette de la machine *locale* (nous en verrons davantage à propos de dd et de /dev/fd0 plus tard). La dernière commande ajoute mon fichier de messagerie électronique locale au fichier de messagerie sur la machine distante. Notez la manière dont stdin, stdout et stderr sont correctement redirigés vers le terminal local. Après avoir lu le chapitre 30, consultez rsh(8) et in.rshd(8) pour

configurer ce service.

A nouveau, pour des raisons de sécurité, `rsh` *ne devrait jamais être utilisé à travers un réseau public.*

13.4 FTP.

FTP signifie *File Transfer Protocol* (*protocole de transfert de fichiers*). Si FTP est activé sur votre machine locale, alors, d'autres machines peuvent télécharger des fichiers qu'elle contient (*download*). Tapez :

```
ftp metalab.unc.edu
```

ou

```
ncftp metalab.unc.edu
```

`ftp` est le client FTP traditionnel en ligne de commande pour UNIX [le terme "client" indique toujours un programme utilisateur accédant à des services à distance] alors que ncftp est un client plus puissant qui ne sera pas forcément installé par défaut.

Vous devriez à présent être dans une *session* FTP. Un identifiant et un mot de passe vous sont demandés. Le site `metalab.unc.edu` est un des nombreux sites qui autorisent des connexions anonymes. Ceci signifie que vous pouvez taper `anonymous` comme identifiant et laisser un blanc pour mot de passe. Vous noterez aussi que la session vous demande une adresse de courriel comme mot de passe. Toute séquence de lettres incluant un arobase (`@`) devrait suffire, mais il est courtois d'indiquer son adresse de courriel.

La session FTP présente un mini-shell. Vous pouvez utiliser `cd`, `ls` et `ls -l` pour visualiser la liste des fichiers et y naviguer. `help` renvoie une série de commandes et vous pouvez aussi saisir `help <commande>` pour obtenir de l'aide à propos d'une commande donnée. Vous pouvez télécharger (*récupérer* = *download*) un fichier en utilisant la commande `get <nom_de_fichier>`. Cependant, auparavant, vous devez activer soit *tranfer type* soit *binary*. Le terme *transfer type* indique si des caractères de mise à la ligne seront transférés au format DOS. Le fait de taper `ascii` active cette propriété, tandis que `binary` la désactive. Vous souhaiterez peut-être utiliser `hash` qui affichera un dièse (`#`) tous les 1024 octets téléchargés. Ceci est utile pour suivre la progression du téléchargement. Descendez dans le répertoire `README` et entrez ceci :

```
get README
```

Le fichier devrait être téléchargé dans votre répertoire personnel courant.

Vous pouvez aussi changer de répertoire par la commande `cd` et vous rendre dans le repertoire `/incoming`, puis charger (=*upload*) un fichier. Essayez :

```
put README
```

de manière à charger le fichier que vous venez de récupérer par téléchargement.

La plupart des sites FTP ont un répertoire `/incoming` qui est vidé périodiquement.

FTP permet bien plus que de charger des fichiers, quoique l'administrateur du site ait à sa disposition l'option de restreindre l'accès à certaines propriétés. Vous pouvez créer des répertoires, changer les droits, et faire presque tout ce que vous réalisez dans votre système de fichiers usuel.

Si vous avez plusieurs machines sur un réseau *LAN* de confiance [*Local Area Network* , c'est-à-dire soit un réseau privé domestique ou de bureau], toutes ces machines devraient avoir un FTP activé pour permettre aux utilisateurs de copier aisément des fichiers d'ordinateur à ordinateur. La manière d'installer et de configurer un des nombreux serveurs FTP disponibles sera discutée plus tard dans ce livre.

13.5 `finger`.

`finger` est un service pour lister sur une machine distante les utilisateurs qui y connectés. Essayez `finger@<hostname>` pour déterminer qui est connecté sur `<hostname>`. Néanmoins, le service *finger* est très souvent désactivé pour des raisons de sécurité.

13.6 Envoyer des fichiers via le courriel.

Le courriel est de plus en plus utilisé pour transférer des fichiers de machine à machine. Envoyer des messages de plus de 64 kilo-octets sur l'internet est une mauvaise pratique parce que cela tend à charger inutilement les serveurs de messagerie. Tout fichier de plus de 64 kilo-octets devrait être chargé par FTP sur un banal serveur FTP. La plupart des images présentent une taille inférieure à cela, de sorte qu'envoyer une petit image *JPEG* est acceptable [il s'agit d'un format de fichier Internet commun spécialement compressé. Usuellement, ces fichiers font moins de 100 kilo-octets pour une image de la taille d'une photographie normale].

13.6.1 `uuencode` et `uudecode`.

Si vous devez envoyer des fichiers via une messagerie, vous pouvez tirer parti d'`uuencode`. Cet utilitaire empaquette des fichiers binaires dans un format que les serveurs de messagerie traitent aisément. Si vous envoyez un message contenant des données binaires quelles qu'elles soient, il sera plus que probablement corrompu parce que les agents de messagerie sont seulement destinés à manipuler une quantité limitée de caractères. `uuencode` représente un fichier binaire à l'aide de caractères permis, quoique prenant un peu plus d'espace.

Voici une astuce pour empaqueter un répertoire et l'envoyer à quelqu'un par courriel :

```
tar -czf - <mon_repertoire> | uuencode <mon_repertoire>.tar.gz \
| mail -s "Voici quelques fichiers" <utilisateur>@<nom_ordinateur>
```

Pour récupérer le fichier :

```
uudecode <mon_fichier>.uu
```

13.6.2 L'encapsulation MIME.

La plupart des logiciels de messagerie graphiques offrent la possibilité d'attacher des documents aux messages proprement-dits et, inversément, de lire ces fichiers attachés. La méthode ne recourt pas à `uuencode` mais à un format spécial impliqué dans l'*encapsulation MIME*. MIME (*Multipurpose Internet Mail Extension*) est en fait une manière de représenter de multiples fichiers à l'intérieur d'un message unique. La manière dont les données binaires sont manipulées est analogue à celle employées lors de l'appel à `uuencode`, mais le format est connu sous le nom de *base64*.

Chaque document MIME attaché à un courriel possède un type particulier : le *type MIME*. Les types MIME correspondent simplement aux types de documents attachés comme une image, un fichier-son, un fichier-vidéo, un document formaté, etc. Le type MIME est un "tatouage" sous forme littérale avec un format `<majeur>/<mineur>`. La partie `<majeur>` est appelé le type MIME majeur ; *mutatis mutandi* pour la partie `<mineur>`. Les types majeurs correspondent à toutes les sortes de fichiers qui peuvent exister : `application`, `audio`, `video`, `image`, `message`, `texte`. Le type `application` désigne un format de fichier spécial ayant une utilité bien déterminée. Les types MIME mineurs se comptent par centaines. Leur liste se trouve dans `/etc/mime.types`.

Si cela est nécessaire, certains utilitaires en ligne de commande dans la même veine qu'`uuencode` peuvent créer et extraire des messages MIME. C'est le cas de `mpack`, `mumpack` et `mnencode` (ou `mimencode`).

Chapitre 14

Ressources LINUX.

Très souvent, il ne vous sera même pas nécessaire de vous connecter à l'internet pour trouver l'information que vous recherchez. Le chapitre 17 contient une description de la documentation principale des distributions LINUX.

Cependant, il est essentiel d'accéder à l'information la plus récente lorsque le matériel et la sécurité sont concernés. Il est également passionnant et tout-à-fait intéressant d'interagir avec des utilisateurs LINUX à travers le monde. Le développement rapide des logiciels libres vous amènera à penser que vous pourriez passer à côté d'informations importantes circulant à propos des technologie de l'information. En conséquence, la lecture de magazines (éventuellement sur le web), de forums (*newsgroups*) et la souscription à des listes de diffusion (*mailing lists*) sont des actions essentielles de l'administrateur.

14.1 Sites FTP et miroir `sunsite`.

Le site FTP `metalab.unc.be` (naguère appelé `sunsite.unc.edu`) est un des sites traditionnels pour les logiciels libres. Il est reflété dans chaque pays qui possède une infrastructure relative aux technologies de l'information suffisante. Si vous y pointez votre navigateur web, vous trouverez une liste des miroirs se situant dans votre pays. Il est recommandé de naviguer sur ce site. En particulier, vous essayerez de trouver l'emplacement :

- des répertoires où sont stockés tous les paquets logiciels officiels GNU. Ceci est un miroir FTP d'archives de la *Free Software Foundation*. Ce sont les paquets commissionnés par la FSF et pas seulement ceux émis sous la GPL (GNU General Public License). La FSF les distribue sous forme de sources (`.tar.gz`) pour leur inclusion dans diverses distributions. Naturellement, ils peuvent être compilés pour fonctionner sous tout type d'UNIX.
- des répertoires génériques de téléchargement LINUX. Il contient d'innombrables paquets UNIX sous forme de sources mais aussi de binaires, présentés dans une arborescence. Par exemple, les clients de messagerie de courriel possèdent leur propre répertoire. `metalab` est le site où les nouveaux développeurs hébergent tout nouveau logiciel qu'ils produisent. Il y a des instructions sur le site FTP pour y charger des logiciels et faire en sorte que ceux-ci soient adéquatement placés.
- des sources du noyau. C'est un miroir des archives du noyau où Linus et ses

collaborateurs chargent les nouvelles versions *stables* [qui ont été testées de manière telle à ne plus contenir de défauts (*bugs*) sérieux] et *béta* [qui sont en développement] ainsi que les correctifs (*patches*) du noyau.
- des diverses distributions. RedHat, Debian et aussi d'autres distributions populaires y sont peut-être présentes.

Cette liste n'est en rien exhaustive.

Un site FTP vous permet de télécharger des logiciels libres. Souvent, les développeurs hébergent leurs logiciels sur un site web, mais toutes les distributions populaires possèdent presque toujours leur propre site FTP où les versions sont mises en réserve de manière durable. Par exemple, *Cooledit* développé par Paul Sheer est enregistré sur `metalab.unc.edu` dans le répertoire `/pub/Linux/apps/editors/X/cooledit`.

14.2 HTTP et sites web.

La plupart des utilisateurs devraient déjà être familiarisés avec les navigateurs web.[1] Vous devriez aussi devenir un habitué de la recherche sur l'internet [...]. Vous effectuez une recherche sur internet lorsque vous pointez votre navigateur web vers des moteurs de recherche populaires comme :

- `http ://www.google.com/` ,
- `http ://www.google.com/linux`,
- `http ://www.infoseek.go.com/`,
- `http ://www.altavista.com/` ou
- `http ://www.yahoo.com`,

Cette recherche se base sur un mot-clé ou un groupe de mots-clés. La recherche sur le web ressemble un peu à de la magie noire car il faut puiser dans quelques milliards de pages. Consultez toujours les options de l'assistant de recherches avancées pour déterminer comment faire des recherches plus complexes que la simple interrogation par mots.

Les sites web dans la FAQ (foire aux questions ou *frequently asked questions*) devraient tous être consultés de manière à obtenir une vue d'ensemble des sites d'intérêt à propos de LINUX.

Il est très important que vous considériez les dernières nouvelles à propos de LINUX :
- *Linux Weekly News* `http ://www.lwn.net/` constitue un excellente source d'informations.
- De la même manière, le célèbre *SlashDot* `http ://slashdot.org` fournit des mises à jour quotidiennes à propos "des choses qui comptent" (et donc, ce site contient de nombreuses informations relatives aux logiciels libres).
- *FreshMeat* `http ://freshmeat.net` est un site internet dévolu à la sortie de nouveaux logiciels. Vous y trouverez de nouveaux paquets et/ou des paquets mis-à-jour. Les annonces ont lieu toutes les deux heures.
- *Linux Planet* `http ://www.linuxplanet.com/` est un site plus récent contenant de nombreuses informations tutorielles pour LINUX.

[1] Alors que `seamonkey` (`mozilla` anciennement), `firefox`, `galeon`, `epiphany`, etc. sont des navigateurs graphiques, `lynx` et `links` sont des navigateurs fonctionnant en console.

- *NewsForge* http ://www.newsforge.net/ contient également des informations quotidennes à propos de la sortie de nouveaux logiciels.
- Lycos http ://download.lycos.com/static/adavanced_search.asp est un moteur de recherche FTP efficace pour localiser des paquets logiciels. C'est un des rares moteurs à comprendre les expressions rationnelles.[2]

Il est à noter qu'un nouveau site consacré à LINUX est créé toutes les semaines. Tout ce qui se rapporte à LINUX fait probablement déjà partie d'un site ou l'autre.

14.3 SourceForge.

Le site web SourceForge est apparu plus récemment dans la communauté du libre, http ://sourceforge.net/. Les développeurs peuvent utiliser ce service gratuit pour héberger le site web de leur projet, des archives FTP et les listes de diffusion. SourceForge s'est developpé de manière si spectaculaire et rapide qu'il héberge la meilleur moitié de tous les projets libres.

14.4 Listes de diffusion.

Une liste de diffusion est une adresse spéciale qui envoye automatiquement le message électronique que vous lui adressez, à une longue liste d'autres adresses. Usuellement, vous souscrivez à une liste de diffusion en envoyant un message spécialement formaté ou en demandant une inscription via le gestionnaire de la liste.

Une fois l'enregistrement effectué, tout message électronique que vous posterez à l'attention de la liste sera envoyé aux autres membres et, tout message venant d'un membre sera répercuté sur toutes les adresses des inscrits.

Il y a trois grands types de listes de diffusion : *majordomo*, *listserv* et **-request*.

14.4.1 Majordomo et Listserv.

Pour s'inscrire sur une liste *majordomo*, il faut envoyer le message à majordomo@<machine> sans sujet, le message contenant une ligne telle que :

```
subscribe <mailing-list-name>
```

Cette commande ajoute votre nom à la liste de diffusion intitulée <mailing-list-name@<machine>. C'est à cette liste que vos messages aboutiront.

Vous ferez la même chose dans le cas des listes listserv, en envoyant le message à list-serv@<machine>.

Par exemple, si vous êtes l'administrateur de machines exposées à l'internet, vous devriez adhérer à bugtraq. Envoyez ce message :

```
subscribe bugtraq
```

à listserv@netspace.org. Dès lors, devenez un des usagers parmi les dizaines

[2] NdT : lors de la traduction, le site était introuvable.

de milliers de ceux qui lisent et rapportent des problèmes de sécurité relatifs à LINUX.

Pour une désinscription, il convient d'envoyer le message :

```
unsubscribe <mailing-list-name>
```

N'envoyez jamais les messages `subscribe` ou `unsubscribe` à la liste de diffusion elle-même. Envoyez les messages `subscribe` ou `unsubscribe` exclusivement à l'adresse `majordomo@<machine>` ou à `listserv@<machine>`.

14.4.2 *-request.

Vous souscrivez à ce type de listes en envoyant un message électronique blanc à `<mailing-list-name>-request@<machine>` avec le mot `subscribe` comme sujet. La désinscription s'effectue de la même manière mais en utilisant `unsubscribe` pour sujet.

A nouveau, n'envoyez jamais les messages `subscribe` ou `unsubscribe` à la liste de diffusion elle-même.

14.5 Forums.

Un forum de discussion (newsgroup) est un espace auquel tout le monde a accès. Il y a des dizaines de milliers de forums de discussion et chacun est unique au monde.

Le logiciel client que vous utilisez pour lire un forum de discussion est appelé un *lecteur de nouvelles* (*news reader* ou *news client* en anglais). `rtin` est un lecteur en mode console tandis que `netscape` fonctionne en mode graphique. `pan` constitue un excellent lecteur en mode graphique.

Les forums de discussion ont le même nom que les hôtes internet. Vous pourriez être intéressé par `comp.os.linux.announce`. Le terme `comp` est le descripteur de sujet pour *computers*; `os` vaut pour *operating system* et ainsi de suite. De nombreux autres forums `linux` traitent de sujets relatifs au système LINUX. Les serveurs de forums de discussion dévorent énormément d'information. Ils forment une structure arborescente sur l'internet. Lorsque vous envoyez un message sur un forum, cela prend environ une journée pour que l'information se propage sur les serveurs du monde entier. Inversément, vous pouvez consulter une liste de tous les messages postés sur chaque forum par n'importe qui se trouvant n'importe où.

Quelle est la différence entre un forum de discussions et une liste de diffusion ? L'avantage d'un forum est qu'il vous est loisible de ne charger que les messages qui vous intéressent. Dans le cas d'une liste, vous téléchargez obligatoirement toute l'information. Dans le cas d'un forum, vous analysez la liste des messages et ne captez que ceux qui vous intéressent.

Pourquoi ne pas mettre une liste de diffusion sur une page web ? Si vous faisiez cela, toute personne dans le monde intéressée par la liste devrait réaliser des recherches via des liens internationaux pour obtenir la page web. Ceci chargerait le serveur proportionnellement au nombre des membres ayant souscrit à la liste. C'est ce que fait SlashDot. Cependant, votre serveur de forum de discussions est local, si bien que vous récupérez les messages via un lien rapide et

sans encombrer le trafic sur l'internet.

14.6 Les RFCs.

Les RFCs (*Request For Comments*) constituent une source indispensable d'information pour les administrateurs et les développeurs sérieux. Les RFCs sont des références sur l'internet, écrites par des autorités qui définissent chaque aspect des communications internet. Très souvent, la documentation se rapporte aux RFCs. [Il y a quelques RFCs qui n'ont pas d'intérêt. [...]. Voir `http ://slashdot.org`].

`ftp ://metalab.unc.edu/pub/docs/rfc/` (et ses miroirs) présente une archive complète des RFCs prêtes au téléchargement. Il y en a environ 2.500. Le fichier d'index `rfc-index.txt` sera probablement votre point de départ. Ses entrées se présentent ainsi :

```
2045 Multipurpose Internet Mail Extensions (MIME) Part One : Format of
     Internet Message Bodies. N. Freed & N. Borenstein. November 1996.
     (Format : TXT=72932 bytes) (Obsoletes RFC1521, RFC1522, RFC1590)
     (Update by RFC2184, RFC2231) (Status : DRAFT STANDARD)

2045 Multipurpose Internet Mail Extensions (MIME) Part Two : Media
     Types. N. Freed & N. Borenstein. November 1996. (Format : TXT=105854
     bytes) (Obsoletes RFC1521, RFC1522, RFC1590) (Status : DRAFT STANDARD)
```

et aussi sous cette forme :

```
2068 Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding,
     J. Gettys, J. Mogul, H. Frystyk, T. Berbers-Lee. January 1997 . (Format :
     TXT=378114 bytes) (Status : PROPOSED STANDARD)
```

A présent, vous devriez avoir une assez bonne idée de ce que sont les RFCs.

Chapitre 15

Droits et les trois types de temps.

Chaque fichier et répertoire d'un système UNIX, en plus d'être la propriété d'un utilisateur et d'un groupe, est caractérisé par des droits d'accès [activés ou non] appelés en anglais *flags* ou *access bits*. Ceux-ci déterminent le type d'accès à un fichier qu'un utilisateur ou un groupe peut avoir.

L'exécution de la commande `ls -ald /bin/cp /etc/passwd /tmp` vous renvoie une liste comme celle-ci :

```
-rwxr-xr-x    1  root    root    28628 Mar 24  1999  /bin/cp
-rw-r--r- -   1  root    root     1151 Jul 23 22 :42  /etc/passwd
drwxrwxrwt    5  root    root     4096 Sep 25 15 :23  /tmp
```

Dans le premier bloc à gauche, sont décrits les droits d'accès. Jusqu'ici, nous avons vu que le premier indicateur pouvait être `-` ou `d`, c'est-à-dire respectivement un fichier ordinaire et un répertoire. Les neuf autres indicateurs présentent un `-` pour droits d'accès désactivés ou divers caractères (r, w, x, s, S, t, ou T). Le tableau 15.1 détaille la description complète des droits sur les fichiers et répertoires.

15.1 La commande `chmod`.

La commande `chmod` permet de modifier les droits d'un fichier. Sa syntaxe se présente comme suit :

```
chmod [-R][u|g|o|a][+|-] [r|w|x|s|t] <fichier> [<fichier>] ...
```

Par exemple,

```
chmod u+x mon_fichier
```

ajoute le droit d'exécution à `mon_fichier` en ce qui concerne l'utilisateur. Par ailleurs,

TAB. 15.1 – Droits d'accès aux fichiers et répertoires.

	Caratères (- =non-fixé)	Effets sur les répertoires	Effets sur les fichiers
Propriétaire, u (users)	r	L'utilisateur peut lire le contenu du répertoire.	L'utilisateur peut lire le fichier.
	w	Avec w ou s, l'utilisateur peut créer ou supprimer le répertoire.	L'utilisateur peut écrire dans le fichier.
	x s S	L'utilisateur peut accéder aux fichiers du répertoire pour x ou s. S n'a pas d'effet.	L'utilisateur peut exécuter le fichier pour x ou s. s -le bit *setuid*- affecte l'ID du propriétaire à l'ID de l'utilisateur pour le fichier et tout processus subsequent. S n'a pas d'effet
Groupe, g (group)	r	Le groupe peut lire le contenu du répertoire.	Le groupe peut lire le fichier.
	w	Avec w ou s, le groupe peut créer et détruire des fichiers du répertoire.	Le groupe peut écrire dans le fichier.
	x s S	Avec x, le groupe peut accéder au contenu du répertoire. Avec s, tous les fichiers créés possèdent le même groupe que le répertoire. S n'a pas d'effet .	Le groupe peut exécuté le fichier pour x ou s. s -le bit *setgid*- affecte d'ID du propriétaire du fichier à l'ID du groupe pour le fichier et tout processus subsequent. S n'a pas d'effet.
Autres, o (others)	r	Tout le monde peut lire le contenu du répertoire.	Tout le monde peut lire le fichier.
	w	Avec w ou t, tout le monde peut créer ou détruire un fichier dans le répertoire	Tout le monde peut écrire dans le fichier
	x t T	Avec x ou t, tout le monde peut accéder au contenu de tous les fichiers du répertoire. t -le bit *sticky*- empêche les utilisateurs d'ôter (détruire) des fichiers qu'ils ne possèdent pas. Donc, les utilisateurs sont libres d'ajouter de nouveaux fichiers mais pas d'enlever des fichiers appartenant à d'autres. T n'a pas d'effet.	Avec x ou t, le groupe peut exécuter le fichier. Avec t, le code du programme est placé automatiquement en mémoire de sorte que les chargements ultérieurs sont accélérés. T n'a pas d'effet.

```
chmod a-rx mon_fichier
```

ôte les droits de *lecture* (*read*) et d'*exécution* (*execute*) pour *tous* (*all*), c'est-à-dire l'utilisateur, le groupe et les autres. L'option `-R` signifie *récursivement*.

Les droits d'accès sont souvent représentés sous leur forme binaire, en particulier dans les programmes. Il est donc utile de voir comment `rwxrwxrwx` se traduit en mode octal [voir la section 3.1]. Les fichiers sont usuellement créés sur un ordinateur selon le *mode* `0644`, ce qui signifie `rw-r- -r- -`.[1] Vous fixez les droits de manière explicite en mode octal de cette manière :

```
chmod 0755 mon_fichier
```

ce qui donne à `mon_fichier` les droits `rwxrw-rw-`. Pour une liste complète des valeurs octales des droits et des types de fichiers, référez-vous à `/usr/include/linux/stat.h`.

Dans le tableau 15.1, vous pouvez lire `s`, le bit *setuid* ou *setgid*. S'il est utilisé sans exécuter de droits, cet indicateur n'a pas de signification et il est écrit comme `S`. Cet indicateur modifie un `x` en `s` de sorte que vous devriez lire un `s` pour "exécuter avec le bit *setuid* (ou *setgid* selon la position)". `t` est le bit *sticky*. Il n'a pas de signification s'il n'y a pas de droits d'exécution et est alors écrit comme `T`.

Le chiffre `0` peut être négligé mais il est souhaitable de l'indiquer pour expliciter complètement les droits. Ce chiffre *peut* prendre une valeur associée aux bits setuid (4), setgid (2) et sticky (1). Ainsi, une valeur de `5764` équivaut à `-rwsrw-r-T`.[2]

15.2 La commande `umask`.

`umask` fixe les droits par défaut pour les fichiers nouvellement créés ; sa valeur usuelle est `022`. Cette valeur par défaut signifie que les droits de tout nouveau fichier (créé avec la commande `touch`, disons) seront masqués par ce nombre.[3] Le masque `022` exlut les permissions d'écriture pour le groupe (`g`) et les autres (`o`). Un `umask` de `026` excluerait les droits de lecture et l'écriture pour les autres (`o`) mais permettrait au groupe (`g`) la lecture et l'écriture. Essayez :

---

[1] NdT : Pour l'utilisateur (u), rwx correspond à 400 + 200 + 100 (=700). Pour le groupe (g), rwx correspond à 40 + 20 + 10 (=70). Pour les autres (o), rwx correspond à 4 + 2 + 1 (=7). La combinaison rw-r- -r- - vaut 400 + 200 + 0 + 40 + 0 + 0 + 4 + 0 + 0 (=644). L'origine du chiffre 0 précédant ce nombre dans l'exemple du texte est expliqué à la fin du paragraphe 15.1.

[2] NdT : Dans ce cas, 764 signifie donc -rwxrw-r- -. Etant donné que le droit d'exécution (x) associé à l'utilisateur est converti en bit setuid et que le dernier bit est converti en bit sticky, on compte 4000 + 0 + 1000 (=5000) auquel on ajoute 764 (=5764). Etant donné que le bit sticky s'applique à un droit désactivé, il passe en majuscule (T).

[3] NdT : Dans le cas d'un fichier exécutable, les droits seront fixés comme `777-022=755`, c'est-à-dire `-rwxr-xr-x`. Dans le cas d'un fichier non-exécutable (texte, image, etc.), les droits seront `666-022=644` soit `-rw-r- -r- -`. C'est du moins ce qui se produit sur la distribution Gentoo du traducteur.

```
umask
touch <fichier1>
ls -al <fichier1>
umask 026
touch <fichier2>
ls -al <fichier2>
```

026 est probablement plus proche du type de masque employs dans le cas des utilisateurs ordinaires. Vérifiez votre fichier `/etc/profile` pour déterminer quelle valeur d'`umask` est associé à votre identifiant, et aussi pourquoi.

15.3 `stat` et les trois types de temps.

En plus des droits, chaque fichier est caractérisé par trois entiers qui représentent, en secondes, le moment du dernier accès en lecture (access time), le moment de la dernière modification (modification time) et le moment du dernier changement en termes de droits (change time). Ces moments sont définis respectivement comme *atime*, *mtime* et *ctime*.

Pour obtenir une liste complète des droits d'un fichier, utilisez la commande `stat`. Voici, à titre d'exemple, le résultat de la commande `stat /etc` :

```
  File : "/etc"
  Size : 4096          Filetype : Directory
  Mode : (0755/drwxr-xr-x)        Uid : (  0/  root)  Gid  (  0/  root)
Device : 3,1     Inode : 14057    Links : 41
Access :  Sat Sep 25 04 :09 :08 1999(00000.15 :02 :23)
Modify :  Fri Sep 24 20 :55 :14 1999(00000.22 :16 :17)
Change : Fri Sep 24 20 :55 :14 1999(00000.22 :16 :17)
```

Le terme `Size :` indiqué ici est la quantité d'espace disque réellement occupée pour stocker le répertoire de *listing*, et cette dernière est la même que celle retournée par la commande `ls`. Dans le cas présent, il s'agit probablement de quatre blocs du disque valant chacun 1024 octets. La taille ainsi mentionnée n'est pas la somme des tailles des fichiers composant le répertoire.[4] En revanche, pour un fichier, `Size :` donnera exactement la taille du fichier en octets telle que retournée par la commande `ls`.

[4] NdT : pour obtenir cette valeur, utilisez la commande `du`.

Chapitre 16

Liens symboliques et physiques.

Très souvent, un fichier doit se trouver dans deux répertoires. Considérez, par exemple, un fichier de configuration requis par deux paquets logiciels différents recherchant ce fichier dans deux répertoires différents. Le fichier pourrait être tout simplement recopié. Toutefois, lors de modifications, vous devrez reproduire ces dernières en plusieurs endroits de votre système, ce qui deviendra rapidement un cauchemar en termes d'administration. L'idéal serait d'avoir un fichier présent dans plusieurs répertoires, mais modifiable en un seul endroit. *Les liens constituent une méthode élégante permettant à deux (ou plusieurs) fichiers de contenir exactement les mêmes données.*

16.1 Liens symboliques.

Pour montrer ce qu'est un lien symbolique, essayez les commandes suivantes :

```
touch mon_fichier
ln -s mon_fichier lien_fichier
ls -al
cat > mon_fichier
quelques
lignes
de texte
^D
cat mon_fichier
cat lien_fichier
```

Notez bien que le résultat de la commande `ls -al` montre qu'il y a une lettre `l` tout à gauche de la ligne correspondant à `lien_fichier` et qu'un tiret (`-`) caractérise le fichier `mon_fichier`. Ceci indique que `lien_fichier` est un *lien symbolique* (*symbolic link*, *symlink* ou *soft-link*) associé à un autre fichier.

Un lien symbolique ne contient pas de données en soi car c'est une référence à un autre fichier. Il peut même contenir une référence à un répertoire. Dans les deux cas, les programmes agissant sur le lien symbolique manipuleront le fichier

ou le répertoire vers lequel le lien pointe.

Essayez à présent :

```
mkdir repertoire
ln -s repertoire lien_repertoire
ls -al .
touch ./repertoire/fichier1
touch ./repertoire/fichier2
ls -al ./repertoire
ls -al ./lien_repertoire
```

Le répertoire `lien_repertoire` est un lien symbolique pointant vers `repertoire` et apparaît comme une réplique de ce dernier. A nouveau, `lien_repertoire` ne consomme pas d'espace mémoire supplémentaire. Un programme lisant le lien n'est pas "conscient" qu'il lit un répertoire différent de `repertoire`.

Les liens symboliques peuvent aussi être copiés et retenir leur valeur :

```
cp lien_repertoire /
ls -al /
cd /lien_repertoire
```

Vous avez copié le lien dans le répertoire racine `/`. Cependant, ce lien pointe selon un chemin relatif vers `repertoire` (qui est dans le même répertoire que le lien). Or, `repertoire` ne se trouve pas dans `/`, d'où l'émission d'une erreur.

Essayez maintenant :

```
rm -f repertoire /lien_repertoire
ln -s `pwd`/repertoire lien_repertoire
ls -al
```

A présent, vous pouvez observer que `lien_repertoire` possède un chemin absolu. Vous pouvez exécuter :

```
cp lien_repertoire /
ls -al /
cd /lien_repertoire
```

et désormais, cela fonctionne.

Un des usages courants des liens symboliques est de permettre aux systèmes de fichiers d'être montés (`mount`*ed*) à partir d'un répertoire différent (voir la section 20.4). Par exemple, vous pourriez disposer d'un répertoire de grande taille devant être réparti sur plusieurs disques physiques. Par souci de clarté, vous pouvez monter les disques comme `/disk1`, `/disk2`, etc., et ensuite lier les différents sous-répertoires de manière à disposer de tout l'espace nécessaire.

Un autre exemple est le lien de `/dev/cdrom` à `/dev/hdc` (disons), si bien que les programmes accédant au fichier du périphérique `/dev/cdrom` (voir le chapitre 19) accèdent en réalité au disque IDE.

16.2 Liens physiques.

UNIX permet aux données d'un fichier d'avoir plus d'un nom en différents endroits du système de fichiers. Un tel fichier ayant donc plusieurs noms pour les mêmes données est appelé un lien physique (*hard-link* en anglais). Essayez :

```
touch mes_donnees
ln mes_donnees mes_donnees2
ls -al
```

Les fichiers `mes_donnees` et `mes_donnees2` sont indiscernables. Ils partagent les mêmes données et présentent un `2` dans la seconde colonne de la sortie de `ls -al`. Ceci signifie qu'ils sont liés "en dur" *deux fois* (c'est-à-dire qu'il y a deux noms pour le fichier).

Les liens physiques sont parfois préférés aux liens symboliques pour éviter que certains programmes ne soient trompés par les liens symboliques. En effet, si un de vos scripts (par exemple) utilise la commande `cp` pour copier un fichier, ce script copiera le lien symbolique au lieu du fichier vers lequel pointe ce dernier [cp possède une option qui contourne ce problème, voir `cp`(1)]. Un lien physique sera cependant toujours perçu comme un fichier réel.

En contrepartie, les liens physiques ne peuvent pas être établis entre des fichiers appartenant à différents systèmes de fichiers, ni entre répertoires.

Chapitre 17

Documentation pré-installée.

Ce chapitre décrit où se trouve la documentation associée à une distribution LINUX usuelle. Les chemins correspondent à une distribution RedHat. Ils sont peut-être un peu différents pour d'autres distributions. Une différence consiste en la migration des sources de documentation de `/usr/????` vers `/usr/share/????` (leur emplacement adéquat vu qu'elles peuvent être partagées par différentes machines). Dans beaucoup de cas, la documentation n'est pas installée ou se trouve dans des répertoires complètement différents. Malheureusement, il est difficile de suivre l'évolution des quelque 20 éditeurs principaux.

Dans le cas de nombreux systèmes propriétaires, la documentation relative au système se présente sous la forme de textes imprimés. Pour LINUX, la majorité de l'information est écrite par les auteurs eux-mêmes et est jointe au code source. Une distribution typique LINUX associera la documentation avec les binaires compilés. Les distributions communes contiennent des centaines de méga-octets de documentation sous forme de texte tel quel, hyperlié ou imprimable. Souvent, il n'est pas nécessaire de visiter le World Wide Web, sauf lorsqu'une information est obsolète. Si ce n'est déjà fait, essayez ceci :

```
ls -ld /usr/*/doc /usr/*/*/doc /usr/share/*/*/doc /opt/*/doc  \
     /opt/*/*/doc
```

Ce n'est là qu'une méthode approximative de recherche des répertoires de documentation, mais elle donne au moins la liste des répertoires d'une RedHat 7.0 officielle avec un jeu complet des paquets installés :

```
/usr/X11R6/doc                  /usr/share/vim/vim57/doc
/usr/lib/X11/doc                /usr/share/doc
/usr/local/doc                  /usr/share/gphoto/doc
/usr/share/texmf/doc            /usr/share/lout/doc
```

17.1 Documentation du noyau :

/usr/src/linux/Documentation

Ce répertoire contient l'information sur tous les pilotes matériels sauf les cartes graphiques. Le noyau possède des pilotes intégrés pour les cartes réseau,

les controleurs SCSI, les cartes son, etc. Si vous devez déterminer qu'un de ces pilotes est supporté, /usr/src/linux/Documentation est le premier répertoire à visiter.

17.2 Support matériel graphique de X-Window :

/usr/X11R6/lib/X11/doc/

Dans ce répertoire (identique à /usr/X11R6/doc/), vous trouverez la documentation sur tous les matériels graphiques supportés par **X**, comment configurer **X**, régler les modes vidéos, faire face à des cartes graphiques incompatibles, etc. Consultez la section 44.5 pour plus de détails.

17.3 TEX et référence de méta-polices :

/usr/share/texmf/doc/

Ce répertoire contient une référence très volumineuse et approfondie du langage de composition de textes TEX et le paquet de production de polices MetaFont. Cependant, ce n'est pas une référence exhaustive.

17.4 Documentation de LATEX-HTML :

/usr/share/texmf/doc/latex/latex2e-html/

Ce répertoire contient une référence détaillée du langage typographique LATEX.

17.5 Les HOWTO :

/usr/doc/HOWTO ou /usr/share/doc/HOWTO

Les HOWTO sont une excellente source de tutoriaux pour le débutant, qui permettent d'installer et d'utiliser touts les services imaginables. RedHat semble ne plus fournir cette documentation avec les paquets de base. Il est intéressant de publier la liste de ces HOWTO pour illustrer la diversité des sujets. Ceux-ci sont réfléchis via des sites-miroirs sur l'internet de sorte qu'il n'y a pas de difficultés à les obtenir à l'aide d'un moteur de recherche (en particulier via http ://www.tldp.org). Voir le tableau 17.1.

17.6 Les Mini-HOWTO :

/usr/doc/HOWTO/mini ou /usr/share/doc/HOWTO/mini

Il y a des tutoriaux de plus petite taille pour une initiation rapide (voir http ://www.tldp.org) (tableau 17.2).

TAB. 17.1 – Liste des HOWTOs.

3Dfx-HOWTO	Finnish-HOWTO	Modem-HOWTO	Security-HOWTO
AX25-HOWTO	Firewall-HOWTO	Multi-Disk-HOWTO	Serial-HOWTO
Access-HOWTO	French-HOWTO	Multicast-HOWTO	Serial-Programming-HOWTO
Alpha-HOWTO	Ftape-HOWTO	NET-3-HOWTO	Shadow-Password-HOWTO
Assembly-HOWTO	GCC-HOWTO	NFS-HOWTO	Slovenian-HOWTO
Bash-Prompt-HOWTO	German-HOWTO	NIS-HOWTO	Software-Release-Pratice-HOWTO
Benchmarking-HOWTO	Glibc2-HOWTO	Networking-Overview-HOWTO	Sound-HOWTO
Beowulf-HOWTO	HAM-HOWTO	Optical-Disk-HOWTO	Sound-Playing-HOWTO
BootPrompt-HOWTO	Hardware-HOWTO	Oracle-HOWTO	Spanish-HOWTO
Bootdisk-HOWTO	Hebrew-HOWTO	PCI-HOWTO	TeTEX-HOWTO
Busmouse-HOWTO	INDEX.html	PCMCIA-HOWTO	Text-Terminal-HOWTO
CD-Writing-HOWTO	INFO-sheet	PPP-HOWTO	Thai-HOWTO
CDROM-HOWTO	IPCHAINS-HOWTO	PalmOS-HOWTO	Tips-HOWTO
COPYRIGHT	IPX-HOWTO	Parallel-Processing-HOWTO	UMSDOS-HOWTO
Chinese-HOWTO	IR-HOWTO	Pilot-HOWTO	UPS-HOWTO
Commercial-HOWTO	ISP-Hookup-HOWTO	Plug-and-Play-HOWTO	UUCP-HOWTO
Config-HOWTO	Installation-HOWTO	Polish-HOWTO	Unix-Internet-Fundamentals-HOWTO
Consultants-HOWTO	Intranet-Service-HOWTO	Portugese-HOWTO	User-Group-HOWTO
Cyrillic-HOWTO	Italian-HOWTO	PostgreSQL-HOWTO	VAR-HOWTO
DNS-HOWTO	Java-CGI-Howto	Printing-HOWTO	VME-HOWTO
Dos-Win-to-Linux-HOWTO	Kernel-HOWTO	Printing-Usage-HOWTO	VMS-to-Linux-HOWTO
Dos-to-Linux-HOWTO	Keyboard-and-Console-HOWTO	Quake-HOWTO	Virtual-Services-HOWTO
DOSEMU-HOWTO	KickStart-HOWTO	README	WWW-HOWTO
Danish-HOWTO	LinuxDoc+Emacs+Ispell-HOWTO	RPM-HOWTO	WWW-mSQL-HOWTO
Distribution-HOWTO	META-FAQ	Reading-List-HOWTO	XFree86-HOWTO
ELF-HOWTO	MGR-HOWTO	Root-RAID-HOWTO	XFree86-Video-Timings-HOWTO
Emacspeak-HOWTO	MILO-HOWTO	SCSI-Programming-HOWTO-	XWindow-User-HOWTO
Esperanto-HOWTO	MIPS-HOWTO	SMB-HOWTO	
Ethernet-HOWTO	Mail-HOWTO	SRM-HOWTO	

TAB. 17.2 – Liste des mini-HOWTOs

3-Button-Mouse	DHCPcd	Leased-Line	Offline-Mailing	Software-Buiding
ADSL	DPT-Hardware-RAID	Linux+DOS+Win95+OS2	PLIP	Software-RAID
ADSM-Backup	Diald	Linux+FreeBSD	Partition	Soundblaster-AWE
AI-Alife	Diskless	Linux+FreeBSD-mini-HOWTO	Partition-Rescue	StarOffice
Advocacy	Ext2fs-Undeletion	Linux+NT-Loader	Path	Term-Firewall
Alsa-sound	Fax-Server	Linux+Win95	Pre-Installation-Checklist	TkRat
Apache+SSL+PHP+fp	Firewall-Piercing	Loadlin+Win95	Process-Accounting	Token-Ring
Automount	GIS-GRASS	Loopback-Root-FS	Proxy-ARP-Subnet	Ultra-DMA
Backup-With-MSDOS	GTEK-BBS-550	Mac-Terminal	Public-Web-Browser	Update
Battery-Powered	Hard-Disk-Upgrade	Mail-Queue	Qmail+MH	Upgrade
Boca	INDEX	Mail2News	Quota	VAIO+Linux
BogoMips	INDEX.html	Man-Page	RCS	VPN
Bridge	IO-Port-Programming	Modules	README	Vesafb
Bridge+Firewall	IP-Alias	Multiboot-with-LILO	RPM+Slackware	Visual-Bell
Bzip2	IP-Masquerade	NCD-X-Terminal	RedHat-CD	Windows-Modem-Sharing
Cable-Modem	IP-Subnetworking	NFS-Root	Remote-Boot	WorldPerfect
Cipe+Masq	ISP-Connectivity	NFS-Root-Client	Remote-X-Apps	X-Big-Cursor
Clock	Install-from-ZIP	Netrom-Node	SLIP-PPP-Emulator	XFree86-XInside
Coffee	Kerneld	Netscape+Proxy	Secure-POP+SSH	Xterm-Title
Colour-ls	LBX	Netstation	Sendmail+UUCP	ZIP-Drive
Cyrus-IMAP	LILO	News-Leafsite	Sendmail-Address-Rewrite	ZIP-Install
DHCP	Large-Disk	Offline	Small-Memory	

17.7 Projet de documentation Linux :

/usr/doc/LDP ou /usr/share/doc/LDP

La page principale du projet LDP (*Linux Documentation Project*) est : http ://www.tldp.org. Ce projet est une consolidation des HOWTOs, des foires aux questions (FAQs), des pages de man, etc. Il n'y a rien de ce qui est installé sur votre système qui ne soit abordé sur ce site.

17.8 Documentation web :

/home/httpd/html ou /var/www/html

Certains paquets peuvent installer de la documentation dans ce répertoire de manière à ce que cette dernière soit en ligne automatiquement si votre serveur web est actif (dans les anciennes distributions, ce répertoire était /home/httpd/html).

17.9 Référence à propos d'Apache :

/home/httpd/html/manual ou /var/www/html/manual

Apache maintient en ligne ce matériel de référence, de sorte que ceci est la page par défaut apparaissant lorsque vous installez Apache pour la première fois. Apache est le serveur web le plus populaire.

17.10 Pages de manuel :

/usr/man/ ou /usr/share/man/

Les pages du manuel sont discutées dans la section 5.7. Les autres répertoires de la superstructure (voir la section 18.2) peuvent contenir des pages de man. Sur certains autres systèmes Unix, ces pages sont hébergées ailleurs.

Pour convertir une page de man en Postscript (en vue d'une impression ou d'une visualisation), utilisez ceci (qui est un exemple se rapportant à la page de man de cp)[1] :

```
groff -Tps -mandoc /usr/man/man1/cp.1 > cp.ps ; gv cp.ps
groff -Tps -mandoc /usr/share/man/man1/cp.1 > cp.ps ; gv cp.ps
```

17.11 Pages d'info :

/usr/info/ ou /usr/share/info/

Les pages d'info sont présentées à la section 5.8.

[1] NdT : il est également possible d'exécuter groffer /usr/share/man/man1/cp.1.gz > $HOME/cp.ps

TAB. 17.3 – Documentation de logiciels, située dans /usr/[share/]doc.

ImageMagick-5.2.2	gcc-c++-2.96	libtool-1.3.5	pmake-2.1.34
LPRng-3.6.24	ghostscript-5.50	libxml-1.8.9	pygtk-0.6.6
XFree86-doc-4.0.1	gimp-1.1.25	lilo-21.4.4	python-docs-1.5.2
bash-2.04	glibc-2.1.92	lsof-4.47	rxtv-22.6.3
bind-8.2.2_P5	gtk+-1.2.8	lynx-2.8.4	sane-1.0.3
cdrecord-1.9	gtk+-devel-1.2.8	ncurses-devel-5.1	sgml-tools-1.0.9
cvs-1.10.8	ipchains-1.3.9	nfs-utils-0.1.9.1	slang-devel-1.4.1
fetchmail-5.5.0	iproute-2.2.4	openjade-1.3	stylesheets-1.54.13rh
freetype-1.3.1	isdn4k-utils-3.1	openssl-0.9.5a	tin-1.4.4
gawk-3.0.6	krb5-devel-3.5.5	pam-0.72	uucp-1.06.1
gcc-2.96	libtiff-devel-3.5.5	pine-4.21	vim-common-5.7

17.12 Documentation de paquets individuels : /usr/doc/* ou /usr/share/doc/*

Enfin, tous les paquets installés sur le système ont leur propre répertoire de documentation individuel. Un paquet foo présentera très probablement un répertoire de documentation /usr/doc/foo (ou /usr/share/doc/foo). Ce répertoire contiendra la documentation fournie avec les sources du logiciel, tel qu'une information de version, des mises-à-jour à propos des caractéristiques, du code à titre d'exemple ou une FAQ. Si vous êtes intéressé par un paquet donné, vous devriez toujours consulter son répertoire dans /usr/doc (ou /usr/share/doc), ou –mieux encore– télécharger les sources.

Dans le tableau 17.3, vous trouverez les répertoires contenus dans /usr/doc (ou /usr/share/doc) qui constituent plus qu'une simple documentation sur les paquets. Dans certains cas, les paquets présentent une documentation tout-à-fait complète. A titre d'exemple, les références de Python ne sont reprises nulle part ailleurs.

Chapitre 18

Survol de la structure des répertoires UNIX.

Ce chapitre décrit succintement la structure d'UNIX. Il s'agit d'un survol à caractère théorique et non d'une description détaillée du système de fichiers LINUX. Le chapitre 36 contient une analyse détaillée des répertoires autorisés ainsi que du type de fichiers pouvant y être accueillis.

18.1 Paquets.

Les systèmes LINUX sont construits à l'aide de centaines de petits paquets logiciels, chacun d'eux réalisant des groupes d'opérations logiques. Sur LINUX, des paquets autonomes intéragissent de manière à produire plus de fonctionnalités que ne le feraient de grands ensembles logiciels unitaires. Il n'y a pas de distinction nette entre les parties propres du système d'exploitation et les applications ; chaque fonction est un paquet.

Un paquet logiciel (*software package*) sur un système RedHat typique est distribué sous forme d'un fichier *RPM* (pour *RedHat Package Manager*) qui présente une extension `.rpm`. Dans le cas d'une distribution *Debian*, un fichier équivalent est caractérisé par une extension `.deb` ; pour une *Slackware*, l'extension est `.tgz`. Les paquets d'une *Gentoo* ont un suffixe `.ebuild` qui permet au sein de `Portage` de compiler les sources sous forme `.tar.gz` ou `.tar.bz2` en vérifiant les dépendances.[1]

Lors de son installation, chaque paquet est developpé en plusieurs fichiers distribués dans tout le système. Les paquets ne créent généralement pas de répertoires majeurs mais au lieu de cela, les fichiers qui le constituent sont disposés dans des répertoires existant clairement identifiés.

18.2 Superstructure des répertoires UNIX.

Le répertoire `racine` (symbolisé par `/`) d'un système UNIX contient typiquement (voir la page de `man` intitulée `hier(7)`) :

[1] le lecteur se rapportera à la description de `Portage` sur http ://www.gentoo.org.

```
drwxr-xr-x   2  root  root   2048  Aug  25  14 :04  bin
drwxr-xr-x   2  root  root   1024  Sep  16  10 :36  boot
drwxr-xr-x   7  root  root  35840  Aug  26  17 :08  dev
drwxr-xr-x  41  root  root   4096  Sep  24  20 :55  etc
drwxr-xr-x  24  root  root   1024  Sep  27  11 :01  home
drwxr-xr-x   4  root  root   3072  May  19  10 :05  lib
drwxr-xr-x   2  root  root  12228  Dec  15   1998  lost+found
drwxr-xr-x   7  root  root   1024  Jun   7  11 :47  mnt
drwxr-xr-x  14  root  root   4096  Apr  27  23 :12  opt
dr-xr-xr-x  80  root  root      0  Sep  16  10 :36  proc
drwx------  27  root  root   4096  May   6  07 :23  root
drwxr-xr-x   3  root  root   3072  Sep  23  23 :41  sbin
drwxr-xr-x  10  root  root      0  May   5  15 :20  sys
drwxrwxrwt   5  root  root   4096  Sep  28  18 :12  tmp
drwxr-xr-x  25  root  root   1024  May  29  10 :23  usr

drwxr-xr-x  17  root  root   4096  avr  29  10 :01  var
```

Le répertoire `/usr` qui est généralement monté depuis une partition séparée, ne devrait contenir que des données partageables, en lecture-seule, afin d'être monté par plusieurs machines utilisant Linux. Il devrait contenir les sous-répertoires suivants :

`/usr/X11R6`	système X-Window, version 11 release 6.
`/usr/bin`	répertoire principal pour les programmes exécutables. La plupart des programmes nécessaires aux utilisateurs, et pas indispensables pour démarrer ou réparer le système sont placés ici, A l'exception des programmes installés uniquement sur cette machine
`/usr/dict`	remplacé par /usr/share/dict
`/usr/doc`	remplacé par /usr/share/doc
`/usr/etc`	Les fichiers de configuration à partager entre plusieurs machines d'un site donné doivent être stockés dans ce répertoire. Néanmoins, les programmes devraient toujours faire référence aux fichiers dans /etc. On construit alors des liens symboliques depuis /etc vers les fichiers appropriés de /usr/etc

/usr/games	Fichiers exécutables des jeux et programmes éducatifs (facultatif)
/usr/include	Fichiers d'en-tête pour le compilateur C
/usr/lib	Contient les bibliothèques, y compris les bibliothèques dynamiques, ainsi que quelques exécutables qu'on n'invoque normalement pas directement. Des applications complexes peuvent disposer de leurs propres sous-répertoires ici
/usr/local	Emplacement habituel pour les programmes spécifiques à la machine ou au site
/usr/man	Remplacé par /usr/share/man
/usr/sbin	contient les programmes d'administration du système qui ne sont pas indispensables pour le démarrage, pour monter /usr, ou pour les réparations du système de fichiers.
/usr/share	on tient des sous-répertoires avec les données des applications, susceptibles d'être partagées entre différentes architectures avec le même système d'exploitation. On trouve souvent ici des choses qui se trouvaient auparavant dans /usr/doc ou /usr/lib ou encore /usr/man
/usr/src	Fichiers sources de différentes parties du système, inclus à titre de référence dans cer- tains paquetages. Ne pas placer ici de projets personnels, car les fichiers se trouvant dans /usr doivent être considérés comme uniquement accessibles en lecture, sauf durant l'installation de nouveaux logiciels.
/usr/tmp	Obsolète, ce doit être un lien symbolique vers /var/tmp. Présent pour des raisons de compati- bilité, ne devrait jamais être utilisé.

Le répertoire `/usr/local` est similairement structuré (voir `hier(7)`). De ceci, il ressort que tous les répertoires contiennent un jeu analogue de sous-répertoires. Ce jeu de sous-répertoires peut être appelé la s*uperstructure des répertoires* ou plus simplement *superstructure*. [Ceci est un nouveau terme qui n'a pas été utilisé jusqu'ici par les administrateurs UNIX et qui est introduit par Paul Sheer].

La superstructure contient toujours les sous-répertoires `bin` et `lib` ; les autres sont optionnels.

Chaque paquet sera installé sous une superstructure. Ceci signifie qu'au cours du désempaquet, les fichiers seront distribués dans de nombreux sous-répertoires de la superstructure. Un paquet RedHat sera toujours installé dans la superstructure `/usr` ou `/`, sauf s'il s'agit d'une application du système X Window. Alors son installation aura lieu dans la superstructure `/usr/X11R6`. Certaines applications de grande taille peuvent être installées dans la superstructure `/opt/<nom_de_paquet>`, et les paquets "faits-maison" seront usuellement installés dans la superstructure `/usr/local` (où `/local` signifie "spécifique à cette machine"). La superstructure de répertoires sous laquelle un paquet est installé est parfois appelée *préfixe d'installation* (ou *installation prefix*). Les paquets n'installent presque jamais leurs fichiers parmi des superstructures différentes. [Les fichiers de configuration font exception à cette règle. Ils sont répartis dans `/etc`].

Typiquement, la majorité d'un système est regroupée sous `/usr`. Ce répertoire peut être en lecture seule, vu que les paquets ne devraient jamais être écrits dans ce répertoire. Toute écriture se fait dans `/var` ou `/tmp` (`/usr/var` et `/usr/tmp` sont souvent liés par un lien symbolique à `/usr` ou `/tmp`, respectivement). La petite partie sous `/` qui n'appartient à aucune autre superstructure (soit environ 40 méga-octets) réalise l'essentiel des tâches d'administration. Ce sont les commandes nécessaires au montage et à la réparation du système en absence du répertoire `/usr`.

Voici la liste des sous-répertoires et leur description :

bin contient les *exécutables binaires*. Usuellement, tous les répertoires `bin` se trouvent dans la variable d'environnement `PATH` ; le shell effectue une recherche des programmes binaires dans ces répertoires.

sbin contient les *exécutables binaires du superutilisateur*. Il s'agit de programmes exclusivement dévolus à l'administration. Seul le `PATH` du supertuilisateur contient le chemin vers ces exécutables.

lib contient les *bibliothèques* (ou *libraries* en anglais). C'est là que toutes les données nécessaires aux programmes se trouvent. La plupart des paquets ont leur propre sous-répertoire sous `lib` afin d'y stocker leurs fichiers de données. Les *bibliothèques liées dynamiquement* (ou *Dynamically Linked Libraries* ; *DLL* ou fichiers avec le suffixe `.so`) sont directement logées dans `lib`. [Les bibliothèques liées dynamiquement hébergent des codes exécutables partagés par plusieurs programmes du répertoire bin de manière à consommer moins d'espace disque et d'espace mémoire].

etc *Et Caetera*. Il s'agit d'un répertoire accueillant les fichiers de configuration.

var contient les *données variables* sous forme de fichiers de données continuellement recréés ou mis à jour.

doc contient la *documentation* ; ce répertoire est étudié au chapitre 17.

man contient les *pages du manuel* ; ce répertoire est étudié au chapitre 17.

info contient les *pages d'info* ; ce répertoire est étudié au chapitre 17.

share contient les *données partagées* (ou *shared data*). Ce sont des fichiers indépendant de l'architecture. C'est dans ce répertoire que se trouvent les

fichiers qui ne dépendent pas du type de plate-forme matérielle. Ceci permet de les partager sur des machines différentes même si ces dernières ont des types de processeurs différents.

include C contient les *fichiers d'en-tête*. Ceux-ci sont associés au développement logiciel.

src C contient les *fichiers sources*. Il s'agit des sources du noyau et/ou des paquets construits localement.

tmp contient les *fichiers temporaires*. C'est là qu'un programme en cours crée des fichiers à usage temporaire.

18.3 LINUX sur une seule disquette.

Une disquette de 1,44 Mo peut très bien contenir un système LINUX pour autant que vous éliminiez tous les fichiers non-indispensables d'une ancienne distribution Slackware à noyau 2.0.3x, par exemple. Vous pouvez compiler un petit noyau qui, une fois compressé, aura une taille de 400 ko environ (voir le chapitre 43). Le système de fichiers peut être réduit à une taille de 2 à 3 Mo pour le strict nécessaire, ce qui n'occupera plus après compression qu'un Mo environ. Si la taille totale est inférieure à 1,44 Mo vous pourrez disposer d'une version LINUX sur une disquette. La liste des fichiers pourrait ressembler à celle reprise dans le tableau 18.1.

Notez que le répertoire `/etc` diffère de celui d'une distribution RedHat. Le système de fichiers de démarrage est considérablement simplifié sous Slackware.

Le répertoire `/lib/modules` a été éliminé pour la création de la disquette. Il faut noter qu'en principe, le fichier `/lib/modules/2.0.36` devrait contenir les pilotes du noyau chargeables dynamiquement (ces pilotes sont aussi appelés "modules"). Au lieu de cela, ici, tous les pilotes sont compilés dans le noyau pour des raisons de simplicité (ceci est expliqué au chapitre 43).

A titre d'exercice, efforcez-vous de créer une distribution sur une seule disquette. Cette tâche devrait être très instructive pour tout administrateur sérieux. A tout le moins, vous devriez regarder de très près les commandes des répertoires `bin` et `sbin`. Ensuite, naviguez parmi les pages de man qui vous paraissent peu famillières.

Le système de fichiers décrit dans le tableau 18.1 provient du paquet `morecram-1.3` disponible sur `http ://rute.sourceforge.net/morecram-1.3.tar.gz`. Il peut être téléchargé pour fournir un méthode d'appoint très utile en cas d'installation et de procédure de secours (*rescue*). Notez qu'il y a *beaucoup* de méthodes de ce type disponibles, plus courantes que `morecram`.

TAB. 18.1 – Linux sur une disquette.

/bin	/etc	/lib	/sbin	/var
/bin/sh	/etc/default	/lib/ld.so	/sbin/e2fsck	/var/adm
/bin/cat	/etc/fstab	/lib/libc.so.5	/sbin/fdisk	/var/adm/utmp
/bin/chmod	/etc/group	/lib/ld-linu.so.1	/sbin/fsck	/var/adm/cron
/bin/chown	/etc/host.conf	/lib/libcurses. so.1	/sbin/ifconfig	/var/spool
/bin/cp	/etc/hosts	/lib/libc. so.5.3.12	/sbin/iflink	/var/spool/uucp
/bin/pwd	/etc/inittab	/lib/libtermcap. so.2.0.8	/sbin/ifsetup	/var/spool/uucp/ SYSLOG
/bin/dd	/etc/issue	/lib/libtermcap. so.2	/sbin/init	/var/spool/uucp/ ERRLOG
/bin/df	/etc/utmp	/lib/libtext2fs. so.2.3	/sbinmke2fs	/var/spool/locks
/bin/du	/etc/networks	/lib/libcom_err. so.2	/sbin/mkfs	/var/tmp
/bin/free	/etc/passwd	/lib/libcom_err. so.2.0	/sbinmkfs/minix	/var/run
/bin/gunzip	/etc/profile	/lib/libex2fs.so.2	/sbin/mklost+ found/sbin	/var/run/utmp
/bin/zip	/etc/protocols	/lib/libm.so.5.0.5	/sbin/mkswap	
/bin/hostname	/etc/rc.d	/lib/libm.so.5	/sbin/mount	/home/user
/bin/login	/etc/rc.d/rc.0	/lib/cpp/lib	/sbin/route	
/bin/ls	/etc/rc.d/rc.K		/sbin/shutdown	/mnt
/bin/mkdir	/etc/rc.d/rc/M	/usr	/sbin/swapoff	
/bin/mv	/etc/rc.d/rc.S	/usr/admin	/sbin/swapon	/proc
/bin/ps	/etc/rc.d/rc.inet1	/usr/bin	/sbin/telinit	
/bin/rm	/etc/rc.d/rc.6	/usr/bin/less	/sbin/umount	/tmp
/bin/stty	/etc/rc.d/rc.4	/usr/bin/more	/sbin/agetty	
/bin/su	/etc/rc.d/rc.inet2	/usr/bin/sleep	/sbin/update	/dev/<various- devices>
/bin/sync	/etc/resolv.conf	/usr/bin/reset	/sbin/reboot	
/bin/zcat	/etc/services	/usr/bin/zless	/sbin/netcfg	
/bin/dircolors	/etc/termcap	/usr/bin/file	/sbin/killall5	
/bin/mount	/etc/motd	/usr//bin/fdformat	/sbin/fsck.minix	
/bin/umount	/etc/magic	/usr/bin/strings	/sbin/halt	
/bin/bash	/etc/DIR_COLORS	/usr/bin/zgrep	/sbin/badblocks	
/bin/domainname	/etc/HOSTNAME	/usr/bin/nc	/sbin/kerneld	
/bin/head	/etc/mtools	/usr/bin/which	/sbin/fsck.ext2	
/bin/kill	/etc/ld.so.cache	/usr/bin/grep		
/bin/tar	/etc/psdevtab	/usr/sbin		
/bin/cut	/etc/mtab	/usr/showmount		
/bin/uname	/etc/fastboot	/usr/chroot		
/bin/ping		/usr/spool		
/bin/ln		/usr/tmp		
/bin/ash				

Chapitre 19

Les périphériques d'UNIX.

UNIX a été conçu de manière à permettre un accès transparent aux périphériques matériels sur diverses architectures CPU. Sa philosophie consiste également à disposer d'accès aux périphériques grâce à un même jeu d'utilitaires en ligne de commandes.

19.1 Fichiers de périphériques.

UNIX possède une très belle méthode permettant aux programmes d'accéder au matériel : chaque périphérique est considéré comme un fichier. Pour démontrer cet aspect, essayons de visualiser ce que contient le fichier `/dev/hda` (vous devez être `superutilisateur` pour réaliser cela) :

```
less -f /dev/hda
```

Bien sûr, `/dev/hda` n'est pas un fichier du tout, en réalité. Lorsque vous pratiquez par la commande que nous venons d'utiliser, vous lisez réellement ce qui se trouve sur le premier disque physique de votre machine. De fait, `/dev/hda` est un fichier de périphérique. Tous les périphériques sont stockés dans le répertoire `/dev`.

Les fichiers de périphériques permettent l'accès au matériel. Si votre carte son est installée et configurée, vous pouvez tester ceci :

```
cat  /dev/dsp > mon_enregistrement
```

Prononcez quelques mots dans votre microphone et ensuite, tapez :

```
cat mon_enregistrement > /dev/dsp
```

Le système émettra les sons que vous avez enregistrés, via les hauts-parleurs. Notez que ceci ne fonctionne pas toujours parce que le volume et/ou la vitesse d'enregistrement ne sont peut-être pas correctement adaptés.

Si, par ailleurs, aucun programme ne fait usage de votre souris, vous pouvez tester ceci :

```
cat /dev/mouse
```

Si, à présent, vous déplacez votre souris, les commandes du protocole qui y sont associées seront écrites directement sur votre écran (cela ressemblera à du gribouillis). C'est une manière directe et aisée de constater que votre souris fonctionne bien (en particulier, lorsque vous désirez tester un port série). Si jamais ce test ne fonctionnait pas (parce qu'une commande a préconfiguré votre port série d'une manière particulière), essayez :

```
cu -s 1200 -l /dev/mouse
```

A plus bas niveau, les programmes accédant aux fichiers de périphériques pratiquent de deux manières :

- soit ils lisent depuis un périphérique et y écrivent en manipulant des données de masse (de manière similaire à ce que font `less` et `cat` dans les exemples vus précédemment),
- soit ils utilisent la fonction **C** `ioctl` (*IO Control*) pour configurer le périphérique. Dans le cas de la carte son, ceci peut permettre le basculement du mode mono vers le mode stéréo et vice-versa.

Etant donné que chaque périphérique (sauf les cartes réseau) peut être amenagé pour correspondre à ces deux modes, la méthode UNIX est toujours utilisée depuis son développement. Elle demeure une méthode de référence d'accès au matériel.

19.2 Fichiers blocs et caractères.

Les périphériques peuvent généralement être répartis en deux grandes catégories : (i) les périphériques à accès aléatoire (disques, lecteurs de bandes magnétiques), (ii) les périphériques série (souris, cartes son, terminaux).

L'accès aux périphériques en mode aléatoire se fait parmi de grands blocs continus de données stockées de manière persistante. La lecture y est réalisée par unités discrètes (pour la plupart des disques, cela se fait par groupe de 1024 octets). Ces périphériques sont appelés périphériques bloc (*block devices*). La commande `ls -l /dev/hda` affiche une ligne avec une lettre `b` à gauche indiquant ainsi que votre disque dur est un périphérique bloc :

```
brw-r- - - -  1 root    disk    3,   64 Apr 27   1995 /dev/hdb
```

En revanche, l'accès aux périphériques série se fait un octet à la fois. Les données peuvent être lues ou écrites une seule fois seulement. Par exemple, après qu'un octet ait été lu depuis votre souris, le même octet ne peut plus être lu par un autre programme. Les périphériques série sont appelés *périphériques caractères* (*character devices*) et sont caractérisés par un `c` sur la partie gauche de la ligne affichée via `ls`. Votre périphérique `/dev/dsp` (*Digital Signal Processor*, c'est-à-dire votre carte son) se caractérise ainsi :

```
crw-r- -r- -  1 root    sys    14,   3 Jul 18   1994 /dev/dsp
```

19.3 Nombres *mineurs* et *majeurs* de fichiers.

Les périphériques sont caractérisés par des *nombres majeurs* (ou *major device numbers*). Par exemple, tous les disques SCSI présentent un *nombre majeur* égal à 8. Par ailleurs, chaque périphérique individuel possède un *nombre mineur*. Ainsi, `/dev/sda` est caractérisé par un *nombre mineur* égal à 0. Les nombres majeur et mineur permettent au noyau d'identifier les périphériques. Le nom de fichier est arbitraire mais il est choisi de manière à apporter de la cohérence. Vous pouvez déterminer les nombres majeur et mineur (`8, 0`) grâce à `ls`, cette commande étant appliquée à `/dev/sda` :

```
brw-r- - - -  1 root    disk    8,   0 May 5   1998 /dev/sda
```

19.4 Noms usuels des périphériques.

Une liste des périphériques communément rencontrés (ainsi que leur description) est donnée ci-dessous. Les nombres majeurs sont placés entre parenthèses. La documentation complète pour les périphériques se trouve dans le fichier `/usr/src/linux/Documentation/devices.txt`.

/dev/hd?? `hd` signifie *hard disk* (disque dur) mais en se restreignant aux périphériques *IDE*, c'est-à-dire aux disques durs les plus communs. La première lettre suivant `hd` indique la préséance des disques :
/dev/hda (3) premier maître (*first drive* ou *primary master*),
/dev/hdb (3) premier esclave (*second drive* ou *primary slave*),
/dev/hdc (22) second maître (*third drive* ou *secondary master*),
/dev/hdd (22) second esclave (*fourth drive* ou *secondary slave*),
Si vous accédiez à un des ces périphériques (avec `less /dev/hda`, par exemple), la lecture se ferait directement sur le disque en commençant du premier secteur de la première piste jusqu'au dernier secteur de la dernière piste.
Les partitions sont nommées `/dev/hda1`, `/dev/hda2`, etc. ce qui signifie respectivement la première, la seconde,... partition du disque physique `a`.

dev/sd?? (8) `sd` signifie *SCSI disk* (disque SCSI), un type de disque haut-de-gamme fréquemment utilisé sur les serveurs. `sda` est le premier disque physique et la logique de dénomination est la même que pour les disques IDE. Cependant, la détection des disques SCSI est réalisée à l'aide d'un SCSI ID (*SCSI IDentifier*), ce qui n'est pas le cas des disques IDE. `/dev/sda1` est la première partition du premier maître, etc.

dev/ttyS? (4) Ce sont des périphériques série dénombrés à partir de `0`. `/dev/ttyS0` est votre premier port-série (il est équivalent à port COM1 sous MS-DOS ou Windows). Si vous avez une carte multiport, les numéros de ports peuvent monter jusqu'à 32, 64 ou même plus.[1]

[1] NdT : le terme **tty** signifie strictement *teletypewriter* et, il désigne sous UNIX :
- tout type de terminal (le terme est parfois employé pour désigner un terminal contrôlant un job donné),
- le nom de la commande qui affiche le nom du terminal courant,
- tout port série, que le périphérique connecté ou non un terminal. Donc, ces périphériques portent un nom de type **tty?**. Il est à noter que les deux dernières définitions peuvent introduire une confusion.

/dev/psaux (10) C'est une souris PS/2.

/dev/mouse Il s'agit d'un lien symbolique vers `/dev/ttyS0` ou `/dev/psaux`. D'autres périphériques sont également supportés.

/dev/modem Il s'agit d'un lien symbolique vers `/dev/ttyS1` ou tout port associé au modem.

/dev/cua ? (4) Identique à `ttyS ?`; désormais tombé en désuétude.

/dev/fd ? (2) `fd` signifie *floppy disk*. `fd0` est équivalent au lecteur `A :` (et `fd1` au lecteur `B :`) de Microsoft. Les périphériques `fd0` et `fd1` autodétectent le format des disquettes mais vous pouvez spécifier de manière explicite une densité plus élevée en utilisant un nom de périphérique tel que `/dev/fd0H1920` qui vous donne accès à une disquette de 3.5 pouces (3.5") de 1.88 Mo, une fois formatée. D'autres lecteurs de disquettes sont repris dans le tableau 19.1.
Consultez la section 20.3.4 pour savoir comment procéder au formatage.

/dev/par ? (6) désigne les ports en mode parallèle. (*parallel port*). `/dev/par0` est votre port parallèle (ou LPT1 sous DOS).

/dev/lp ? (6) Imprimante en ligne. Identique à `/dev/par ?`.

/dev/urandom est un générateur de nombres aléatoires. La lecture à partir de ce périphériques engendre des nombres pseudo-aléatoires.

/dev/st ? (9) Il s'agit d'un lecteur de bandes magnétiques SCSI utilisé pour les sauvegardes.

/dev/zero (1) est un périphérique qui, à la demande, produit des octets de valeur "zéro". Il est utile si vous voulez produire un bloc de zéros. Son utilisation avec la commande `dd` est donnée à la section 19.5.2.

/dev/null (1) C'est le périphérique "null" (*null device*) qui ne lit rien. Cependant tout ce que vous y écrivez est éliminé. Ceci est très utile pour éliminer l'affichage normalement produit par un programme, par exemple.

/dev/pd ? désigne un *disque IDE à port parallèle.*

/dev/pcd ? désigne un *CD-ROM ATAPI à port parallèle.*

/dev/pf ? désigne un *disque ATAPI à port parallèle.*

/dev/sr ? signifie CD-ROM SCSI.

/dev/scd ? idem.

/dev/sg ? désigne un *périphérique SCSI générique*. C'est une interface de commandes SCSI à caractère général pour les périphériques comme les numériseurs (ou *scanners* en anglais).

/dev/fb ? (29) pour *frame buffer*. Le mode "frame buffer" est une projection de la mémoire RAM d'une carte graphique vers la mémoire RAM principale. Ceci permet à des applications d'accéder à la mémoire graphique sans devoir communiquer directement avec la carte graphique.

/dev/cdrom est un lien symbolique vers `/dev/hda`, `/dev/hdb` ou `/dev/hdc`. Ce lien peut également pointer vers un CD-ROM SCSI.

/dev/ttyI ? modems RNIS (pour *Réseau Numérique à Intégration de Services*) dont la dénomination anglo-saxonne est ISDN (*Integrated Services Digital Network*).[2]

[2] NdT : par exemple *Numéris*.

TAB. 19.1 – Noms de lecteurs de disquettes.

l	0 1	lecteur A : lecteur B : *(Note : Les lecteurs sont notés /dev/fdlmnnnn.)*
m	d h q D H E u	"double densité" 360 Ko (5.25") "haute densité" 1.2 Mo (5.25") "quadruple densité" (5.25") "double densité" 720 Ko (3.5") "haute densité" 1.44 Mo (3.5") "Densité extra" (3.5") Toute disquette de 3.5". Notez que désormais u remplace D, H et E, ce qui permet à l'utilisateur de décider si la disquette présente une densité suffisante pour le format.
n	360 410 410 720 800 820 830 880 1040 1120 1200 1440 1476 1494 1600 1680 1722 1743 1760 1840 1920 2880 3200 3520 3840	La taille du format. Avec D, H et E, toutes les combinaisons ne sont pas permises dans le cas des disquettes de 3.5". Ainsi, il n'y a pas de /dev/fd0D1440 parce que les disquettes de double densité ne gèrent pas 1440 ko. /dev/fd0H1440 et /dev/fd0H1920 sont probablement les seules qui vous intéresseront.

/dev/tty ? (4) console virtuelle. En fait, il s'agit du périphérique "terminal" pour les consoles virtuelles elles-mêmes. La numérotation va de /dev/tty1 à /dev/tty63.

/dev/tty ? ? (3) et **/dev/pty ? ? (2)** sont d'autres périphériques *TTY* utilisés pour émuler un terminal. Ils sont nommés pseudo-TTY et sont identifiés par une lettre et un nombre comme par exemple, ttyq3. Hormis pour les développeurs, ceci n'a qu'un intérêt théorique.

Le fichier /usr/src/linux/Documentation/devices.txt nous apprend aussi ceci :

– **Liens recommandés** :
Il est recommandé que ces liens existent sur votre système :

/dev/core	/proc	lien symbolique	rétro-compatibilié
/dev/ramdisk	ram0	lien symbolique	rétro-compatibilié
/dev/ftape	qft0	lien symbolique	rétro-compatibilié

/dev/bttv0	video0	lien symbolique	rétro-compatibilié
/dev/radio	radio0	lien symbolique	rétro-compatibilié
/dev/i2o*	/dev/i2o*	lien symbolique	rétro-compatibilié
/dev/scd?	sr?	lien physique	nom alternatif pour CD-ROM SCSI

– **Liens définis localement :**
Les liens suivants peuvent être établis localement pour se conformer à la configuration du système. Il s'agit simplement d'une table de pratiques existantes ; cela ne constitue pas une recommandation. Cependant, si ces liens existent, ils devraient être dévolus aux usages suivants :

/dev/mouse	mouse port	symbolique	souris
/dev/tape	tape device	symbolique	bande magnétique
/dev/cdrom	CD-ROM device	symbolique	CD-ROM
/dev/cdwriter	CD-writer	symbolique	graveur de CD
/dev/scanner	scanner	symbolique	scanner
/dev/modem	modem port	symbolique	dialout
/dev/root	root device	symbolique	système de fichiers racine
/dev/swap	swap device	symbolique	swap

/dev/modem ne devrait pas être utilisé dans le cas d'un modem supportant la gestion d'appels entrants (*dial-in*) aussi bien que celle d'appels sortants (*dial-out*), du fait que cela tend à induire des problèmes de verrouillage de fichiers. Si /dev/modem existe, il devait pointer sur le premier périphérique TTY (l'utilisation de périphériques d'appels externes est obsolète).
Concernant le matériel SCSI, /dev/tape et /dev/sr* devraient pointer vers les périphériques "combinés" (par exemple, /dev/st* et /dev/sr*), tandis que /dev/cdwriter et /dev/scanner devraient pointer vers les périphériques SCSI génériques (/dev/sg*).
/dev/mouse peut pointer vers le premier périphérique TTY, un périphérique matériel de souris ou encore une connexion pour un module de souris (par exemple, /dev/gpmdata).

– **Sockets et tubes :**
Les *sockets*non-transitoires et les tubes peuvent exister dans /dev.[3] Les entrées fréquentes sont :

[3] NdT : le terme *socket*, désormais consacré, désigne une connexion (comme une prise électrique).

/dev/printer	socket	socket local lpd
/dev/log	socket	socket local syslog
/dev/gmpdata	socket	multiplexeur de souris

19.5 dd, tar et astuces avec les fichiers blocs.

Le nom de la commande dd provient probablement de *disk dump* (littéralement "purge de disque"). Son fonctionnement est le même que celui de cat sauf qu'elle peut lire et écrire dans des blocs discrets. Essentiellement, elle lit et écrit sur les périphériques tout en convertissant les données. Cette commande est généralement utilisée selon un de ces trois modes suivants :

```
dd if=<fichier_entree> of=<fichier_sortie> [bs=<taille_de_blocs>] \
     [count=<nombre_de_blocs>] [seek=<decalage_sortie>]
     [skip=<decalage_entree>]

dd if=<fichier_entree> [bs=<taille_de_blocs>] [count=<nombre_de_blocs>] \
     [skip=<decalage_entree>] <fichier_sortie>

dd of=<fichier_sortie> [bs=<taille_de_blocs>] [count=<nombre_de_blocs>] \
     [seek=<decalage_sortie>] < <fichier_entree>
```

Pour utiliser dd, vous devez spécifier un fichier d'entrée et un fichier de sortie avec les options if= et of=, respectivement. Si l'option of= est omise, dd écrira vers stdout. Si l'option if= est omise, dd lit à partir de stdin. [Si vous ressentez une difficulté, rappelez-vous que dd fonctionne en termes d'entrée et sortie par rapport à lui-même].

Notez que dd est une commande implacable et destructrice qui ne devrait être utilisée qu'avec beaucoup de précautions.

19.5.1 Création d'une disquette de démarrage à partir des images de démarrage.

Pour créer une disquette de démarrage RedHat, il faut trouver le fichier boot.img sur ftp.redhat.com, et avec une nouvelle disquette, lancer :

```
dd if=boot.img of=/dev/fd0
```

Cette commande écrit l'image brute du disque sur la disquette. Toutes les distributions ont des images similaires afin de permettre la création de disquettes d'installation (et parfois de disquettes de secours).

19.5.2 Effacement des disques.

Si jamais vous essayez un repartionnement de votre disque LINUX en un disque DOS/Windows, vous devez savoir qu'FDISK de DOS/Windows est bogué concernant la recréation d'une table de partition. Un rapide :

```
dd if=/dev/zero of=/dev/hda bs=1024 count=10240
```

écrira une collection de zéros sur les 10 premiers Mo de votre disque IDE. Ceci nettoyera la table de partition aussi bien que toute information à propos du système de fichiers de manière à vous restituer un disque "flambant neuf".

Il est tout aussi aisé d'encoder des zéros sur une disquette :

```
dd if=/dev/zero of=/dev/fd0 bs=1024 count=1440
```

Cependant, même le transfert de zéros sur une disquette ne sera pas toujours suffisant. Des équipements spécialisés peuvent encore lire des supports magnétiques après qu'ils aient été effacés à plusieurs reprises. Si, cependant, vous écrivez des octets de manière aléatoire sur une disquette, il devient assurément impossible de déterminer ce qui y était inscrit au préalable :

```
mknod dev/urandom c 1  9
for i in 1 2 3 4 ; do
    dd if=/dev/urandom of=/dev/fd0 bs=1024 count=1440
done
```

19.5.3 Identification des données sur les disques.

Voici une jolie astuce pour trouver des informations à propos d'un disque dur :[4]

```
dd if=/dev/hda1 count=1 bs=512 | file --
```

Ceci vous donne `/dev/stdin : x86 boot sector, code offset 0x48`.

Pour déterminer le type d'une disquette, essayez :

```
dd if=/dev/fd0 count=1 bs=512 | file --
```

ce qui retournera : `x86 boot sector, system )k?/bIHC, FAT (12 bit)` dans le cas des disquettes formatées DOS.

19.5.4 Dupliquer un disque.

Supposons que vous possédiez deux disques IDE ayant la même taille. Pour autant que vous soyez sûr qu'ils ne contiennent pas de secteurs défectueux (*bad sectors*) et pour autant qu'*ils ne soient pas montés*, vous pouvez exécuter ceci :

```
dd if=/dev/hdc of=/dev/hdd
```

pour copier le disque entier et éviter de devoir installer un système d'exploitation "ab initio". Il n'y a pas d'intérêt à savoir quel est l'original (LINUX, Windows ou quelqu'autre système que ce soit) puisque chaque secteur est exactement répliqué. Aussi, le nouveau système sera-t-il parfaitement opérationnel.

Si la taille des disques diffère, il faudra utiliser `tar` ou `mirrordir` pour répliquer le système de fichiers de manière exacte.

[4] NdT : on peut aussi utiliser la commande `hdparm -i`

19.5.5 Sauvegardes sur disquettes.

Quel que soit le périphérique, vous pouvez utiliser la commande `tar`. Considérons les sauvegardes périodiques d'un disque IDE ordinaire au lieu d'une bande magnétique. Nous sauvons sur le second disque esclave :

```
tar -cvzf /dev/hdd /bin /boot /dev /etc /home /lib /sbin /usr /var
```

`tar` peut aussi sauvegarder sur de multiples disquettes :

```
tar -cvMf /dev/fd0 /home/simon
```

19.5.6 Sauvegardes sur bandes.

Traditionnellement, `tar` permet la sauvegarde sur des bandes magnétiques. La première commande :

```
mt -f /dev/st0 rewind
tar -cvf /dev/st0 /home
```

rebobine la bande "scsi tape 0" , alors que la seconde y archive le répertoire `/home`. Vous devriez éviter la compression des données avec les lecteurs de bandes parce qu'il leur arrive de produire des erreurs et qu'une seule de celles-ci peut rendre votre archive irrécupérable. La commande `mt` signifie magnetic tape et elle contrôle les périphériques de bandes SCSI génériques. Consultez `mt`(1).

19.5.7 Cacher le résultat d'un programme, créer des blocs de zéros.

Si vous ne désirez pas voir les messages ou les résultats émis par les programmes, il suffit d'ajouter l'expression `> /dev/null` à la commande d'exécution. Quels programmes sont-ils concernés ? Par exemple, nous ne sommes pas très souvent intéressés par le résultat de `make` [la commande make est discutée plus tard]. Ici, nous éliminons tous les messages, sauf les messages d'erreurs :

```
make > /dev/null
```

Donc, si tous les messages, y compris les messages d'erreurs, doivent être éliminés, il faut utiliser :

```
make >& /dev/null
```

ou,

```
make > /dev/null 2>&1
```

Le périphérique `/dev/null` trouve des utilisations innombrables dans les scripts de shell pour supprimer les messages en sortie de commandes ou pour alimenter une entrée factice (c'est-à-dire vide). `/dev/null` est un fichier sûr. Il est

souvent utilisé quand un fichier est requis pour certaine fonctionnalité dans les scripts de configuration ou, quand vous souhaitez désactiver une fonctionnalité particulière. Par exemple, le fait de spécifier un shell `/dev/null` pour certains utilisateurs dans le fichier password préviendra tout usage incertain d'un shell. C'est une méthode explicite pour spécifier que le compte de ces programmes utilisateurs *n'*autorise *pas* de connexion à un shell.

Vous pouvez également utiliser `/dev/null` afin de créer un fichier ne contenant rien :

```
cat /dev/null mon_fichier
```

et pour créer un fichier ne contenant que de zéros :

```
dd if=/dev/zero bs=1024 count=<nombre_kilo-octets> > mon_fichier
```

19.6 Créer un fichier `/dev/...` avec `mknod` et `/dev/MAKEDEV`.

Bien que tous les fichiers soient listés dans le répertoire `/dev`, il est toujours possible de créer un fichier de périphérique n'importe où sur le système en utilisant la commande `mknod` :

```
mknod [-m <mode>] <nom_fichier> [b|c] <numero_majeur> <numero_mineur>
```

Les lettres `b` et `c` sont requises pour créer un périphérique bloc (*block*) ou caractère (*character*), respectivement. Pour démontrer cela, essayons :

```
mknod -m 0600 ~/disquette b 2 0
ls -al /dev/fd0 ~/disquette
```

De cette manière, `/disquette` pourra être utiliser exactement comme l'est `/dev/fd0`.

Observez attentivement le *mode* (c'est-à-dire les droits) de `/dev/fd0`. Seuls `root` et les utilisateurs du groupe `floppy` devraient pouvoir lire ce fichier et y écrire, étant donné que nous ne voulons bien évidemment pas qu'un utilisateur quelconque puisse se connecter à distance et écrire sur une disquette.

En fait, ceci est la raison essentielle pour avoir des périphériques représentés comme des fichiers et ce, à l'emplacement approprié. Les fichiers UNIX tolèrent de manière naturelle le contrôle des accès par les groupes des fichiers, et par conséquent des périphériques associés.

Pour créer des périphériques qui manquent dans le répertoire `/dev` (c'est-à-dire des périphériques ésotériques qui ne sont pas présent par défaut), extrayez d'abord les nombres majeur et mineur en consultant le fichier `/usr/src/linux-/Documentation/devices.txt`. Ensuite, utilisez la commande `mknod`. Remarquez que cette méthode est assez lourde. Le script `/dev/MAKEDEV` est usuellement disponible pour parer cet inconvénient. *Vous devez absolument être dans le répertoire* `/dev` *avant que le script ne soit exécuté.*

Une utilisation typique de `MAKEDEV` est :

```
cd /dev
./MAKEDEV -v fd0
./MAKEDEV -v fd1
```

de manière à créer un jeu complet de lecteurs de disquettes.

La page de `man` de `MAKEDEV` contient tous les détails. En particulier, elle donne l'information suivante :

Remarquez que les programmes donnant l'erreur "ENOENT : No such file or directory" indiquent normalement que le fichier de périphérique est manquant, alors que "ENOENT : No such device" signifie que, concernant le noyau, le pilote n'est pas configuré ou chargé.

19.7 `devfs`*.[5]

19.7.1 Fonctionnement général de `devfs`.

`devfs` signifie *Device File System* (ou *système de fichier de périphériques*). Ce système de gestion de périphériques a été conçu pour éviter les inconvénients associés au répertoire `/dev`. En effet, la gestion des droits sur les fichiers de `/dev` est complexe puisqu'à côté des périphériques de votre système, il existe bien d'autres fichiers de périphériques prévus mais non-utilisés. En outre, quand un programme interagit avec un fichier de périphérique, la partition `/` est automatiquement montée en lecture/écriture alors que souvent le montage en lecture seul suffit. Finalement, `/dev` ne peut être placé sur une partition séparée car `mount` nécessite `/dev` pour monter les partitions.

`devfs` a été incorporé au noyau LINUX pour corriger ces problèmes. Avec `devfs`, vous ne devez plus vous préoccuper des paires "majeur/mineur". Elles sont encore supportées pour des raisons de compatibilité avec le schéma existant, sans être nécessaires toutefois. Pour utiliser `devfs`, les options :

```
[*]/dev file system support
[*]  Automatically mount at boot
```

doivent être sélectionnées dans le noyau (ces options se trouvent dans la section `File Systems – Pseudo filesystems`). Voir la section 43.11 pour la sélection des options et la compilation du noyau.

`devfs` utilise des sous-répertoires pour améliorer la lisibilité du répertoire `/dev`. Ainsi, les périphériques IDE sont logés dans le sous-répertoire `/dev/ide/`. Les périphériques SCSI se retrouvent dans `/dev/scsi/`. Tous les deux présentent la même structure de sous-répertoires à un niveau encore plus bas.

Les disques SCSI et IDE sont contrôlés par un adaptateur *host* (*hôte*) se trouvant sur la carte mère ou sur une carte séparée. Comme il peut y avoir différents canaux (appelés *bus*), il peut y avoir plusieurs IDs sur chaque canal, permettant d'identifier un disque de manière univoque. Cet identifiant est appelé la *cible* (ou *target* en anglais). Pour les périphériques SCSI, il peut y avoir divers *LUN* (*Logic Unit Number* ou encore *numéro d'unité logique*).

Donc, en considérant `/dev/hda1`, la structure du répertoire `/dev/ide` se présente de la manière suivante pour ce périphérique :

[5] Cette section est un ajout au texte original.

```
/dev/ide/host0/bus0/target0/lun0/part1
```

L'arborescence de `/dev` devrait ressembler à :

cdroms	cpu	discs	floppy
ide	input	loop	misc
netlink	printers	pts	pty
scsi	sg	shm	sound
sr	usb	vc	vcc

Pour assurer une compatibilité avec le système simple vu à la section 19.6, le démon `devfsd` établit des liens symboliques entre les "anciens" noms et les nouvelles dénominations de fichiers. Par exemple, on a pour `/dev/hda1` :

```
lr-xr-xr-x 1 root root 33 oct 29 20 :51 /dev/hda1 -> ide/host0/bus0/target0/lun0/part1
```

Le fichier de configuration de `devfs` est `/etc/devfsd.conf`. Il ressemble à ceci :

```
# Sample /etc/devfsd.conf configuration file.
# Richard Gooch <rgooch@atnf.csiro.au> 3-JUL-2000
#
# The Gentoo Linux Team - http ://www.gentoo.org/
# - Many fixes, etc
#
# $Header : /home/cvsroot/gentoo-src/rc-scripts/etc/devfsd.conf,v 1.16 2003/05/18
21 :51 :34 azarah Exp $
# Enable full compatibility mode for old device names. You may comment these # out if you
don't use the old device names. Make sure you know what you're # doing!

REGISTER .* MKOLDCOMPAT
UNREGISTER .* RMOLDCOMPAT

# You may comment out the above and uncomment the following if you've
# configured your system to use the original "new" devfs names or the really # new names
#REGISTER vc/.* MKOLDCOMPAT
#UNREGISTER vc/.* RMOLDCOMPAT
#REGISTER pty/.* MKOLDCOMPAT
#UNREGISTER pty/.* RMOLDCOMPAT
#REGISTER misc MKOLDCOMPAT
#UNREGISTER misc RMOLDCOMPAT

# You may comment these out if you don't use the original "new" names REGISTER .*
MKNEWCOMPAT UNREGISTER .* RMNEWCOMPAT

# Enable module autoloading. You may comment this out if you don't use # autoloading
LOOKUP .* MODLOAD

# Uncomment the following if you want to set the group to "tty" for the
# pseudo-tty devices. This is necessary so that mesg(1) can later be used to #
enable/disable talk requests and wall(1) messages.
REGISTER ^pty/s.* PERMISSIONS -1.tty 0600
REGISTER ^pts/.* PERMISSIONS -1.tty 0600
# Uncomment this if you want permissions to be saved and restored
# NB : Do NOT change the following!
# Do not do this for pseudo-terminal devices
REGISTER ^pt[sy]/.* IGNORE
CHANGE ^pt[sy]/.* IGNORE
CREATE ^pt[sy]/.* IGNORE
DELETE ^pt[sy] IGNORE

REGISTER ^log IGNORE

CHANGE ^log IGNORE

CREATE ^log IGNORE
```

```
DELETE ^log IGNORE
REGISTER .* COPY /lib/dev-state/$devname $devpath
CHANGE .* COPY $devpath /lib/dev-state/$devname
CREATE .* COPY $devpath /lib/dev-state/$devname
DELETE .* CFUNCTION GLOBAL unlink /lib/dev-state/$devname
RESTORE /lib/dev-state

# You can force default like this :
# PERMISSIONS owner_and_group access_mode

# Video4Linux devices
REGISTER v4l/.* PERMISSIONS root.video 660

LOOKUP ^video$ CFUNCTION GLOBAL mksymlink ${mntpnt}/v4l/video0 video REGISTER ^v4l/video0
CFUNCTION GLOBAL mksymlink ${devpath} video
UNREGISTER ^v4l/video0 CFUNCTION GLOBAL unlink video

# ALSA/OSS stuff
# Comment/change these if you want to change the permissions on
# the audio devices
LOOKUP snd MODLOAD ACTION snd
LOOKUP dsp MODLOAD
LOOKUP mixer MODLOAD
LOOKUP midi MODLOAD
REGISTER sound/.* PERMISSIONS root.audio 660
REGISTER snd/.* PERMISSIONS root.audio 660

# Uncomment this to let PAM manage devfs
#REGISTER .* CFUNCTION /lib/security/pam_console_apply_devfsd.so pam_console_apply_single
$devpath

# Autoload the sg module if generic scsi driver compiled as module.
#LOOKUP ^sg$ MODLOAD ACTION sg
# Give the cdrw group write permissions to /dev/sg0
# This is done to have non root user use the burner (scan the scsi bus) #REGISTER
^scsi/host.*/bus.*/target.*/lun.*/generic PERMISSIONS root.cdrw 660
# General note for the following auto creation of symlinks :
#
#    If you change the device that the symlink points to,
#    you should also remove the symlink before restarting
#    devfsd

# Create /dev/cdrom for the first cdrom drive
LOOKUP ^cdrom$ CFUNCTION GLOBAL mksymlink cdroms/cdrom0 cdrom
REGISTER ^cdroms/cdrom0$ CFUNCTION GLOBAL mksymlink $devname cdrom UNREGISTER
^cdroms/cdrom0$ CFUNCTION GLOBAL unlink cdrom

# Create /dev/dvd for the second cdrom drive
# (change 'cdroms/cdrom1' to suite your setup)
# NOTE : We add the fully qualified path here, else some apps
# have problems to resolve the true device (drip comes to mind)
#LOOKUP ^dvd$ CFUNCTION GLOBAL mksymlink ${mntpnt}/cdroms/cdrom1 dvd #REGISTER
^cdroms/cdrom1$ CFUNCTION GLOBAL mksymlink ${devpath} dvd #UNREGISTER ^cdroms/cdrom1$
CFUNCTION GLOBAL unlink dvd

# Create /dev/cdrw for the first cdrom on the scsi bus
# (change 'sr0' to suite your setup)
#LOOKUP ^cdrw$ CFUNCTION GLOBAL mksymlink sr0 cdrw
#REGISTER ^sr0$ CFUNCTION GLOBAL mksymlink $devname cdrw
#UNREGISTER ^sr0$ CFUNCTION GLOBAL unlink cdrw

# Create /dev/mouse
LOOKUP ^mouse$ CFUNCTION GLOBAL mksymlink misc/psaux mouse
REGISTER ^misc/psaux$ CFUNCTION GLOBAL mksymlink $devname mouse
UNREGISTER ^misc/psaux$ CFUNCTION GLOBAL unlink mouse

# Manage USB mouse
REGISTER ^input/mouse0$ CFUNCTION GLOBAL mksymlink $devname usbmouse UNREGISTER
^input/mouse0$ CFUNCTION GLOBAL unlink usbmouse

REGISTER ^input/mice$ CFUNCTION GLOBAL mksymlink $devname usbmouse UNREGISTER

^input/mice$ CFUNCTION GLOBAL unlink usbmouse
```

```
# Support additional config installed by packages ...

INCLUDE /etc/devfs.d

# devfsd.conf ends here
```

Lorsque ce fichier est modifié, le démon `devfsd` doit être relancé par les commandes suivantes :

```
kill -s SIGHUP `pidof devfsd`
```

ou

```
killall -s SIGHUP devfsd
```

Par défaut, les liens de compatibilité sont activés et certaines distributions GNU/Linux ne peuvent fonctionner sans eux. Pour les supprimer, commentez les lignes du fichier `/etc/devfsd.conf` de la manière suivante :

```
# REGISTER .* MKOLDCOMPAT
# UNREGISTER .* RMOLDCOMPAT
```

Dans ce cas, le système doit être redémarré pour que les modifications soient prises en compte.

Lors du chargement d'un module, `devfsd` crée automatiquement le fichier périphérique correspondant. Pour désactiver ce comportement par défaut, commentez ou supprimez la ligne associée à la fonction d'autochargement dans `/etc/devfsd.conf` :

```
# LOOKUP .* MODLOAD
```

19.7.2 Attribution et modifications des droits sur les fichiers de `devfs`.

Utilisez la syntaxe suivante, qui traite le cas d'un périphérique CD-ROM :

```
REGISTER ^cdroms/.* PERMISSIONS root.cdrom 0660
```

Le second champ est une expression rationnelle désignant des fichiers commençant par `/dev`. Le quatrième champ représente le propriétaire et le groupe du fichier de périphérique. Le cinquième donne les droits sous forme octale.

Sur certaines distributions,[6] le comportement par défaut lors d'une modification des droits par `chmod` ou `chown` est que `devfsd` enregistre la modification au moment de la procédure d'extinction de l'ordinateur. Ce comportement est configuré par les lignes suivantes du fichier `/etc/devfsd.conf` :

[6] Gentoo, par exemple.

```
# Uncomment this if you want permissions to be saved and restored
# NB : Do NOT change the following !
# Do not do this for pseudo-terminal devices
REGISTER ^pt[sy]/.* IGNORE
CHANGE ^pt[sy]/.* IGNORE
CREATE ^pt[sy]/.* IGNORE
DELETE ^pt[sy] IGNORE
REGISTER ^log IGNORE
CHANGE ^log IGNORE
CREATE ^log IGNORE
DELETE ^log IGNORE
REGISTER .* COPY /lib/dev-state/$devname $devpath
CHANGE .* COPY $devpath /lib/dev-state/$devname
CREATE .* COPY $devpath /lib/dev-state/$devname
DELETE .* CFUNCTION GLOBAL unlink /lib/dev-state/$devname

RESTORE /lib/dev-state
```

De cette manière, les fichiers de périphériques modifiés sont transcrits dans `/lib/dev-state` à l'extinction de l'ordinateur et recopiés dans `/dev` lors du redémarrage du système.

Il est également possible qu'au démarrage, `/lib/dev-state` soit monté sur `/dev`. Il faut alors que `devfs` ne soit pas monté automatiquement au démarrage (voir au début de la présente section pour modifier les options du noyau) et que `/dev/console` existe. Par ailleurs, au début des scripts de démarrage de votre système, vous devez écrire :

```
mount --bind /dev /lib/dev-state mount -t devfs none /dev devfsd /dev
```

19.7.3 Modifications des droits avec PAM.

PAM signifie *Pluggable Authentification Modules*. Etant donné que PAM a la priorité en termes d'enregistrement de droits, la modification des droits avec PAM l'emporte sur la methode décrite à la sous-section 19.7.2. PAM utilise le fichier `/etc/security/console.perms` pour gérer les droits. La première partie de ce fichier décrit les groupes et la seconde les droits. Voici un exemple de fichier `/etc/security/console.perms` :

```
# /etc/security/console.perms
#
# This file determines the permissions that will be given to priviledged # users of the
console at login time, and the permissions to which to
# revert when the users log out.

# format is :
#   <class>=list of regexps specifying consoles or globs specifying files
#   file-glob|<class> perm dev-regex|<dev-class> \
#    revert-mode revert-owner[.revert-group]
# the revert-mode, revert-owner, and revert-group are optional, and default # to 0600,
root, and root, respectively.
#
# For more information :
# man 5 console.perms
```

```
# file classes -- these are regular expressions
<console>=tty[0-9][0-9]* vc/[0-9][0-9]* :[0-9]\.[0-9] :[0-9]
<xconsole>= :[0-9]\.[0-9] :[0-9]

# device classes -- these are shell-style globs
<serial>=/dev/ttyS*
<floppy>=/dev/fd[0-1]* \
         /dev/floppy/* /mnt/floppy*
<sound>=/dev/dsp* /dev/audio* /dev/midi* \
        /dev/mixer* /dev/sequencer* \
        /dev/sound/* /dev/snd/* /dev/beep \
        /dev/admm* \
        /dev/adsp* /dev/aload* /dev/amidi* /dev/dmfm* \
        /dev/dmmidi* /dev/sndstat
<cdrom>=/dev/cdrom* /dev/rdvd /dev/ide/*/*/*/*/cd /dev/scsi/*/*/*/*/cd \
        /dev/cdroms/* /dev/cdwriter* /mnt/cdrom*
<pilot>=/dev/pilot <
jaz>=/mnt/jaz*
<zip>=/mnt/pocketzip* /mnt/zip* <ls120>=/dev/ls120 /mnt/ls120*
<scanner>=/dev/scanner /dev/usb/scanner*
<rio500>=/dev/usb/rio500
<camera>=/mnt/camera* /dev/usb/dc2xx* /dev/usb/mdc800*
<memstick>=/mnt/memstick*
<flash>=/mnt/flash*
<diskonkey>=/mnt/diskonkey*
<rem_ide>=/mnt/microdrive*
<fb>=/dev/fb /dev/fb[0-9]* \
     /dev/fb/* <kbd>=/dev/kbd
<joystick>=/dev/js[0-9]*
<v4l>=/dev/video* /dev/radio* /dev/winradio* /dev/vtx* /dev/vbi* \
      /dev/video/* /dev/vttuner
<gpm>=/dev/gpmctl
<dri>=/dev/nvidia* /dev/3dfx*
<mainboard>=/dev/apm_bios
<burner>=/dev/scd* /dev/sg* /dev/pcd* /dev/pg* /dev/cdwriter /dev/scsi/*/*/*/*/generic
<usb>=/dev/usb/dabusb* /dev/usb/mdc800* /dev/usb/rio500 /dev/ttyUSB* \
      /proc/usb/[0-9][0-9][0-9]/[0-9][0-9][0-9]

# permission definitions
<console> 0660 <serial> 0660 root.tty
<console> 0660 <floppy> 0660 root.floppy
<console> 0600 <sound> 0600 root.audio
<console> 0600 <cdrom> 0660 root.cdrom
<console> 0600 <pilot> 0660 root.uucp
<console> 0600 <jaz> 0660 root.disk
<console> 0600 <zip> 0660 root.disk
<console> 0600 <ls120> 0660 root.disk
<console> 0600 <scanner> 0600 root
<console> 0600 <camera> 0600 root
<console> 0600 <memstick> 0600 root
<console> 0600 <flash> 0600 root
<console> 0600 <diskonkey> 0660 root.disk
<console> 0600 <rem_ide> 0660 root.disk
<console> 0600 <fb> 0600 root
<console> 0600 <kbd> 0600 root
<console> 0600 <joystick> 0600 root
<console> 0600 <v4l> 0600 root.sys
<console> 0700 <gpm> 0700 root
<console> 0600 <mainboard> 0600 root
<console> 0660 <burner> 0660 root.cdrw
<console> 0600 <usb> 0660 root.usb
<console> 0600 <rio500> 0600 root


<xconsole> 0600 /dev/console 0600 root.root
<xconsole> 0600 <dri> 0600 root
```

Concernant les groupes de périphériques, prenons l'exemple du groupe `cdrom` :

```
<cdrom>=/dev/cdrom* /dev/rdvd /dev/ide/*/*/*/*/cd /dev/scsi/*/*/*/*/cd \
        /dev/cdroms/* /dev/cdwriter* /mnt/cdrom*
```

La syntaxe indique d'abord le groupe (<cdrom>) et la liste des périphériques (/dev/cdrom* ... /mnt/cdrom*) qui appartiennent à ce groupe. La seconde partie indique comment les groupes décrits dans la première partie du fichier fonctionnent en termes de droits :

```
<console> 0600 <cdrom> 0660 root.cdrom
```

Le premier champ représente le contrôle du terminal. De manière générale, il s'agit du groupe <console>. PAM vérifie ce paramètre lors de chaque connexion sur le système. Si la connexion a lieu sur le périphérique appartenant au groupe <console>, PAM vérifie et modifie les droits de certains fichiers périphériques.

Le second champ reprend les droits attribuées aux fichiers de périphérique lors d'une connexion réussie. Quand un utilisateur se connecte au système et que les fichiers périphériques appartiennent au propriétaire ou au groupe par défaut, PAM enregistre l'utilisateur connecté comme s'il s'agissait du propriétaire. PAM attribue les droits décrits dans le deuxième champ. Dans ce cas, 0600 est employé : l'utilisateur a le droit d'accès en lecture et en écriture, à l'exclusion de tout autre utilisateur.

Le troisième champ contient le groupe de périphériques auquel les droits sont conférés. En l'occurrence, pour le groupe <cdrom>, tous les fichiers de périphériques relatifs au lecteur de CD-ROMs sont modifiés.

Le quatrième champ définit les droits associés au fichier périphérique après retour à l'état par défaut : si le propriétaire des fichiers de périphérique se déconnecte, PAM modifie alors les droits pour reprendre les droits de l'état par défaut, tels que décrits dans ce quatrième champ.

Le cinquième champ définit la propriété (avec le groupe, le cas échéant) donnée aux attributs du périphérique après le retour dans l'état par défaut : si le propriétaire des fichiers de périphériques se déconnecte, PAM remet le propriétaire précédent correspondant à l'état par défaut tel que décrit dans ce cinquième champ.

19.8 udev*.[7]

19.8.1 Avantages d'udev.

Avec les noyaux LINUX-2.6, de nouvelles technologies sont apparues. udev est l'une d'elles. Nous en verrons deux autres ci-après. udev –pour u*serspace implementation of* devfs– est un gestionnaire de périphériques de petite taille (~6 ko) mis en oeuvre dans l'espace utilisateur, contrairement à devfs dont le démon (devfsd) fonctionne dans l'espace noyau. L'objectif est de disposer d'un système ayant les caractéristiques suivantes :

– un fonctionnement dans l'espace utilisateur,

[7] Cette section est un ajout au texte original. Elle est directement inspirée de G.Kroah-Hartman, udev – A Userspace Implementation of devfs *in* Proceedings of the Linux Symposium, (Eds. A.J. Hutton, S. Donovan and C. C. Ross), July 23-26, 2003, Ottawa, Canada, pp.263-271.

- ajouts, suppressions et attributions de noms de périphériques dans `/dev` de manière dynamique, en s'appuyant sur `hotplug` (le mécanisme de détection des périphériques pouvant être connectés à chaud) et `sysfs` qui fournit divers éléments comme la localisation, les attributs (noms et numéros de série) ainsi que les nombres majeurs et mineurs des périphériques,
- règles cohérentes d' attribution de noms,
- existence d'une interface de programmation API (*Application Programming Interface*).

Outre le fonctionnement dans l'espace utilisateur, le premier avantage provient de la création, de la suppression ou de la réaffectation de noms de manière dynamique dans `/dev` d'où une simplification et une lisibilité accrues de ce répertoire.

Le second avantage vient de ce que les codes majeurs et mineurs sont attribués par le noyau lorsque les gestionnaires de périphériques s'initialisent. Il en découle que le noyau assure l'unicité des codes utilisés (suppression des risques de conflits entre deux périphériques) et qu'il n'y a plus de limitation du nombre de périphériques utilisables.

19.8.2 Installation d'`udev`.

Cette partie peut vous paraître difficile car elle s'appuie sur des sections vues ultérieurement (installation de paquets, compilation du noyau). Vous pourrez donc y revenir par la suite si vous ne vous sentez pas à l'aise.

La dernière version d'`udev` peut être téléchargée sur http ://www.us.kernel-.org/pub/linux/utils/kernel/hotplug/.

Le paquet qui doit être installé est `udev` (au moment de la préparation de cette section, la version stable était `udev-045`). Pour l'installer le paquet, utilisez les commandes `rpm` (RedHat), `urpmi` (Mandriva), `apt-get install` (Debian) ou `emerge` (Gentoo) [reportez-vous au chapitre 22 et plus précisément à la section 22.2 pour savoir comment installer un paquet] :

```
rpm -i udev-045
( urpmi udev )
( apt-get install udev )
( emerge udev )
```

Les paquets `hotplug` et `coldplug` doivent également être installés en utilisant les commandes décrites dans l'encadré ci-dessus. Dans le cas de la commande `rpm`, il est nécessaire de connaître la version du paquet. Pour cela utilisez :

```
rpm -qa | grep -i hotplug
rpm -qa | grep -i colplug
```

Si le paquet n'est pas présent sur le disque dur, il peut être installé aisément à partir du CD-ROM.

Lorsque ces paquets sont installés, vous avez deux possibilités pour utiliser `udev`.

Dans le premier cas, il faut recompiler le noyau (voir le chapitre 43). Puisque vous utilisez un noyau 2.6, descendez dans l'arborescence du noyau :

```
cd /usr/src/linux
make menuconfig
```

vérifiez que les options suivantes sont sélectionnées :

```
General setup --->
   [*] Support for hot-pluggable devices

File systems --->
 Pseudo filesystems --->
   [*] /proc file system support
   [*] Virtual memory file system support (former shm fs)
```

et que celles-ci sont désactivées :

```
File systems --->
 Pseudo Filesystems --->
   [ ] /dev file system support (OBSOLETE)
   [ ] Automatically mount at boot
```

L'option `/dev file system support (OBSOLETE)` pourrait rester cochée mais l'option `Automatically mount at boot` *doit* être désactivée.

Après avoir remonté votre partition `/boot` :

```
mount /boot
```

recompilez votre noyau (la première commande ne vaut que pour un noyau `2.6`, voir le chapitre 43) :

```
make && make modules_install
cp /usr/src/linux/arch/i386/boot/bzImage /boot/<noyau>
cp System.map /boot/System.map-<noyau>
cp .config /boot/config-<noyau>
modules-update
```

A présent, pourvu que l'entrée dans `/etc/lilo.conf` ou `/boot/grub/grub.conf` (ou encore `/boot/grub/menu.lst`, selon votre distribution) soit correctement définie (voir les chapitres 32 et 43), redémarrez votre machine. Si vous possédez une cartes graphique Nvidia, il se peut que vous observiez des problèmes. Rapportez-vous à l'article de Daniel Drake sur http ://www.reactivated.net/udev-rules.php.

La seconde possibilité consiste à laisser votre noyau en l'état, c'est-à-dire avec les options `/dev file system support (OBSOLETE)` et `Automatically mount at boot` cochées. Dans ce cas, vous pouvez passer une option au noyau. Elle sera prise en compte lors de l'amorçage. A cette fin, modifiez `/etc/lilo.conf`, `/boot/grub/grub.conf` ou `/boot/grub/menu.lst` pour y ajouter (entre parenthèse, la commande pour Gentoo) :

```
devfs=nomount
( gentoo=nodevfs )
```

19.8.3 Fichiers principaux.

Au minimum, le répertoire /etc/udev contient :
- le fichier cdsymlinks.conf,
- le fichier principal udev.conf et,
- les répertoires :
  - permissions.d, qui contient le fichier 50-udev.permissions répertoriant les droits sur les fichiers de périphériques,
  - rules.d , qui contient le fichier 50-udev.rules répertoriant les règles d'attribution de nom des fichiers périphériques et,
  - scripts. Ce dernier répertoire est composé des scripts cdsymlinks.sh (permettant la cartographie des cdrom, cdm, cdmrv, cdrw, dvd, dvdrw et dvdram), ide-devfs.sh (qui retourne les noms devfs pour les périphériques IDE) et scsi-devfs.sh (qui agit similairement pour les périphériques SCSI).

Enfin, /etc/dev.d peut contenir les répertoires defaut et net pour l'exécution d'udev. Par exemple, net est un programme invoqué par udev pour assurer que tout périphérique réseau q'udev renomme, maintient en activité le script hotplug avec le nom courant.

Les informations contenues dans udev.permissions devraient ressembler à ceci :

```
# /etc/udev/udev.permissions : permission/ownership map for udev
# $Header : /home/cvsroot/gentoo-x86/sys-fs/udev/files/udev.permissions,v 1.3
2004/01/01 03 :41 :24 azarah Exp $
# console devices console :root :tty :0600 tty :root :tty :0666
tty[0-9]* :root :tty :0660 vc/[0-9]* :root :tty :0660
# pty devices # Set this to 0660 if you only want users belonging
to tty group # to be able to allocate PTYs ptmx :root :tty :0666
pty[p-za-e][0-9a-f]* :root :tty :0660 tty[p-za-e][0-9a-f]* :root :tty :0660
pty/m* :root :tty :0660 vc/s* :root :tty :0660
...
...
# mainboard devices apm_bios :root :root :0600
# scsi devices sg* :root :disk :0660 pg* :root :disk :0660
cdwriter :root :disk :0660
# usb devices usb/dabusb* :root :usb :0660 usb/mdc800* :root :usb :0660
usb/rio500 :root :usb :0660 usb/legousbtower* :root :usb :0660
sgi_fetchop :root :root :666
iseries/vcd* :root :disk :660
iseries/vd* :root :disk :660
```

Par défaut, les droits sont fixés à 0660.

Voici un échantillon du ficher udev.rules (voir la sous-section 19.8.6) :

```
# /etc/udev/udev.rules : device naming rules for udev
#
# Gentoo specific rules, based a bit on devfs rules, but much simpler.
#
# There are a number of modifiers that are allowed to be used in some of the
# fields. See the udev man page for a full description of them.
```

```
############################################################
#
# For devfs similar /dev layout (neater)
#
############################################################

# cdrom symlinks and other good cdrom naming
BUS="ide", KERNEL="hd[a-z]", PROGRAM="/etc/udev/scripts/cdsymlinks.sh %k", SYMLINK="%c{1}
%c{2} %c{3} %c{4} %c{5} %c{6}"
BUS="scsi", KERNEL="sr[a-z]", PROGRAM="/etc/udev/scripts/cdsymlinks.sh %k", SYMLINK="%c{1}
%c{2} %c{3} %c{4} %c{5} %c{6}"
BUS="scsi", KERNEL="scd[a-z]", PROGRAM="/etc/udev/scripts/cdsymlinks.sh %k",
SYMLINK="%c{1} %c{2} %c{3} %c{4} %c{5} %c{6}"

# devfs-names for ide-devices (uncomment only one)
# /dev/ide/.../{disc,cd} and /dev/{cdroms,discs}/* type names BUS="ide", KERNEL="hd*",
PROGRAM="/etc/udev/scripts/ide-devfs.sh %k %b %n", NAME="%k", SYMLINK="%c{1} %c{2}"

# dri devices KERNEL="card*", NAME="dri/card%n"

# alsa devices

KERNEL="controlC[0-9]*", NAME="snd/%k"
KERNEL="hw[CD0-9]*", NAME="snd/%k"
KERNEL="pcm[CD0-9cp]*", NAME="snd/%k"
KERNEL="midiC[D0-9]*", NAME="snd/%k"
KERNEL="timer", NAME="snd/%k"
KERNEL="seq", NAME="snd/%k"

...
...
# vc devices
KERNEL="vcs", NAME="vcc/0", SYMLINK="%k"
KERNEL="vcs[0-9]*", NAME="vcc/%n", SYMLINK="%k"
KERNEL="vcsa", NAME="vcc/a0", SYMLINK="%k"
KERNEL="vcsa[0-9]*", NAME="vcc/a%n", SYMLINK="%k"

# v4l devices
KERNEL="video[0-9]*", NAME="v4l/video%n", SYMLINK="video%n" KERNEL="radio[0-9]*",
NAME="v4l/radio%n"
KERNEL="vbi[0-9]*", NAME="v4l/vbi%n", SYMLINK="vbi%n" KERNEL="vtx[0-9]*",
NAME="v4l/vtx%n"
```

19.8.4 Mécanisme d'udev.

19.8.4.1 Détection et attribution de nom.

Lorsqu'un périphérique est branché, le service hotplug en informe le noyau. Les données que le noyau a obtenues en interrogeant le matériel se retrouvent dans /sys (pseudo-système de fichiers sysfs). Le contenu de /sys est dynamique. Le noyau appelle la commande udevsend qui place l'événement dans une

file d'attente. Le démon udevd gère cette file d'attente et exécute les commandes selon un numéro d'ordre transmis par le noyau au cas où plusieurs périphériques seraient branchés simultanément. La commande udevsend reçoit du noyau un jeu de variables d'environnement et les informations relatives au périphérique branché (ajout, suppression ou ré-attribution de nom; établissement du chemin vers le périphérique dans /sys). udevd utilise alors ces données et consulte les fichiers /etc/udev/rules.d et /etc/udev/permissions.d pour déterminer le(s) fichier(s) de périphérique à créer, les droits à lui (ou leur) appliquer et les liens symboliques à établir.

udev garantit que le nom de fichier spécial affecté à un périphérique donné ne change pas et est explicite (voir la section 19.8.6). Ceci est vrai quelle que soit la manière et l'ordre dont les périphériques sont connectés au système. L'attribution explicite sur base des données matérielles empêche la confusion entre périphériques. La liste des paramètres utilisés pour lever les ambiguités de matériel peut être obtenue avec la commande systool (du paquet sysfsutils). Par exemple, si vous venez de brancher une clé USB (disque SCSI), vous pouvez exécuter :

```
systool -bv scsi
```

L'option -v active le mode babillard (autrement, vous n'obtiendrez qu'un résumé). L'option -b demande d'indiquer un bus spécifique.

19.8.4.2 D-BUS et HAL.

En plus d'udev, deux autres technologies ont fait leur apparition avec les noyaux-2.6 : D-BUS et HAL.

Lorsque le fichier de périphérique a été créé dans /dev (depuis l'espace utilisateur), l'utilisateur doit être averti que le nouveau périphérique est disponible. Le noyau poste un message à D-BUS (Data-BUS, un bus de messagerie utilisé pour échanger des messages entre applications), afin d'indiquer que le périphérique "untel" est présent sous la forme du fichier /dev/<nom_fichier>. Le programme qui écoute D-BUS s'appelle HAL (*Hardware Abstraction Layer*). Son but est de convertir /dev/<nom_fichier> (par exemple : /dev/lp0) en une expression plus intelligible. En outre, HAL est linguistiquement configurable.

A terme, le gestionnaire de volumes du système graphique utilisé pourra créer une icône[8] et ouvrir le périphérique tout en connaissant le type de données telles que collationnées par HAL.

19.8.4.3 Périphériques montables – /etc/fstab.

Le fichier /etc/fstab contient les entrées pour le montage des partitions (voir la section 20.7) et des périphériques montables (disquettes, CD-ROMs, clés-USB, disques amovibles, etc).

Lorsque HAL a obtenu les informations relatives au périphérique nouvellement branché, l'utilitaire fstab-sync est exécuté pour consulter /etc/fstab afin de le mettre à jour en écrivant éventuellement une ligne associée à ce nouveau périphérique. Par défaut, les nouveaux points de montage se trouvent sous

[8] Au moment de la rédaction de cette partie, c'était déjà le cas pour le bureau Gnome. Voir la sous-section 44.2.3 (et aussi la sous-section 44.2.2 pour KDE).

/media. Lorsque le périphérique est déconnecté, la ligne est supprimée. On a donc à faire à une table /etc/fstab dynamique. Etant donné que HAL est un service lancé sous root au démarrage, fstab-sync qui est exécuté par HAL possède les droits en écriture sur /etc/fstab.

fstab-sync utilise l'option sync. Aussi, les clés-USB peuvent-elles être retirées "à chaud" (en principe du moins) sans démontage explicite. Informé de cet événement, HAL éliminera dans /etc/fstab la ligne associée à la clé-USB et le gestionnaire de fichiers supprimera l'icône sur le bureau.

Cas des CD-ROMs et DVDs. Le montage des CD-ROMS et DVDs se fait avec un mécanisme de verrouillage. Lors du démontage automatique tel que décrit dans la sous-section 19.8.4.3, il faut déverrouiller le CD-ROM ou le DVD. Ceci est réalisé en mettant l'option :

```
dev.cdrom.lock = 0
```

dans le fichier /etc/sysctl.conf.

Cas des lecteurs de disquettes. Lorsqu'une disquette est insérée dans un lecteur, le système d'exploitation ne reçoit aucun message particulier. De même au démontage. Il vaut donc mieux effectuer les opérations de montage et de démontage manuellement (en cliquant sur l'icône affichée sur le bureau sous Gnome ou KDE, ou encore en utilisant le mécanisme de supermount qui ne sera pas vu ici).

19.8.5 Installation des services D-BUS et HAL.

D-BUS est une dépendance de HAL. Il suffit donc d'installer le paquet hal, soit avec la commande rpm (RedHat), urpmi (Mandriva), apt-get install (Debian) ou emerge (Gentoo). Après quoi, il faut initialiser le service :

```
/etc/rc.d/init/hald start
( /etc/rc.d/init.d/hald start )
```

19.8.6 Règles d'udev.

Note : cette partie est très largement inspirée du document Writing udev rules *de Daniel Drake sur http ://www.reactivated.net/udevrules.php.*

La rédaction de règles pour udev est optionnelle. Néanmoins, on peut personnaliser udev par convenance personnelle ou pour nommer des périphériques de manière persistante.

Prenons le cas de deux imprimantes (nommées pour l'exemple, *Laser* et *Jet_ encre*). Si *Laser* est d'abord branchée, un fichier /dev/lp0 lui est attribué et ensuite, *Jet_ encre* sera associée à /dev/lp1. Si on en venait à permuter les imprimantes, par exemple, suite à l'utilisation d'un connecteur "hub" et que nous utilisions des scripts faisant appel à /dev/lp0 en pensant avoit à faire à *Laser*, il y aurait confusion car, suite à la reconnexion, /dev/lp0 serait associé à *Jet_ encre*.

Au contraire, si *Laser* avait été affectée à /dev/lp_laser et *Jet_ encre* à /dev/lp_je, les scripts s'adresseraient toujours à la (ou aux) bonne(s) imprimante(s). Or, udev peut contrôler l'attribution de nom de périphérique et garantir que les *noms persistents* des fichiers de /dev pointent toujours vers les périphériques auxquels ces noms sont associés.

Par ailleurs, dans le cas des périphériques de masse (par exemple, les disques durs externes USB), l'*attribution de nom persitent* permet d'ajouter des lignes "en dur" dans /etc/fstab de manière automatique comme nous l'avons vu à la section 19.8.4.3.

19.8.6.1 Les bases.

Les règles d'udev étant dans /etc/udev/rules.d/50-udev.rules, il convient de ne pas écrire dans ce fichier pour éviter des problèmes résultant de mises à jour. Les fichiers de règles sont lus dans l'ordre croissant des nombres qui précèdent "-udev.rules". Il est impératif que les règles que vous écrivez soient lues avant celles par défaut. Par exemple, vous pouvez créer un fichier /etc/udev/-rules.d/10-udev.rules pour vos propres règles, si le fichier n'existe pas. Ce fichier sera lu avant le fichier par défaut et vos règles masqueront les règles par défaut qui créent la structure /dev de base. Etant donné que le style des règles d'udev est apparenté à celui de devfs, il est recommandé que vous écriviez vos règles en conservant un style "devfs" pour les noms de fichiers et de liens symboliques dans vos propres règles.

Dans tous les fichiers de règles, les lignes débutant par # sont commentées alors que toute ligne décommentée est interprétée comme une règle. La structure fondamentale d'une règle est :

```
key,[key,...] name [, symlink]
```

Voici le principe général :

1. Au minimum, une clef key doit être mentionnée. Une clef sert à identifier le périphérique auquel la règle s'applique.
2. Le paramètre NAME est obligatoire. Il indique à udev que ce périphérique doit être nommé comme dans l'arborescence de /dev. Le format employé est NAME="X" où X désigne le nom du fichier de périphérique (également appelé *noeud*). Vous pouvez indiquer plusieurs liens symboliques, mais ils doivent être séparés par un espace.
3. Le paramètre SYMLINK est optionnel. Il vous permet d'indiquer des emplacements supplémentaires où le fichier de périphérique sera lié. Notez qu'udev crée un fichier par périphérique (et un seul). Si vous souhaitez que ce périphérique soit accessible par différents noeuds, vous devrez indiquer les autres noeuds dans le paramètres SYMLINK.

Considérons un exemple un peu modifié d'une règle udev :

```
BUS="usb", SYSFS{serial}="HXOLL0012202323480", NAME="lp_epson",
SYMLINK="printers/epson_stylus"
```

Les clefs sont : BUS et SYSFS{serial}. udev appliquera cette règle au périphérique qui est connecté au bus USB *et* dont le numéro de série est HXOLL001220232-

3480. Notez qu'il faut une correspondance sur *toutes* les clefs pour qu'udev utilise la règle d'attribution de nom du périphérique.

Le fichier (noeud) sera nommé lp_epson et celui-ci pourra être trouvé en tant que /dev/lp_epson. udev créera aussi un lien symbolique printers/epson_-stylus dont le chemin absolu est /dev/printers/epson_stylus (s'il n'existe pas, le répertoire /dev/printers sera créé). Avec cette règle, vous pouvez imprimer en envoyant vos données sur /dev/printers/epson_stylus ou /dev/lp_epson.

19.8.6.2 Usage des opérateurs dans NAME et SYMLINK.

Il vous est possible d'utiliser des opérateurs élémentaires dans les paramètres NAME et SYMLINK afin de faciliter l'attribution de noms des périphériques. Cette méthode est connue sous le nom de *substitution de chaîne "à la" printf* (*printf-like string substitution*). Référez-vous au chapitre 23, relatif au langage **C** pour apprécier ce que l'instruction printf signifie. Voici un exemple :

```
BUS="usb", SYSFS{vendor}="FUJIFILM", SYSFS{model}="M100", NAME="camera%n"
```

A la lecture de la règle, l'opérateur %n sera remplacé par le nombre fourni par le noyau à propos du périphérique camera, de manière à produire un nom tel que CAMERA0, CAMERA1, etc.

%k est un autre opérateur courant. %k sera remplacé par le nom par défaut donné par le noyau au périphérique (par exemple, sda1). Vous verrez souvent des règles avec NAME="%k" (un exemple est donné dans la sous-section 19.8.6.3). La personnalisation se fait avec SYMLINK="%k".

La liste complète des opérateurs se trouve dans la page de man d'udev, mais %n et %k sont de loin les plus fréquents.

19.8.6.3 Usage des motifs de type shell dans les clefs.

Vous pouvez encore ajouter de la souplesse dans vos règles grâce aux motifs de type shell. Considérons la règle :

```
KERNEL="ts*", NAME="input/%k"
```

L'opérateur * a la même signification que dans le shell; il s'agit d'un caractère d'englobement (dont la value peut être : rien, 0, 1, ..., a, A, ..., toute chaîne de caractères).

Que dit cette règle ? Littéralement : "Associez un périphérique, identifié par un nom utilisé par le noyau, commençant par ts et eventuellement suivi par une chaîne quelconque de caractères ; ensuite, nommez-le avec le nom utilisé par le noyau (%k) sous le répertoire input."

L'opérateur ? substitue tout caractère unique (mais pas le caractère nul).

Les crochets [ ] peuvent également être utilisés pour remplacer tout caractère unique dans une gamme donnée. Par exemple, "tty[SR]" correspondra soit à "ttyS" ou à "ttyR". Vous pouvez aussi utiliser une expression comme [0-9] pour une correspondance sur une gamme de chiffres uniques. Par exemple :

```
KERNEL="fd[0-9]*", NAME="floppy/%n"
```

associe un périphérique, identifié par un nom utilisé par le noyau, commençant par `fd`, suivi par un chiffre de la gamme `0` à `9` et, éventuellement suivi par une chaîne quelconque de caractères (`*`) ; ensuite la règle nomme le périphérique avec le nombre donné par le noyau à ce périphérique (`%n`), sous le répertoire `floppy`.

Ces caractères de remplacement peuvent être utilisés avec n'importe quelle clef (clefs de base ou identification basée sur `sysfs` –voir ci-dessous). Pour aller plus loin avec les motifs basés sur `[ ]`, consultez `udev`(8).

19.8.6.4 Identifier les périphériques.

Il est possible de récupérer de l'information concernant les périphériques en utilisant les clefs mais aussi en consultant le pseudo-système de fichiers `/sys`. Une règle peut donc faire appel aux clefs usuelles (`BUS` ou `KERNEL`) mais aussi aux clefs de `sysfs` pour différencier les différents périphériques. `udev` fournit aussi des clefs appelant des scripts, mais ceci sort du cadre de cet ouvrage.

Comment connaître les numéros de série et les modèles de périphériques ? L'essentiel consiste à utiliser et à sélectionner l'information contenu dans `/sys`.

19.8.6.5 Identifier les périphériques grâce aux clefs de base.

En réalité, les clefs utilisables sont :
- BUS qui associe le type de bus du périphérique,
- KERNEL qui associe un nom de périphérique donné par le noyau,
- ID qui associe le numéro du périphérique sur le bus,
- PLACE qui associe la position physique occupée par le périphérique (utile dans le cas de l'USB).

Cependant, les clefs `ID` et `PLACE` sont rarement utilisées.

19.8.6.6 Identifier les périphériques grâce aux fichiers de /sys – udevinfo.

Le pseudo-système de fichiers `/sys` contient une arborescence de répertoires renfermant des information sur votre matériel. Les données sont typiquement le nom du périphérique, le fabricant et/ou le numéro identificateur.

Notez aussi que les clefs de type `SYSFS{...}` peuvent être combinées avec les clefs de bases.

Si vous avez branché un appareil photo USB, un fichier sera créé dans `/sys/block/sda/device/model/` contenant par exemple "USB 2.0M DSC". La clef SYSFS sera donc : `SYSFS{model} = "USB 2.0M DSC"`. [9]

Naturellement, l'établissement de règles ne se fait pas en parcourant manuellement les fichiers de `/sys`. C'est ici qu'intervient l'utilitaire `udevinfo` associé à votre paquet `udev`.

La première action à mener consiste à trouver un répertoire de `/sys` qui corresponde à votre périphérique (un fichier du type `/dev/...`). La recherche ne doit se faire que dans `/sys/block` ou `/sys/class`. Alors, `udevinfo` suivra les liens symboliques vers d'autres répertoires. Lorsque vous avez trouvé le répertoire adéquat, vous pouvez lancer `udevinfo`. Supposons que vous venez de

[9] Notez que tout fichier dans `/sys` peut être l'objet d'une correspondance selon cette méthode. Cependant, si vous établissez une correspondance sur plus d'un fichier en utilisant des clefs multiples, vous ne pourrez effectuer des correspondances que dans un même répertoire alors qu'il est fréquent que plusieurs répertoires soient dévolus à un périphérique.

brancher une clé USB. Le noeud `/dev/sda` est créé. Exécutez :

```
udevinfo -q path -n /dev/sda
```

qui vous retourne :

```
/block/sda
```

Ceci indique que le chemin de `sysfs` est `/sys/block/sda`. Vous pouvez lancer la commande :

```
udevinfo -a -p /sys/block/sda
```

mais en général, les deux commandes précédantes peuvent être combinées :

```
udevinfo -a -p ‘udevinfo -q path -n /dev/sda‘
```

Les informations que vous allez recueillir devraient ressembler à ceci :

```
follow the class device's "device"
looking at the device chain at '/sys/devices/pci0000 :00/0000 :00 :02.1 \
/usb3/3-3/3-3 :1.0/host0/0 :0 :0 :0' :
  BUS="scsi" ID="0 :0 :0 :0"
  SYSFS{detach_state}="0"
  SYSFS{type}="0"
  SYSFS{max_sectors}="240"
  SYSFS{device_blocked}="0"
  SYSFS{queue_depth}="1"
  SYSFS{scsi_level}="3"
  SYSFS{vendor}=" "
  SYSFS{model}="USB 2.0M DSC "
  SYSFS{rev}="1.00"
  SYSFS{online}="1"

looking at the device chain at '/sys/devices/pci0000 :00/0000 :00 :02.1/usb3/3-3' :
  BUS="usb"
  ID="3-3"
  SYSFS{detach_state}="0"
  SYSFS{bNumInterfaces}=" 1"
  SYSFS{bConfigurationValue}="1"
  SYSFS{bmAttributes}="c0"
  SYSFS{bMaxPower}=" 0mA"
  SYSFS{idVendor}="052b"
  SYSFS{idProduct}="1514"
  SYSFS{bcdDevice}="0100"
  SYSFS{bDeviceClass}="00"
  SYSFS{bDeviceSubClass}="00"
  SYSFS{bDeviceProtocol}="00"
```

```
  SYSFS{bNumConfigurations}="1"
  SYSFS{speed}="12"
  SYSFS{manufacturer}="Tekom Technologies, Inc"
  SYSFS{product}="USB 2.0M DSC"
```

A ce stade, il est important de constater qu'`udevinfo` a retourné deux types d'informations différentes représentées dans l'encadré avec deux types de polices de caractères pour bien les différencier. La première sources d'information provient de `/sys/devices/pci0000 :00/0000 :00 :02.1/usb3/3-3/3-3 :1.0/host-0/0 :0 :0 :0`, l'autre, de /sys/devices/pci0000 :00/0000 :00 :02.1/usb3/3-3. En effet, les règles ne fonctionnent que si nous ne mélangeons pas des informations de type différents. Par exemple, la règle suivantes ne fonctionnera pas :

```
BUS="scsi", SYSFS{manufacturer}="Tekom Technologies, Inc", NAME="%k"
```

En revanche, la règle suivante est opérationnelle :

```
BUS="usb", SYSFS{manufacturer}="Tekom Technologies, Inc", NAME="%k"
```

19.8.6.7 Exemple 1 : règle pour une imprimante USB.

Après avoir branché une imprimante USB, nous effectuons une recherche pour ce périphérique dans `/sys`. Supposons que le périphérique a reçu un fichier `/dev/lp0`. La commande :

```
udevinfo -q path -n /dev/lp0
```

donne comme résultat :

```
/class/usb/lp0
```

La seconde commande :

```
udevinfo -a -p /sys/class/usb/lp0
```

donne :

```
looking at the device chain at '/sys/devices/pci0000 :00/0000 :00 :02.1 \
/usb3/3-3' :
  BUS="usb"
  SYSFS{manufacturer}="EPSON"
  SYSFS{product}="USB Printer"
  SYSFS{serial}="L72010011070626380"
```

La règle peut donc être formulée ainsi :

```
BUS="usb", SYSFS{serial}="L72010011070626380", NAME="%k",
SYMLINK="epson_680"
```

Naturellement, le fichier de périphérique /dev/lp0 existera toujours mais /dev/-epson_680 pointera toujours vers l'imprimante qui vient d'être branchée, même si celle-ci est débranchée et reconnectée après qu'une autre imprimante ait été connectée.

19.8.6.8 Exemple 2 : règle pour un appareil photo numérique USB.

Supposons que nous ayons un appareil photo numérique se comportant comme un disque SCSI externe et exploitant le pilote du noyau usb-storage qui est également utilisé par une carte flash et un disque USB externe. Nous voudrions monter la partition associée au disque de l'appareil numérique et tranférer son contenu. Notez qu'usuellement l'extraction de photos requiert un logiciel approprié comme digikam (basé sur gphoto2 en ligne de commande) ou flphoto.

Le point majeur est que, vu la création de plusieurs fichiers (/dev/sda, /dev/sda1 et peut-être /dev/sg1), vous devrez créer une règle suffisamment spécifique pour ne pas avoir une correspondance avec ces trois fichiers.

/dev/sda1 est le noeud que nous voudrions utiliser avec un lien symbolique /dev/camera. Dans ce cas, udevinfo ne donnant pas de différences significatives concernant les fichiers /dev/sda, /dev/sda1 et /dev/sg1, il est utile de considérer le nom fourni par le noyau (clef de base : KERNEL).

Une clef comme KERNEL=''sd ?1'' est intéressante car elle établit une correspondance sur sda1, sdb1, sdc1, etc., mais pas –et c'est le plus important– sur sda, sdb, ou encore sg1.

A présent, comme sda1 est un périphérique bloc, il est logique de parcourir /sys/block/ et de fait, il y a un fichier sda1 dans /sys/block/sda/. Etant donné que ces deux derniers répertoires possèdent des fichiers dev, nous pouvons utiliser udevinfo :

```
udevinfo -a -p /sys/block/sda/sda1
```

L'information cruciale obtenue avec cette commande est :

```
SYSFS{product}="USB 2.0M DSC"
```

La règle est donc :

```
BUS="usb", SYSFS{product}="USB 2.0M DSC", KERNEL="sd?1", NAME="%k",
SYMLINK="camera"
```

L'appareil photo sera donc associé au fichier /dev/dsa1 mais aussi au noeud plus explicite /dev/camera. (Si /dev/sda1 n'est pas disponible, /dev/sdb1 sera utilisé, voire /dev/sdc1 si /dev/sdb1 n'est pas non plus disponible, et ainsi de suite.) Bien sûr, /dev/sda sera créé. Toutefois, ce qui est important c'est que le lien symbolique persistent nommé camera pointera toujours vers la partition montable /dev/sda1.

19.8.6.9 Supplément pour les périphériques de masse USB.

Dans le cas des disques externes USB, il est utile d'utiliser les commandes `fdisk` (commande de réparation d'un système de fichier ou de partitionnement, voir 20.5) ou `hdparm` (commande d'optimisation d'accès aux disques IDE, voir `hdparm`(8)). Dans ce cas, l'usage de `/dev/sda` est fondamental.

Une règle comme celle-ci :

```
BUS="usb", KERNEL="sd*", SYSFS{product}="USB 2.0 Storage Device",
NAME="%k", SYMLINK="usbhd%n"
```

permet de créeer les liens symboliques :

/dev/usbhd : fichier sur lequel `fdisk` et/ou `hdparm` sont applicables,

/dev/usbhd1 : première partition montable,

/dev/usbhd2 : seconde partition montable.

Un autre cas compliqué concerne les lecteurs de cartes mémoire USB à *slots* multiples. En général, lorsqu'une carte est introduite alors que le lecteur est branché, aucun message n'est envoyé au noyau si bien qu'aucun nouveau fichier de périphérique n'est créé pour le montage.[10]

`udev` apporte une solution en créant des fichiers pour toutes les partitions du périphérique bloc. Pour une règle donnée, 16 fichiers de partition seront crées. Il suffit pour cela de modifier la clef NAME de cette façon :

```
BUS="usb", SYSFS{product}="USB 2.0 Storage Device",
NAME{all_partitions}="usbhd"
```

Les fichiers `usbhd`, `usbhd1`, `usbhd2`, ..., `usbhd15` seront crées.

19.8.6.10 Exemple 3 : règle de convenance pour lecteur CD.

A présent, supposons que vous ayez deux lecteurs CD : un lecteur DVD (`hdc`, disons) et un graveur CD-ROM (`hdd`, disons). L'attribution des fichiers `hdc` et `hdd` ne devrait pas changer à moins de permuter les nappes.

Cependant, il est plus pratique d'avoir à faire à `/dev/dvd` et `/dev/cdrw`. Par avance, nous connaissons la valeur d'`X` dans `hdX`. Les règles sont donc :

```
BUS="ide", KERNEL="hdc", NAME="%k", SYMLINK="dvd cdroms/cdrom%n"
BUS="ide", KERNEL="hdd", NAME="%k", SYMLINK="cdrw cdroms/cdrom%n"
```

Comme ces règles apparaissent dans un fichier lu avant que `50-udev.rules` ne soit lu lui-même (supposons que vous ayez appelé le fichier contenant vos propres règles : `10-udev.rules`), les règles d'attribution de noms des périphériques blocs contenues dans `50-udev.rules` n'auront pas la préseance sur les vôtres.

[10] C'est également vrai pour les disques USB. Lorsqu'une nouvelle partition est créée, le nouveau fichier associé à cette partition n'est pas créé tant que le périphérique n'est pas rebranché.

19.8.6.11 Exemple 4 : règle d'attribution de noms d'une interface réseau.

Les nouvelles versions d'udev permettent d'écrire des règles d'attribution de noms pour les interfaces réseau. Ces dernières n'apparaissent pas dans /dev. Vous procédez à leur configuration et à la lecture de celle-ci au moyen de la commande ifconfig (voir le chapitre 26). Néanmoins, les règles d'attribution de noms sont rédigées de manière classique. Renommons notre carte eth0. A nouveau, udevinfo vous vient en aide :

```
udevinfo -a -p /sys/class/net/eth0/
```

Le retour de cette commande (quelque peu toiletté) est :

```
looking at class device '/sys/class/net/eth0' :
SYSFS{address}="00 :06 :5b :02 :17 :8a"
```

Etant donné que chaque carte réseau possède une adresse MAC unique apportée à sa fabrication, il est pertinent de s'en servir pour établir la règle. A moins que vous ne changiez de carte, la règle restera valable. En voici un exemple typique :

```
KERNEL="eth*", SYSFS{address}="00 :52 :8b :d5 :04 :48", NAME="lan"
```

Pour que cette règle prenne effet, il faudra soit réamorcer votre système, soit décharger puis recharger le pilote de la carte réseau. Il faudra également reconfigurer la carte pour lui donner le nom lan au lieu d'eth0. Pour être sûr que cela fonctionne, il est possible qu'il faille supprimer toutes les références à eth0 au préalable.

19.8.6.12 Règles basées sur le paramètre SYMLINK.

Avec les versions récentes d'udev, il est possible de rédiger des règles qui ne spécifient pas de clefs NAME, mais qui incluent simplement des clefs SYMLINK. Ceci vous évite décrire des règles qui masqueraient les règles par défaut d'udev.

Prenons le cas de la règle suivante :

```
KERNEL="hdc", SYMLINK="dvd"
```

Lorsqu'udev la lit, il la mémorise. Lorsqu'il lit une autre règle associée au même périphérique (mais qui comprend une clef NAME), udev crée le fichier tel qu'indiqué par la règle comprenant la clef NAME et, en plus, les liens symboliques définis par les clefs SYMLINK des deux règles.

En l'occurrence, l'attribution de nom du périphérique hdc se passe selon les règles par défaut applicables les périphériques blocs, mais un lien symbolique dvd sera aussi créé compte tenu de la règle personnelle citée ci-dessus.

Bien sûr, udev ne tient compte de votre règle personnelle que si elle peut être lue avant que les règles par défaut ne le soient. Consultez udev(8).

Chapitre 20

Partitions, systèmes de fichiers, formatage et montage.

20.1 Structure physique des disques.

Les disques sont divisés en partitions [voir `/dev/hd??` dans la section 19.4]. L'information relative au partionnement d'un disque est enregistrée dans une *table de partition* qui est une petite zone du disque séparée des partitions elles-mêmes.

20.1.1 Cylindres, têtes et secteurs.

Un disque dur est en fait constitué de plusieurs disques dont les deux faces sont utilisées. Ces dernières, numérotées 0, 1, 2, etc., sont également appelées *têtes* (*heads*) parce qu'une tête magnétique y effectue les opérations de lecture et d'écriture. Chaque tête est constituée de *pistes* (*tracks*) elles-mêmes divisées en segments appelés *secteurs* (*sectors*). Chaque secteur contient typiquement 512 octets. La quantité totale d'espace sur un disque, exprimée en octets est donc :

$$512 \times (\textit{nombre de secteurs par pistes}) \times$$

$$(\textit{nombre de pistes par tetes}) \times (\textit{nombre de tetes})$$

Une simple piste mais aussi l'empilement de toutes les pistes de même diamètre (sur touts les disques individuels du disque dur) forment un *cylindre*. On parle donc des disques en mentionant le nombre de "cylindres et de secteurs" au lieu du nombre de "têtes, pistes ou secteurs". Les partitions sont usuellement délimitées par les bords des cylindres. Donc, les disques ne peuvent contenir de partitions de taille arbitraire : la taille d'une partition est un multiple de la quantité de données contenue dans un cylindre. Par conséquent, les partitions présentent des diamètres interne et externe bien définis. La figure 20.1 illustre la structure générale d'un disque dur.

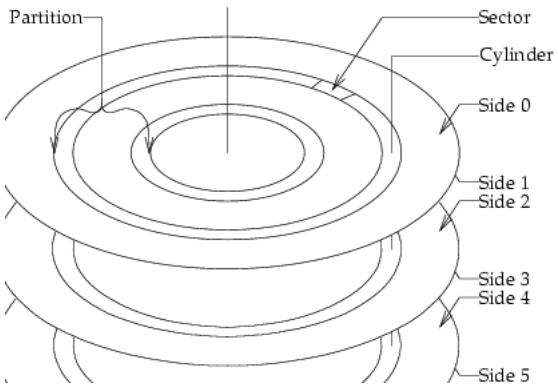


FIG. 20.1 – Représentation simplifiée d'un disque dur et de la dispostion des secteurs.

20.1.2 Adressage de blocs de grande taille ("mode LBA").

Ce qui est décrit dans la section précédente est correct si ce n'est concernant une limite apparemment curieuse : les tables de partitions possèdent seulement 10 bits dans lesquels est enregistrée la disposition des cylindres (en anglais : *partition's cylinder offset*). Ceci signifie qu'aucun disque ne peut avoir plus de 1024 cylindres ($\underbrace{2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2}$). Cette limitation est contournée de manière logicielle en multipliant le nombre de têtes de manière à réduire le nombre de cylindres [mode LBA (pour *Large Block Addressing)*]. En pratique, l'utilisateur ne doit jamais se soucier de la manière dont son disque dur est réorganisé par le mode LBA.

20.1.3 Partitions étendues.

La table des partitions contient quatre partitions, appelées *partitions primaires*. S'il faut davantage de partitions, une de ces partitions primaires doit être divisée en partitions de plus petite taille, appelées *partitions logiques*. S'il est envisagé qu'une partition primaire (*primary partition*) soit subdivisée de manière à contenir des partitions logiques, il est convenu de l'appeler *partition primaire étendue* ou *partition étendue* (*extended primary partition*). Typiquement, la première partition sera de petite taille (disons qu'il s'agit de `/dev/hda1`). La seconde partition primaire (`/dev/hda2`) occupera le reste du disque. Dans ce cas, les entrées de la table de partition (`/dev/hda3` et `/dev/hda4`) seront vides. La partition étendue pourra alors être subdivisée pour accueillir `/dev/hda5`, `/dev/hda6`, etc.

20.2 Partitionnement d'un nouveau disque.

Un disque neuf n'est jamais formaté. En tapant `fdisk`, vous lancez un utilitaire interactif de partitionnement. La commande :

```
fdisk /dev/hda
```

`partitionnera` votre premier maître.

Ce qui suit constitue un exemple de partitionnement d'un nouveau disque dur. La plupart des distributions présentent des outils graphiques pour partitionner des disques, ce qui fait que la commande `fdisk` n'est pas nécessaire lors d'une installation. Cependant, l'addition d'un nouveau disque ou les opérations de transfert/copie d'un système LINUX vers un nouveau disque requerront un partitionnement.

Sous UNIX, chaque partition possède son propre *répertoire. Les fichiers présents dans un répertoire peuvent être répartis sur un disque différent ou sur une partition différente par rapport aux fichiers d'un autre répertoire.* Typiquement, le répertoire `/var` et les sous-répertoires qui y sont associés, sont enregistrés sur une partition différente du répertoire `/usr` (et de tous les sous-répertoires qui y sont associés).

Le tableau 20.1 présente des conseils généraux quant à la manière dont un serveur devrait être monté (avec les ordinateurs familiaux, vous pouvez prendre davantage de liberté; la plupart des PCs familiaux se satisfont d'une partition d'échange ou *swap*, d'une partition `/` et d'une partition `/home`).

Lorsque vous installez un serveur, votre distribution devrait vous permettre de personnaliser vos partitions pour correspondre au tableau 20.1.

Si un autre système d'exploitation est déjà présent sur la première partition, vous pouvez taper `p` :

```
Command (m for help) : p

Disk /dev/hda : 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes

   Device  Boot     Start      End   Blocks   Id   System
/dev/hda1               1      312  2506108+  c   Win95 FAT32 (LBA)
```

Dans cette situation, vous pourrez ajouter d'autres partitions.

La même méthode vaut dans le cas des disques SCSI. La seule différence est que `/dev/hd?` est remplacé par `/dev/sd?` (voir la section 43.6.9 pour les informations sur les pilotes de périphériques SCSI).

Voici une session de partitionnement sur un disque vide avec la commande `fdisk` :

```
[root@cericon /root]# fdisk /dev/hda
Devices contains neither a valid DOS partition table, nor Sun or SGI disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.
```

Avant tout, nous utilisons l'option `p` afin d'afficher la table des partitions courante (voir l'encadré suivant le tableau 20.1).

TAB. 20.1 – Répertoires et tailles de partitons (cas d'un serveur).

Répertoire	Taille (Mo)	Motivations.
swap	2 fois la RAM	C'est ici que la mémoire est étendue. La swap donne l'impression que vous avez plus de RAM qu'enréalité, par déplacement de données. L'accès au disque est évidemment lent par rapport à l'accès en RAM, mais lorsqu'il y a de nombreux programmes en attente, leur déplacement en swap permet d'alléger la RAM pour les programmes requièrent l'usage de cette dernière.
/boot	5-10	Ce répertoire ne doit pas forcément se trouver sur une autre partition de / (voir ci-dessous). Quoique vous choisissiez, il ne doit y avoir aucune possibilité qu'un fichier sous /boot puisse s'étendre sur des secteurs se trouvant au-delà de la limite des 1024 cylindres (c'est-à-dire au-delà des 500 premiers Mo de votre disque dur). C'est pourquoi /boot (ou /) sont souvent placés sur la première partition primaire du disque dur. Si cette contrainte n'est pas observée, vous obtiendrez la fameuse invite LI d'un système non-amorçable. Voir la section 32.2.4.
/var	100-1000	Ici se trouvent les données variables, comme les journaux (*logs*), les fichiers d'attente du courriel (*mail spool files*), les fichiers de base de données et votre fichier *cache* mandataire web (*web proxy cache*) Ce fichier et les bases de données peuvent nécessiter un espace *beaucoup* plus grand, cependant). Dans le cas des distributions récentes, ce répertoire contient aussi toute donnée locale que la machine sert (fichiers FTP ou pages web). Si vous envisagez d'utiliser un fichier de *cache* web, enregistrez les informations dans une partition ou un disque séparés, ou prévoyez une très grande partition /var. Notez que la taille des fichiers de logs peut augmenter de manière considérable lorsque des problèmes surgissent. Vous ne désirerez pas qu'une partition pleine ou corrompue /var affecte le reste de votre disque. D'où une partition dédiée.
/tmp	50	C'est ici que se siègent les données temporaires. Les programmes accèdent fréquemmement à ce répertoire et avec rapidité. Il est utile de faire une partition séparée car, régulièrement, les programmes doivent *réellement* créer un ficher temporaire, et cette pratique ne doit pas être entravée par le remplissage d'autres partitions. Cette partition est aussi celle qui a le plus de chance de se corrompre.
/usr	500-1500	C'est ici que votre distribution (Debian, Gentoo, RedHat, Mandriva, ...) se trouve. Elle est montée en lecture seule. Si vous avez un disque dont l'accès en écriture peut être désactivé physiquement (comme certains SCSI), vous pouvez mettre /usr sur un disque séparé. Ceci rendra votre système plus sûr.
/home	Reste du disque	Les répertoires /home des utilisateurs sont logés ici.
/	50-100	Tout ce qui n'est pas dans les autres répertoires se trouve sous /. Ce sont /bin (5Mo), (éventuellement) /boot (3Mo), /dev(0,1 Mo), /etc(4Mo), /lib(20 Mo), /mnt(0 Mo), /proc(0 Mo) and /sbin(4 Mo). Ces répertoires sont essentiels au démarrage du système et contiennent des utilitaires minimaux pour récupérer les autres partitions en cas d'urgence. Comme décrit ci-dessus, si /boot est dans une partition séparé, la racine (/) doit se trouver sous la limite des 1024 cylindres (c'est-à-dire dans les 500 premiers Mo de votre disque dur).

```
Command (m for help) : p

Disk /dev/hda : 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes

   Device  Boot     Start      End   Blocks   Id   System
```

Nous voyons qu'il n'y a pas encore eu de partionnement sur `/dev/hda`. A présent, ajoutons une nouvelle partition (introduisons d'abord `n` puis `p`) :

```
Command (m for help) : n
Command action
   e  extended
   p  primary partition (1-4)
p
```

Nous voulons définir la première partition physique en la faisant commencer au début du premier cylindre :

```
Partition number (1-4) : 1
First cylinder (1-788, default 1) : 1
```

Nous souhaitons définir une partition de 80 Mo. `fdisk` calcule le dernier cylindre de manière automatique :

```
Last cylinder or +size or +sizeM or +sizeK (1-788, default 788) : +80M
```

Notre partition suivante occupera le reste du disque et sera de type "étendue" :

```
Command (m for help) : n
Command action
   e  extended
   p  primary partition (1-4)
e
Partition number (1-4) : 2
First cylinder (12-788, default 12) : 12
Last cylinder or +size or + sizeM or +sizeK (12-788, default) : 788
```

Nos partitions logiques correspondent à la partition étendue :

```
Command (m for help) : n
Command action
   l  logical (5 or ever)
   p  primary partition (1-4)
l
First cylinder (12-788, default 12) : 12
Last cylinder or +size or + sizeM or +sizeK (12-788, default 788) : +64M

Command (m for help) : n
Command action
   l  logical (5 or ever)
```

```
   p  primary partition (1-4)
l
First cylinder (21-788, default 21) : 21
Last cylinder or +size or + sizeM or +sizeK (21-788, default 788) : +100M

Command (m for help) :n
Command action
   l  logical (5 or ever)
   p  primary partition (1-4)
l
First cylinder (34-788, default 34) : 34
Last cylinder or +size or + sizeM or +sizeK (34-788, default 788) : +200M

Command (m for help) : n
Command action
   l  logical (5 or ever)
   p  primary partition (1-4)
l
First cylinder (60-788, default 60) : 60
Last cylinder or +size or + sizeM or +sizeK (60-788, default 788) : +1500M

Command (m for help) : n
Command action
   l  logical (5 or ever)
   p  primary partition (1-4)
l
First cylinder (252-788, default 252) : 252
Last cylinder or +size or + sizeM or +sizeK (252-788, default 788) : +788M
```

Le *type de partition* par défaut (*partition type*) consiste en un octet unique que le système d'exploitation prend en compte pour déterminer le type de fichier. Entrer la lettre `L` fournit les types connus :

```
Command (m for help) : L

0  Empty           16 Hidden FAT16    61 SpeedStor     a6 OpenBSD  [...]
8  AIX             4d QNX4.x          82 Linux swap    db CP/M /  CTOS /  .
9  AIX bootable    4e QNX4.x 2nd part 83 Linux         e1 DOS access [...]
12 Compaq diagnostic 56 Golden Bow     a5 BSD/386       ff BBT
14 Hidden FAT16 <3  5c Priam Edisk
```

Par défaut, `fdisk` fixe le type à `Linux`. Nous devons seulement déclarer explicitement le type de la partition swap :

```
Command (m for help) : t
Partition number (1-9) : 5
Hex code (type L to list codes) : 82
Changed system type of partition 5 to 82 (Linux swap)
```

A ce stade, nous devons activer l'indicateur "bootable" sur la première partition, vu que les sémaphores (*flags*) du BIOS ne permettront pas de démarrer un disque sans au moins une partition d'amorçage :

```
Command (m for help) : a
Partition number (1-10) : 1
```

Pour résumer le partionnement :

```
Command (m for help) : p

Disk /dev/hda : 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes

 Devices  Boot  Start    End      Blocks   Id  System
/dev/hda1    *       1     11       88326   83  Linux
/dev/hda2           12    788     6241252+   5  Extended
/dev/hda3           12     20       72261   82  Linux swap
/dev/hda4           21     33      104391   83  Linux
/dev/hda5           34     59      208813+  83  Linux
/dev/hda6           60    251     1542208+  83  Linux
/dev/hda7          252    788     4313421   83  Linux
```

A ce stade, rien n'a encore été écrit sur le disque. Nous pratiquons de la manière suivante (sachant que **cette étape est irréversible**) :

```
Command (m for help) : w
The partition table has been altered!

Calling iostl() to re-read partition table.
Syncing disks.

WARNING : If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
```

Même après avoir écrit (*written*) la table de partition, `fdisk` peut donner un avertissement au sujet du fait que le noyau ne connait pas l'existence des nouvelles partitions. Ceci arrive lorsque le disque est déjà en activité. Dans ce cas, vous devez relancer le système. Pour le partitionnement précédent, le noyau émettra l'information que voici lors du ré-amorçage de la machine (*reboot*) :

```
Partition check :
hda : hda1 hda2 < hda5 hda6 hda7 hda8 hda9 >
```

L'expression `<...>` indique que la partition `hda2` est étendue et subdivisée en cinq partitions de plus petite taille.

20.3 Formater les périphériques.

20.3.1 Systèmes de fichiers.

Un disque est usuellement lu par blocs de 1024 octets (c'est-à-dire par groupe de 2 secteurs). Du point de vue de tout qui accède au périphérique, les blocs sont stockés consécutivement (il ne faut donc pas penser en termes de cylindres ou de têtes) si bien que tout programme peut lire le disque comme s'il s'agissait d'une bande linéaire. Essayez :

```
less /dev/hda1
less -f /dev/hda1
```

Maintenant, une structure de répertoires complexe avec de nombreux fichiers de taille arbitraire doit être enregistrée dans cette partition contiguë (en l'occurrence `/dev/hda1`). Il faut donc savoir que faire d'un fichier effacé qui laisse un "trou" parmi les données stockées dans la partition, ou d'un fichier qui a été divisé en plusieurs parties car il n'y a pas d'espace contigu suffisamment grand pour le contenir. Les fichiers doivent aussi être indicés de manière telle qu'ils soient retrouvés rapidement (considérez la situation de 10.000 fichiers ou plus sur un système). Les liens physiques ou symboliques de UNIX et les fichiers de périphériques doivent également être enregistrés.

Pour traiter une telle complexité, les sytèmes d'exploitation ont une méthode pour héberger les fichiers. Celle-ci est désignée sous le vocabulaire de *système de fichiers* (ou *file system* ; `fs`). Comme MS-DOS avec son fichier FAT (c'est-à-dire FAT16) ou MS-Windows avec le fichier VFAT (ou FAT32)[1], LINUX possède son propre système de fichier appelé *second système de fichiers étendu* (*2nd extended file system*), désigné sous le nom d'`ext2`.

En fait, `ext2` est le système de fichiers natif traditionnel de LINUX. Cependant, trois autres systèmes de fichiers natifs sont devenus disponibles récemment : le système `XFS` de SGI, le système `ext3fs`, et le système de fichiers `reiserfs`. Ceux-ci permettent la récupération rapide et fiable de fichiers suite à des événements comme l'arrêt brutal de l'alimentation électrique, et ce, en utilisant un procédé de *journalisation* (*journaling*). Un système de fichiers journalisé pré-écrit les altérations que subissent les disques dans un journal (*log*) afin de faciliter la récupération au cas où le système de fichiers atteint un état incohérent. (Voir la section 20.5).

20.3.2 `mke2fs`.

Pour créer un système de fichiers sur une partition vierge, utilisez la commande `mkfs` (ou une de ses variantes). Ainsi, pour créer un système de fichiers `ext2` de LINUX sur votre premier maître, exécutez :

```
mkfs -t ext2 -c /dev/hda1
```

ou de manière alternative,

```
mke2fs -c /dev/hda1
```

L'option `-c` conduit à une vérification des blocs défectueux (*bad blocks*) par lecture du disque entier au préalable. Cette vérification se fait en mode lecture-seule et marque les blocs altérés de manière à ce qu'il ne soient plus utilisés. Si vous voulez faire une vérification en lecture-écriture, utilisez la commande `badblocks`. Cette dernière écrit et vérifie chaque octet dans la partition testée. Bien que l'option `-c` doive toujours être utilisée dans le cas d'un nouveau disque, réaliser un test en lecture-écriture total est quelque peu excessif. Pour la parti-

[1] NdT : et plus récemment NTFS.

tion précitée, la commande de ce test aurait la forme suivante :

```
badblocks -o liste-de-blocs.txt -s -w /dev/hda1 88326
mke2fs -l liste-de-blocs.txt /dev/hda1
```

Après avoir exécuté `mke2fs`, nous trouvons que la commande :

```
dd if=/dev/hda1 count=8 bs=1024 | file -
```

retourne `Linux/i386 ext2 filesystem`.

20.3.3 Formatage de disquettes et périphériques amovibles.

De nouveaux types de périphériques amovibles sont commercialisés en permanence. Quel que soit le périphérique, la même méthode de formatage est utilisée. Pour la plupart, ces périphériques sont compatibles IDE, ce qui signifie que vous pouvez y accéder via `/dev/hd?`.

Les exemples suivant s'adressent respectivement à un disque IDE à port parallèle, à un lecteur CD-ROM ATAPI à port parallèle, à un disque ATAPI à port parallèle et à votre lecteur de disquettes ("A :" pour les utilisateurs de MS-Windows) :

```
mke2fs -c /dev/pda1
mke2fs -c /dev/pcd0
mke2fs -c /dev/pf0
mke2fs -c/dev/fd0
```

En réalité, l'usage d'un système de fichiers de type `ext2` sur une disquette prend beaucoup de place. Préférez-lui un système de fichiers FAT ou VFAT, qui possède moins d'en-têtes tout en présentant l'avantage d'être lu par de très nombreux utilisateurs (voir la section 20.3.4).

Souvent, vous n'irez pas jusqu'à partitioner un périphérique susceptible de ne contenir qu'une seule partition. Vous utiliserez l'entièreté du disque comme partition unique. C'est le cas, par exemple, d'un lecteur IDE comme premier esclave [les lecteurs LS120 et Jazz sont des exemples commerciaux] :

```
mke2fs -c /dev/hdb
```

20.3.4 Formatage de disquettes MS-DOS.

L'accès aux fichiers des disquettes de type MS-DOS/Windows est expliqué dans la section 20.3.4. La commande `mformat A :` formatera une disquette, mais en initialisant seulement le système de fichiers. Elle ne vérifie pas la présence de blocs défectueux , ni ne fait le formatage de bas-niveau pour reformater des disquettes de tailles spéciales.

Une commande, appelée `superformat` et provenant du paquet `fdutils` [vous trouverez ce paquet sur internet ; voir le chapitre 25 pour apprendre à compiler et installer des sources], effectue des formatages dans tous les modes possibles. Plus commune (mais moins puissante) `fdformat` appartient au paquet `util-linux`.

Elle vérifie que chaque piste fonctionne correctement et compense les variations d'ordre mécanique des différents lecteurs de disquette. Pour formater une disquette de 3.5" à 1440 ko, à 1620 ko ou à 1920 ko (respectivement), exécutez :

```
cd /dev
./MAKEDEV -v fd0
superformat /dev/fd0H1440
superformat /dev/fd0H1690
superformat /dev/fd0H1920
```

Notez que ces disquettes ont des noms de fichiers longs (*long file name* ; VFAT) mais pas comme dans le cas des anciennes disquettes MS-DOS à 13 caractères.

La plupart des utilisateurs n'emploieront que des disquette de 3.5" à 1.44 Mo. En fait, le support "disque" et les têtes magnétiques peuvent écrire de manière plus dense que cela, permettant à 24 secteurs par pistes de faire du stockage au lieu de 18 secteurs, usuellement. Ceci explique pourquoi il y a plus d'un fichier de périphérique pour le même périphérique dans le répertoire `/dev`. Des disques de qualité inférieure présenteront des erreurs lors d'un stockage à plus grande densité. La commande `superformat` affiche ces erreurs.

Consultez la table 19.1 de la page 166 pour les conventions de dénomination des lecteurs de disquettes et de leur format respectif.

20.3.5 `mkswap`, `swapon` et `swapoff`.

La commande `mkswap` formate une partition utilisée comme périphérique "swap". Pour notre disque, il convient d'effectuer :

```
mkswap -c /dev/hda5
```

Notez que l'option `-c` a la même signification que précédemment (c'est-à-dire, vérifier les blocs défectueux).

Une fois la partition formatée, le noyau doit être informé qu'il peut l'utiliser comme partition de dégagement. Cela se fait à l'aide de la commande suivante :

```
swapon /dev/hda5
```

L'opération inverse est réalisée par la commande :

```
swapoff /dev/hda5
```

Naguère, les partitions de dégagement ne devaient pas être plus grandes que 128 Mo, bien qu'il était permis de les multiplier. Cette limitation ne concerne plus les noyaux récents. Grâce à `swapon`, vous pouvez activer plusieurs partitions de dégagement (*swap*) simultanément.

20.4 Opérations de montage.

Dans cette section, nous expliquons comment accéder à des fichiers sur un disque donné (cela va de soi, sans utiliser la notation `C :`, `D :`, etc.).

Sous UNIX, il y a un seul système de fichiers "racine" qui s'étend éventuellement sur plusieurs disques. Les différents répertoires peuvent exister sur des disques physiques différents.

L'opération de montage (mounting) permet de lier un répertoire à un périphérique physique (comme une partition ou un CD-ROM) de manière à lire le système de fichiers du périphérique.

La commande mount est utilisée de cette manière :

```
mount [-t <système_de_fichier>] [-o <option>] <périphérique>  <répertoire>
umount [-f] [<périphérique>|<répertoire>]
```

L'option -t spécifie le type de système de fichiers et peut souvent être omise puisque LINUX est capable d'autodétecter la plupart des systèmes de fichiers. <système_de_fichier> peut être du type : adfs, affs, autofs, coda, coherent, devpts, efs, ext2, hfs, hpfs, iso9660, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, romfs, smbfs, sysv, ufs, umsdos, vfat, xenix ou xiafs. Les systèmes de fichiers les plus communs sont discutés ci-dessous. L'option -o n'est normalement pas utilisée. Consultez mount(8) pour toutes les options possibles.

20.4.1 Montage de CD-ROMs.

Placez le disque CD-ROM de votre distribution dans le lecteur adéquat et montez-le ainsi :

```
ls /mnt/cdrom
mount -t iso9660 -o ro /dev/hdb /mnt/cdrom
```

Votre CD-ROM peut être désigné comme /dev/hdc ou /dev/hdd (dans ce cas, vous devrez établir un lien symbolique /dev/cdrom pointant vers le périphérique adéquat). Votre distribution peut préférer la dénomination /cdrom ou /mnt/cdrom.

A présent, descendez dans le répertoire associé à votre lecteur en utilisant la commande cd /mnt/cdrom. Vous observerez que celui-ci n'est plus vide : il contient les fichiers de votre CD-ROM. Vous pouvez naviguer parmi les fichiers comme s'ils avaient été copiés sur votre disque dur. C'est une des caractéristiques qui rendent UNIX si agréable.

Lorsque vous aurez terminé de consulter les fichiers du CD-ROM, passez dans un autre répertoire que /mnt/cdrom et procédez à l'opération de démontage (en anglais *"unmount"*; notez que la commande est umount) :

```
umount /dev/hdb
eject /dev/hdb
```

20.4.2 Montage de disquettes.

Au lieu d'utiliser mtools, vous pouvez aussi monter une disquette de la manière suivante :

```
mkdir /mnt/floppy
mount -t vfat /dev/fd0 /mnt/floppy
```

ou, dans le cas d'anciennes disquettes MS-DOS, employer les commandes suivantes :

```
mkdir /mnt/floppy
mount -t msdos /dev/fd0 /mnt/floppy
```

Avant que vous n'éjectiez la disquette, il est essentiel d'exécuter :

```
umount /dev/fd0
```

de sorte que les données situées dans le *cache* soient écrites sur le disque. Ne pas démonter la disquette avant de l'éjecter corrompra sans aucun doute son système de fichiers.

20.4.3 Montage de partitions Windows et NT.

Le montage d'une partition Windows peut être réalisé avec le système de fichiers `vfat`, et celui de partitions NT (en lecture seule) avec le système de fichiers `ntfs`. VFAT est aussi supporté et autodétecté. Par exemple,

```
mkdir /windows
mount -t vfat /dev/hda1 /windows
mkdir /nt
mount -t ntfs /dev/hda2 nt
```

20.5 Réparation des systèmes de fichiers : `fdisk`.

La commande `fsck` permet la vérification du système de fichiers (*file system check*). `fsck` effectue un balayage du système de fichiers, tout en affichant un rapport et en fixant les erreurs. En principe, celles-ci apparaissent seulement lorsque le noyau a été arrêté avant que le système de fichiers n'ait été démonté. Dans ce cas, il se peut qu'une opération d'écriture soit inachevée : le système de fichiers se retrouve alors dans un état incohérent. Ceci se produit lors d'une panne de courant brutale. Dans la terminologie anglo-saxonne, le système est qualifié d'*unclean*.

`fsck` est utilisé de la manière suivante :

```
fsck [-V] [-a] [-t <système_de_fichiers>] <périphérique>
```

`-V` indique qu'il faut produire une sortie en mode bavard (*verbose*) ; `-a` force la vérification du système de fichiers de manière non-interactive, c'est-à-dire en ne consultant pas l'utilisateur avant d'effectuer les réparations.

La commande que vous utiliserez normalement avec LINUX si vous ne connaissez pas l'ensemble des propriétés du système de fichiers `ext2` est :

```
fsck -a -t ext2 /dev/hda1
```

bien que vous puissez omettre l'option `-t` puisque LINUX autodétecte le système de fichiers. Notez que vous ne devriez pas exécuter `fsck` sur un système de fichiers monté. Dans certains cas exceptionnels, il est possible de lancer la commande `fsck` sur un système de fichiers monté en lecture seule.

En fait, `fsck` ne fait que lancer un programme analysant le système de fichiers. Dans le cas d'`ext2`, la commande `e2fsck` (aussi connue sous le nom de `fsck.ext2`) est exécutée. Voir `e2fsck`(8) pour une description exhaustive.

Pendant une vérification interactive (lorsque l'option `-a` n'est pas utilisée ou lorsque l'option `-r` est activée par défaut), diverses questions sont posées à l'utilisateur. Elles concernent la sauvegarde et les corrections. Il est prudent de sauver les données si vous n'êtes pas sûr de vous. La sauvegarde aura lieu dans le répertoire `lost+found` qui se trouve sur le répertoire racine du périphérique. Ainsi, les répertoires suivants pourraient exister : `/lost+found`, `/home/lost+found`, `/var/lost+found`, `/usr/lost+found`, etc. Après avoir demandé la vérification du répertoire `/dev/hda9`, par exemple, affichez le répertoire `/home/lost+found` à l'écran et éliminez les éléments qui ne sont plus nécessaires. La plupart du temps, il s'agira des fichiers temporaires et des journaux, c'est-à-dire des fichiers qui changent de contenu fréquemment. Il est rare de perdre des fichiers importants lors d'une opération où l'ordinateur est éteint de manière incorrecte (en anglais, *shutdown* désigne l'opération consistant à éteindre un ordinateur proprement).

20.6 Erreurs du système de fichiers au démarrage.

Relisez essentiellement la section 20.5, puis exécutez `fsck` sur le système de fichiers pour lequel une erreur est mentionnée.[2]

20.7 Montage automatique : `fstab`.

Les montages manuels expliqués dans la section 20.4 sont destinés aux nouveaux périphériques ou aux éléments amovibles. Naturellement, il est nécessaire d'automatiser le montage au cours de l'initialisation de la machine (*boot*). Le fichier `/etc/fstab` contient la description des périphériques à monter et leur méthode de montage.

`/etc/fstab` devrait ressembler à ceci, qui correspond au partitionnement décrit à la section 20.2 :

[2] NdT : dans ce cas, il sera certainement utile d'utiliser le mode *rescue* en tirant parti du CD d'installation ou d'une disquette de sauvetage.

```
/dev/hda1      /              ext2        defaults        1 1
/dev/hda6      /tmp           ext2        defaults        1 2
/dev/hda7      /var           ext2        defaults        1 2
/dev/hda8      /usr           ext2        defaults        1 2
/dev/hda9      /home          ext2        defaults        1 2
/dev/hda5      swap           swap        defaults        0 0
/dev/fd0       /mnt/floppy    auto        noauto,user     0 0
/dev/cdrom     /mnt/cdrom     iso9660     noauto,ro,user  0 0
none           /proc          proc        defaults        0 0
none           /dev/pts       devpts      mode=0622       0 0
```

Intéressons-nous pour l'instant aux six premières lignes. Les trois premiers champs (ou colonnes) indiquent respectivement les partitions, les répertoires où elles doivent être montées et les systèmes de fichiers qui sont associés. Le quatrième champ donne les options, c'est-à-dire les arguments passés à l'option `-o` de la commande `mount`.

Le cinquième champ précise si le système de fichiers contient des fichiers réels. Ce champ, qui communément n'est pas utilisé, est exploité par la commande `dump` pour décider si une sauvegarde est possible.

Le dernier champ nous indique l'ordre dans lequel un `fdisk` devrait être réalisé sur les partitions. La partition `/` devrait être la première concernée (avec un `1`) suivie de toutes les autres partitions (avec un `2`). En plaçant un `2` partout ailleurs, on s'assure que les partitions placées sur différents disques seront vérifiées en parallèle, ce qui améliore la vitesse à laquelle la phase d'initialisation (*boot*) procède.

Les entrées `floppy` et `cdrom` vous permettront d'utiliser une forme contractée pour la commande `mount`. Cette commande ne considérera que le répertoire correspondant et le système de fichiers dans `/etc/fstab`. Essayez :

```
mount /dev/cdrom
```

Ces entrées possèdent aussi l'option `user`, qui permet aux utilisateurs ordinaires de monter ces périphériques. L'option `-ro` nous apprend que le CD-ROM est monté en lecture seule (*r ead-o nly*) et la commande `noauto` indique à `mount` de ne pas monter les systèmes de fichiers lors de la phase d'initialisation de la machine (des explications plus précises se trouvent plus bas dans le texte).

`proc` est une base de données d'information du noyau, qui ressemble à un système de fichiers. Par exemple, `/proc/cpuinfo` ne constitue pas un type de fichier réellement présent sur votre disque. Essayez `cat /proc/cpuinfo`.

Beaucoup de programmes utilisent `/proc` pour obtenir des informations dynamiques sur l'état et la configuration de votre machine. Davantage à ce propos est décrit dans la section 43.4.

Le système de fichiers `/dev/pts` est un autre pseudo-système de fichiers qui produit des paires de terminaux maîtres/esclaves pour les programmes. Ceci s'adresse plus particulièrement aux développeurs.

20.8 Montage manuel de /proc.

Vous pouvez monter le système de fichiers /proc avec la commande :

```
mount -t proc /proc /proc
```

Cette méthode constitue une exception à l'usage normal de mount. Notez que toutes les installations communes de LINUX requiert que /proc soit monté lors de l'initialisation de votre système. Les seules occasions où cette commande est nécessaire sont le démarrage manuel ou l'usage de chroot (voir la section 21.7.1 à la page 220).

20.9 RAM et périphérique loopback.

Un *périphérique RAM* est un périphérique bloc qui peut être employé comme un disque, mais en pointant réellement vers une zone de la RAM.

Un périphérique *loopback* est aussi un périphérique bloc mais qui, en fait, pointe vers un fichier ordinaire.

Pour illustrer ces propos, vous pourriez créer une disquette avec son sytème de fichiers et des fichiers *(le tout sans avoir vraiment de disquette)* ; ensuite, vous pourriez transférer ce que vous venez de créer vers le lecteur de disquette grâce à dd (voir la section 20.9.1). Vous avez aussi la possibilité d'avoir un autre système LINUX complet dans un fichier de 500 Mo sur une partition Windows *et* de pouvoir bouter dessus ; c'est-à-dire en évitant d'avoir à faire un partitionnement sur une machine Windows pour utiliser LINUX. Tout ceci est faisable avec les périphériques *RAM* et *loopback*.

20.9.1 Formater une disquette à l'intérieur d'un fichier.

Les opérations sont simples. Pour créer une disquette ext2 dans un *fichier* de 1440 ko, exécutez :

```
dd if=/dev/zero of=~/ficher-disquette count=1440 bs=1024
losetup /dev/loop0 ~/fichier-disquette
mke2fs /dev/loop0
mkdir ~/mnt
mount /dev/loop0 ~/mnt
ls -al ~/mnt
```

Lorsque vous aurez terminé la copie des fichiers que vous souhaitez avoir dans ~/mnt, exécutez simplement :

```
umount ~/mnt
losetup -d /dev/loop0
```

Pour transférer vers la disquette, faites :

```
dd if=~/fichier-disquette of=/dev/fd0 count=1440 bs=1024
```

Une méthode similaire pour les périphériques RAM consiste à lancer les commandes suivantes :

```
dd if=/dev/zero of=/dev/ram0 count=1440 bs=1024
mke2fs /dev/ram0
mkdir /dev/ram0 ~/mnt
ls -al ~/mnt
```

Quand vous en avez terminé avec la copie des fichiers que vous voulez avoir dans `~/mnt`, exécutez simplement :

```
umount ~/mnt
```

Pour transférer le système de fichiers vers une disquette (ou un fichier), exécutez respectivement :

```
dd if=/dev/ram0 of=/dev/fd0 count=1440 bs=1024
dd if=/dev/ram0 of=~/fichier-disquette count=1440 bs=1024
```

20.9.2 Fichiers CD-ROM.

Une autre astuce consiste à déplacer votre lecteur CD-ROM vers un fichier pour un accès à haute-vitesse. Ici, nous utilisons un raccourci au lieu de la commande `losetup` :

```
dd if=/dev/cdrom of=un_nom.iso
mount -t iso9660 -o ro,loop=/dev/loop0 un_nom.iso /cdrom
```

20.10 Remontage du mode “lecture-seule” au mode “lecture-écriture”.

Un système de fichiers qui est déjà monté en lecture seule (*read-only*) peut être remonté en lecture-écriture (*read-write*) de la manière suivante :

```
mount -o rw,remount /dev/hda1 /
```

Cette commande est très utile quand vous vous connectez en mode *monoutilisateur* (*single-user*) sans accès en écriture à votre partition racine.

20.11 `sync` sur un disque.

Les fichiers stock ou *cache* du noyau écrivent leurs opérations en mémoire pour améliorer les performances. Des purges (*flush*) sont réalisées assez souvent par transfert sur un support magnétique. Parfois, vous souhaiterez forcer la purge, d’où l’usage de la commande :

```
sync
```

Chapitre 21

Scripts de shell avancés.

Ce chapitre complète notre discussion à propos des scripts de shell `sh`, commencée au chapitre 8 et poursuivie au chapitre 10. Ces trois chapitres représentent pratiquement tout ce que vous pouvez faire avec le shell `bash`.

21.1 Liste de commandes.

Les opérateurs spéciaux `&&` et `| |` permettent d'exécuter des fonctions en série, comme par exemple :

```
grep '^harry :' /etc/passwd | | useradd harry
```

Le symbole `| |` signifie qu'il ne faut exécuter le seconde commande que si la première retourne une erreur. Dans la commande ci-dessus, `grep` émet un code de sortie (`1`) si `harry` ne se trouve pas parmi la liste des utilisateurs reprise dans le fichier `/etc/passwd`. Ceci provoque alors l'exécution de la commande `useradd`.

Une représentation alternative consiste en :

```
grep -v '^harry :' /etc/passwd &&  useradd harry
```

où l'option `-v` inverse la logique de recherche des correspondances de `grep` (ici, si `grep` ne trouve pas le nom `harry`, il y a réussite et la commande `useradd` doit être exécutée). Le symbole `&&` a donc la signification contraire de `| |` (c'est-à-dire l'exécution de la seconde commande si la première réussit).

Les adeptes des scripts enchaînent souvent plusieurs commandes pour créer une opération sous une forme succinte :

```
grep -v '^harry :' /etc/passwd && useradd harry | | \
    echo "`date` : useradd failed" >> /var/log/mon_journal_special
```

21.2 Paramètres spéciaux : $?, $*, ...

Le contenu d'une variable ordinaire (appelons-la VARNAME) peut être obtenu en faisant précéder le nom de la variable d'un signe $, comme dans $VARNAME. Les variables communément utilisées comme PATH et les variables spéciales PWD et RANDOM sont discutées au chapitre 10. D'autres cas sont documentés dans le paragraphe qui suit (il s'agit d'extraits de la page de man de bash).[1]

Paramètres spéciaux.
Le shell traite plusieurs variables (ou paramètres) de manière spéciale. Ces paramètres peuvent seulement être lus ; il n'est pas possible de les modifier.

$* s'applique aux paramètres de position en commençant la numérotation à partir d'1 (c'est-à-dire, aux arguments en ligne de commande passés au script de shell, avec **$1** comme premier argument, **$2** pour le second, etc.). Quand l'extension (**$***) apparaît dans des guillemets doubles, elle se comporte comme un mot unique bien que chaque paramètre soit séparé par le premier caractère de la variable spéciale **IFS** (**I***nternal* **F***ield* **S***eparator*).[2] Cela signifie que **"$*"** est équivalent à **"$1***c***$2***c***..."**, où *c* est le premier caractère de la valeur de la variable **IFS**. Si la variable **IFS** n'est pas définie, les paramètres sont séparés par des espaces. Si **IFS** est nulle, les paramètres sont joints sans espacements.

$@** s'applique aux paramètres de position, en commençant la numérotation à partir de 1. Lorsque le développement se produit entre des guillemets doubles, chaque paramètre se comporte comme un mot séparé. C'est-à-dire que **"$@" est équivalent à **"$1" "$2"** ... Lorsqu'il n'y a pas de paramètres de position, **"$@"** et **$@** ne s'applique à rien (donc, ils sont éliminés). [Remarque : ceci est utile pour écrire des scripts de shell d'interfaces qui ne font qu'ajouter un argument].

$# se développe pour fournir le nombre de paramètres de position en mode décimal .

$? renvoie le code de retour du tube[3] le plus récemment exécuté en avant-plan [c'est-à-dire au code de retour de la dernière commande].

$- se développe en une liste des options courantes du shell telles que spécifiées par la commande interne **set** ou attribuées par le shell lui-même (comme dans l'option **-i**).

$$ se développe en le PID du shell courant. Dans un sous-shell [où les commandes sont placées entre ()],[4] cette variable s'applique au PID du shell courant et non du sous-shell.

$! se développe en le PID de la commande en arrière-plan (ou commande asynchrone) la plus récemment exécutée. [C'est-à-dire qu'après avoir exécuté une commande en arrière-plan avec *commande* **&**, la variable **$!** donnera l'ID du processus].

$0 se développe pour donner le nom du shell ou du script de shell. Ce paramètre est fixé lors de l'initialisation du shell. Si **bash** est invoqué avec un fichier de commande, **$0** contiendra le nom de ce fichier. Si **bash** est démarré avec l'option **-c**, **$0** contiendra le premier argument (s'il y en a un) après la chaîne à exécuter. Autrement, ce paramètre contiendra le chemin utilisé pour invoquer **bash,** tel que donné par l'argument zéro. [Notez que **basename $0** est un moyen élégant d'obtenir le nom de la commande courante, isolément].

$_ lors de démarrage du shell, ce paramètre contient le chemin absolu du shell ou du script de shell en cours d'exécution, tel que passé sur la ligne de commande. Subséquemment, la variable est développée en le dernier argument de la commande précédente. Elle

[1] Brian Fox et Chet Ramey doivent être remerciés pour leur contribution.
[2] NdT : la variable IFS est souvent un espace.
[3] NdT : il s'agit d'un tube associé à un enchaînement de tubes, appelé en anglais *pipeline*.
[4] NdT : ceci est une note pour donner un exemple de *sous-shell* en relation avec l'usage de la commande tar pour copier un répertoire intitulé origine (en l'occurrence) d'un disque vers un autre. Il est entendu que le chemin chemin_vers_repertoire_origine est le chemin absolu donnant accès à ce répertoire :

```
cd /repertoire_destination
(cd /chemin_vers_repertoire_origine ; tar cf - *) | tar xvf -
```

correspond aussi au chemin d'accès complet de chaque commande exécutée et, elle est placée dans l'environnement exporté vers cette commande. Lors de la vérification de l'arrivée de courriel, ce paramètre contient le nom du fichier de courriel en cours de vérification.

21.3 Développements.

Le terme développement (ou en anglais : *expansion*) se rapporte à la manière dont `bash` modifie la ligne de commande avant de l'exécuter. `bash` réalise plusieurs modifications de texte de la ligne de commande, en procédant dans l'ordre que voici :

Développement d'accolades (*brace expansion*) : vous pouvez utiliser, par exemple, le raccourci `touch fichier{1,2,3}.txt` pour créer plusieurs fichiers intitulés `fichier1.txt`, `fichier2.txt` et `fichier3.txt`. Cette méthode est connue sous le nom de développement d'accolades (*brace expansion*). Elle se produit avant tout autre type de modification de la ligne de commande.

Développement de tilde (*tilde expansion*) : le caractère spécial ~ (tilde) est remplacé par le chemin complet contenu dans la variable d'environnement `HOME`, ou par le répertoire home propre à chaque utilisateur (si `$HOME` est vide). ~+ est remplacé par le répertoire de travail courant et ~- est substitué par le répertoire de travail, précédant qui est le plus récent. Ces deux derniers développements sont peu utilisés.

Développement de paramètre (*parameter expansion*) : ce terme se rapporte au développement de tout terme commençant par un signe `$`. Notez que `$`*VAR* et `${`*VAR*`}` opèrent exactement la même manière, à ceci près que, dans le second cas, la variable peut contenir des caractères ne formant pas de mots entiers (*non-"whole word"*) qui normalement pourraient amener une confusion dans l'interprétation que doit en faire `bash`.
Il y a plusieurs astuces pour le développement de paramètres dont vous pouvez tirer parti afin de manipuler des chaînes. La plupart des programmeurs du shell ne les exploitent pas, probablement parce que les autres UNIX les tolèrent mal.

${VAR :–default} A moins que le paramètre `$`*VAR* ne soit vide ou inexistant (dans ce cas, le message d'erreur *"message"* sera émis), cette expression équivaut à `$`*VAR*.

${VAR :=default} Comme précédemment, mais *default* est attribué à `VAR` au cas où cette variable est vide.

${VAR :-default} Ceci produit une chaîne vide dans $*VAR* sauf si *VAR* est inexistant ou vide. Le comportement est inverse de celui de `${`*VAR*`:-default}`

${VAR :?message} Ceci produira `$`*VAR* sauf si *VAR* est vide ou inexistant. Dans ce cas, le message d'erreur "*message*" sera affiché.

${VAR :offset} ou ${VAR :n :l} Ceci renvoie le $\mathbf{n}^{\text{ième}}$ caractère de `$`*VAR* et les *l* caractères suivants. Si *l* n'est pas stipulé, tous les caractères à la droite du $n^{\text{ième}}$ caractère seront affichés. Ceci est utile lorsqu'il faut diviser une chaîne. Essayez :

```
TEXT=scripting_for_phun
echo ${TEXT :10 :3}
echo ${TEXT :10}
```

${#VAR} affiche le nombre de caractères composant $*VAR*.

${ !PRE*} affiche la liste de toutes les variables dont les noms commencent pas *PRE*.

${VAR#pattern} le contenu de $*VAR* est affiché alors que l'expression globale *pattern* est supprimée dans la partie de tête de la chaîne. Par exemple, lorsque `${TEXT#scr}` est appliqué à l'exemple précédant, la chaîne suivante est affichée `ipting_for_phun`.

${VAR##pattern} effectue le même travail que dans le cas qui précède mais si *pattern* contient des caractères de remplacement (*jokers)*, l'expression tentera de rechercher une correspondance sur toute la chaîne des caractères.

${VAR%pattern} Idem que dans le cas de ${*VAR#pattern*}, sauf que les caractères sont éliminés de la partie résiduelle de la chaîne.

${VAR%%pattern} Idem que dans ${*VAR##pattern*}, sauf que les caractères sont éliminés de la partie résiduelle de la chaîne.

${VAR/search/replace} $*VAR* est retournée avec la première occurrence de la chaîne *search* remplacée par *replace.*

${VAR/#search/replace} Idem que dans le cas de ${*VAR/search/replace*} si ce n'est que la correspondance est testée sur la partie de tête de $*VAR*.[5]

${VAR/%search/replace} Idem que dans le cas de ${*VAR/search/replace*} sauf que la correspondance est entreprise sur la partie finale de$*VAR*.

${VAR//search/replace} Idem que dans le cas de ${*VAR/search/replace*} sauf que toutes les occurrences de *search* sont remplacées.

Développement de guillemets inverses : (**backquote expansion**) Nous avons déjà rencontré un développement de ce type dans le paragraphe 8.12. Observez que la notation avec le signe additionnel $(*commande*) équivaut à *'commande'* excepté que le caractère d'échappement (\) n'est pas nécessaire pour les caractères spéciaux.

Développement arithmétique : (ou **arithmetic expansion**) Nous avons déjà réalisé un développement arithmétique à la page 92. La notation en $((*expression*)) est équivalente à $[*expression*] vue à la page 92.

Finalement, la dernière modification relative à la ligne de commande est la division de la ligne de commande en mots selon les espaces qui les séparent. La variable d'environnement `IFS` (*Internal Field Separator*) détermine quel caractère sépare les mots dans une commande (usuellement, il s'agit d'un espace). Dans le cas de commandes constituées de mots, les

[5] NdT : dans la continuité de l'exemple précédent avec TEXT, `echo ${TEXT/#scr/scc}` renverra `sccipting_for_phun`. Alors qu'`echo ${TEXT/ipt/atch}` renverra `scratching_for_phun`.

chemins sont développés selon les caractères de remplacement globaux. Consultez `bash`(1) pour une description des motifs correspondant aux options que beaucoup d'utilisateurs méconnaissent.

21.4 Commandes internes.

Beaucoup de commandes exécutent des fonctionnalités internes au `bash` ou sont interprétées de manière particulière. Elles n'invoquent pas d'exécutables en dehors du système de fichiers. Certaines d'entre elles sont décrites au chapitre 8 et quelques unes sont discutées ici, en complément. Pour une description exhaustive, consultez `bash`(8) :

: un double-point en soi ne signifie rien. Il représente une ligne sans opération, comme dans :

```
if <commande> ; then
   :
else
   echo "la <commande> a échoué"
fi
```

. filename args un simple point agit comme la commande `source`. Voyez ci-dessous.

alias ***commande*=*valeur*** crée un synonyme (ou *alias*) associé à une commande. Essayez :

```
alias necho="echo -n"
necho "hello"
```

Certaines distributions présentent des `alias` pour les commandes `mv`, `cp` et `rm` de manière à inclure l'option interactive `-i`. Ceci empêche la suppression de fichiers sans avertissement (pour `rm` du moins), mais cela peut gêner le travail de l'administrateur. Voyez votre fichier `~/.bashrc` pour ces commandes. Essayez aussi la commande `unalias`.

unalias ***commande*** détruit le synonyme créé avec la commande `alias`.

alias -p affiche la liste des synonymes.

eval ***argument...*** exécute l'argument (ou les arguments) comme une ligne de script shell.

exec ***commande argument...*** commence l'exécution de *commande* sous le même PID que le script courant. Ceci est souvent utilisé dans les scripts de shell qui constitutent des interfaces pour certains programmes. Le script d'interface installe des variables d'environnement et ensuite, il exécute le programme binaire en dernière ligne. `exec` ne devrait jamais rien retourner.

local ***variable*=*valeur*** attribue une valeur à une variable. La variable résultante est seulement visible pour la fonction en cours.

pushd ***repertoire*** **(et popd)** ces deux commandes sont utiles pour passer d'un répertoire à l'autre. `pushd` peut être utilisée au lieu de `cd`, mais

contrairement à `cd`, le répertoire est sauvé dans une liste de répertoires. A tout moment, le fait d'entrer `popd` vous renvoie au répertoire précédent. Cela est très pratique pour naviguer étant donné que ces commandes conservent un historique des répertoires visités.

printf ***format arguments...*** Cette fonction ressemble à la fonction `printf` en **C**. Son résultat est similaire à celui d'`echo` mais cette commande interne est utile pour des formatages complexes. Consultez `printf`(3) pour plus de détails et essayez à titre d'exemple : `printf "%10.3e\n" 12` .

pwd affiche le répertoire courant.

set affiche la valeur de toutes les variables d'environnement. Voyez la section 21.6 consacrée à la commande `set`.

source ***nom_de_fichier arguments...*** lit *nom_de_fichier* dans l'environnement du shell courant. Ceci est utile pour exécuter un script de shell quand les variables d'environnement fixées par ce script doivent être préservées.

times affiche les durées d'utilisation "système" et "utilisateurs" pour le shell et ce, pour les processus exécutés dans le shell.

type ***commande*** indique si *commande* est un synomyne, une commande interne ou un exécutable du système.

ulimit affiche et fixe les différentes limites de ressources des utilisateurs telles que les limites d'utilisation de la mémoire et de la CPU/ Voir `bash` (1) pour de plus amples détails.[6]

umask consultez la section 15.2.

unset ***VAR*** efface une variable ou une variable d'environnement.

unset -f ***fonction*** supprime une fonction.

wait effectue une pause jusqu'à ce que tous les tâches en arrière-plan soient terminés.

wait ***PID*** effectue une pause jusqu'à ce que le processus en arrière-plan (dont le ProcessIDentifier est *PID*) soit terminé puis, retourne le code du processus en arrière-plan.

wait ***%job*** idem mais en fonction d'une tâche particulière.

21.5 Capture de signaux – la commande `trap`.

Souvent, vous voudrez que votre script réagisse en réponse à un signal. Une liste des signaux se trouve à la page 115. Pour capter un signal, il faut créer une fonction et, ensuite, utiliser la commande `trap` afin d'associer la fonction au signal.

```
#!/bin/sh

function on_hangup ()
{
    echo 'Hangup (SIGHUP) signal received'
}
```

[6] NdT : `man ulimit` indique que cette commande est obsolète. Consultez `man limits`.

```
trap on_hangup SIGHUP

while true ; do
    sleep 1
done

exit 0
```

Lancez le script et ensuite, envoyez à son PID le signal `-HUP`, à titre de test (rapportez-vous aussi à la section 10.5).[7]

Une autre fonction importante pour un programme consiste à effectuer un nettoyage après la terminaison. Le signal spécial `EXIT` (qui n'est pas *stricto sensu* un signal) exécute le code suivant à la terminaison d'un script :

```
#!/bin/sh

function on_exit ()
{
    echo 'Je vais effacer les fichierstemporairesàprésent'
}

trap on_exit EXIT

while true ; do
    sleep 1
done

exit 0
```

Arrêter le programme précédent amène ce dernier à éditer sa propre épitaphe.[8]

Si, au lieu du nom d'une fonction, le symbole `-` est passé, le signal devient "unbounded" (c'est-à-dire qu'on lui attribue sa valeur par défaut).

21.6 Réglages internes ; la commande `set`.

La commande `set` peut modifier certains réglages dans le comportement du shell. Les options courantes sont affichées à l'aide de la commande `echo $-`. Les diverses commmandes `set` (c'est-à-dire set avec diverses options) sont usuellement entrées au début d'un script ou données comme option en ligne de commande à `bash`. Le fait d'utiliser `set +option` au lieu de `set -option` désactive l'option. Voici quelques exemples :

set -e provoque une sortie immédiate si toute commande simple produit une erreur.

[7] NdT : dans ce cas précis, vous pouvez exécuter le script et ensuite, en supposant que vous ayez appelé ce script `hangup.sh`, exécutez la commande : `killall -HUP hangup.sh`. La répétition de la commande provoque l'affichage à l'écran du commentaire "`Hangup (SIGHUP) signal received`".

[8] NdT : en supposant que vous ayez nommé ce programme `exit.sh`, vous pouvez l'arrêter en exécutant : `killall EXIT exit.sh`.

set -h enregistre l'emplacement des commandes dans votre `PATH`. Le shell sera affecté si des binaires sont insérés dans les répertoires de votre `PATH`, ce qui provoquera peut-être une erreur du type `No such file or directory`. Dans ce cas, désactivez cette option ou redémarrez votre shell. Cette option est activée par défaut.

set -n lit les commandes sans les exécuter. Cette commande est utile pour la vérification de la syntaxe.

set -o posix se conforme exactement à la norme POSIX 1003.2

set -u rapporte une erreur lorsqu'on essaye de référencer une variable qui est vide ou inexistante. Usuellement, `bash` remplit une chaîne vide.

set -v affiche chaque ligne d'un script au fur et à mesure de son exécution.

set -x affiche chaque développement de commande au fur et à mesure de l'exécution.

set -C n'écrase pas le contenu de fichiers existants lors de l'usage de `>`. Vous pouvez utiliser `> |` pour forcer l'écrasement.

21.7 Scripts utiles et commandes.

Dans cette section, vous trouverez une collection de scripts très utiles que les utilisateurs demandent très souvent sur les listes de diffusion. Consultez la section 45.3 à la page 570 pour des scripts de vérification relatifs à la sécurité.

21.7.1 `chroot`.

La commande `chroot` fait croire à un processus que son système de fichiers n'est pas `/`. Par exemple, vous pouvez avoir une installation Debian complète résidant sous le répertoire `/mnt/debian`. Vous pouvez faire usage de la commande :

```
chroot /mnt/debian bash -i
```

pour lancer un shell `bash` interactif, sous le système de fichiers racine `/mnt/debian`.[9] Cette commande exécute donc `/mnt/debian/bin/bash -i`. Toute commande ultérieure exécutée sous ce shell n'aura pas connaissance du vrai répertoire `/` de votre système, de sorte que vous pouvez utiliser votre Debian sans avoir à réamorcer. Toutes les commandes se comporteront comme si elles étaient lancées depuis une machine UNIX indépendante. Un avertissement toutefois : vous pourriez devoir remonter votre système de fichier `/proc` à l'intérieur de votre système de fichier `chroot` (voir la section 20.8).

Ceci est utile pour améliorer la sécurité. Les services non-sécurisés du réseau peuvent changer d'un répertoire racine à l'autre. Toute corruption affectera le système entier.

La plupart des disques de secours possèdent une commande `chroot`. Après l'amorçage, vous pouvez monter manuellement les systèmes de fichiers sur votre propre disque, et ensuite utiliser un `chroot` pour commencer à exploiter une machine de manière normale. Notez que la commande `chroot <nouvelle_racine>` sans arguments invoque un shell par défaut.

[9] NdT : tout se passe donc comme si `/mnt/debian` était devenu une nouvelle `/` (indépendante de la racine initiale qui reste la seule "vraie" dans l'absolu).

21.7.2 `if` conditionnels.

Au chapitre 8, `if test ...` a été utilisé pour contrôler le flux de programmation. Cependant, `bash` possède un alias interne pour la fonction `test`, le crochet gauche, `[`.

L'usage de `[` au lieu de `test` apporte de l'élégance aux scripts :

```
if [ 5 -le 3 ] ; then
    echo '5 < 3'
fi
```

A ce stade, il est important de réaliser que la commande `if` ne comprend rien à l'arithmétique. Elle exécute seulement une commande `test` (ou ici `[`) et analyse le code qui résulte. Si ce dernier est le nombre zéro, la commande est supposé avoir réussi et `if` procède avec le corps contenu dans le bloc qui suit `if`. Il incombe à la commande `test` d'évaluer correctement l'expression arithmétique qui lui est passée.

`if` peut également être utilisée sans commande :

```
if echo "$PATH" | grep -qwv /usr/local/bin ; then
    export PATH= "PATH :/usr/local/bin"
fi
```

Ces lignes permettent d'ajouter `/usr/local/bin` dans votre `PATH` si `grep` n'y trouve pas ce répertoire.

21.7.3 `patch` et `diff`.

Vous désirerez peut-être trouver les différences éventuelles entre deux fichiers, par exemple pour estimer les changements qui ont été introduits dans un fichier appartenant à deux versions. Il y a aussi d'autres cas : lorsqu'une grande quantité de code source a été mise à jour, il est absurde de charger l'arborescence entière des répertoires si seulement quelques changements ont eu lieu. Aussi, ne souhaiterez-vous obtenir que la liste des modifications.
L'utilitaire `diff` extrait et affiche les lignes qui diffèrent dans deux fichiers. Cette commande peut être utilisée de la manière suivante :

```
diff -u <ancien_fichier> <nouveau_fichier>
```

`diff` peut également être employée pour extraire les différences entre deux arborescences de répertoires. `diff` compare récursivement tous les fichiers correspondant :

```
diff -u --recursive --nouveau_fichier <ancien_repertoire> \
<nouveau_repertoire> > <correctif>.diff
```

Le résultat de cette commande constitue en fait un *correctif* (*patch file*) associé à l'arborescence. Ce dernier vous permet de mettre à jour `<ancien_repertoire>` pour obtenir `<nouveau_repertoire>`.

Les fichiers correctifs se termineront peut-être par `.patch` ; ils sont très sou-

vent gzipés. Le fichier correctif ainsi extrait peut être appliqué à <ancien_repertoire> de cette manière :

```
cd <ancien_repertoire>
patch -p1 -s < <correctif>.diff
```

Ceci rendra <ancien_repertoire> identique à <nouveau_repertoire>. L'option -p1 extrait le nom du répertoire principal dans le fichier correctif. La présence d'un nom de répertoire principal dans le correctif perturbe souvent la commande patch.

21.7.4 Test de connectivité internet.

Il se peut que vous souhaitiez passer ce paragraphe pour y revenir après la lecture de la théorie relative au réseau.

Le véritable test concernant une connexion internet consiste à réussir une requête DNS.[10] Naturellement vous pouvez employer la commande ping pour déterminer si un serveur fonctionne. Cependant, certains réseaux filtrent les messages ICMP[11] et de ce fait, la commande ping ne vérifie pas le fonctionnement de votre DNS. dig envoie en paquet UDP[12] similaire à celui émis par ping. Malheureusement, la méthode impliquant dig prend beaucoup de temps et on a souvent envie de l'arrêter au bout de quelques secondes.

Le script qui suit permet d'interroger un serveur de noms distant. Typiquement, ce script pourrait lancer fetchmail et peut-être uucico. Fixez l'adresse IP du serveur de noms à une valeur adéquate telle que celle de votre fournisseur d'accès internet (FAI) ;[13] ensuite, augmentez la durée à 2 secondes ou plus si votre serveur de noms prend plus de temps à vous répondre.

```
MY_DNS_SERVER=197.22.201.154

while true ; do
   (
     dig @$MY_DNS_SERVER netscape.com IN A &
     DIG_PID=$!
     { sleep 2 ; kill $DIG_PID ; } &
     sleep 1
     wait $DIG_PID
   ) 2 > /dev/null | grep -q '^[^;]*netscape.com' && break
done
```

[10] NdT : DNS est l'abréviation de *Domain Name Server*. Quand un utilisateur envoie un courriel ou pointe son navigateur sur un domaine internet (comme www.unixtech.be), le DNS traduit cette adresse aisée à mémoriser en une adresse internet (IP). Ce terme recouvre deux définitions : (i) celle que nous venons de voir (la conversion de nom d'adresses) mais aussi (ii) l'attribution de noms d'hôtes.

[11] NdT : ICMP est l'abréviation d'*Internet Control Message Protocol*.

[12] NdT : UDP est l'acronyme d'*User Data Protocol*.

[13] NdT : le terme anglais est ISP (acronyme d'*Internet Service Provider*).

21.7.5 grep récursif (recherche).

La recherche récursive dans une arborescence est très aisée grâce aux commande `find` et `xargs`. N'hésitez pas à consulter leur page de man. Le tube suivant recherche dans les sources du noyau une carte Ethernet "pcnet" et il affiche le numéro de la ligne :

```
find /usr/src/linux -follow -type f | xargs grep -iHn pcnet
```

Il ne vous échappera pas que cette commande retourne un grand nombre de données. Parcourez-les, cela sera instructif.

Il est également possible de limiter la recherche à une extension de fichier donnée, ce qui constitue un exemple typique du tube décrit ci-dessus :

```
find /usr/src/linux -follow -type f -name '*.[ch]' | xargs grep - iHn
pcnet
```

Notez que les nouvelles versions de `grep` possèdent l'option `-r` pour effectuer une recherche récursive parmi les répertoires.

21.7.6 Recherche et remplacement récursifs.

Vous devrez parfois réaliser une opération de recherche et remplacement parmi tous les fichiers sources d'une arborescence complète. Un cas typique consiste à changer le nom d'appel d'une fonction dans de nombreux fichiers sources en **C**. Le script qui suit est indispensable pour tout répertoire `/usr/local/bin`. Observez bien la manière avec laquelle ce script fait appel à lui-même et ce, de manière récursive.

```
#!/bin/sh

N=`basename $0`

if [ "$1" = "-v" ] ; then
    VERBOSE="-v"
    shift
fi

if [ "$3" = "" -o "$1" = "-h" -o "$1" = "--help" ] ; then
    echo "$N : Usage"
    echo "        $N [-h|--help] [-v] <regexp-search> \
<regexpr-replace> <glob-file>"
    echo
    exit 0
fi

S="$1"; shift; R="$1"; shift
T=$$replc

if echo "$1" | grep -q /; then
    for i in "$@"; do
```

```
      SEARCH=‘echo “$S” | sed ’s,/,\\\\/,g’‘
      REPLACE=‘echo “$R” | sed ’s,/,\\\\/,g’‘
      cat $i | sed “s/$SEARCH/$REPLACE/g” > $T
      D=”$?”
      if [ ”$D” = ”0” ] ; then
          if diff -q $T $i >/dev/null ; then
             :
          else
             if [ ”$VERBOSE” = ”-v” ] ; then
               echo $i
             fi
             cat $T > $i
          fi
          rm -f $T
      fi
    done
else
    find . -type f -name “$1” | xargs $0 $VERBOSE “$S” “$R”
fi
```

21.7.7 `cut` et `awk` – manipulation des champs de fichiers-textes.

La commande `cut` est utilisée pour isoler un des champs présents dans certains fichiers ; les deux commandes qui suivent effectuent le même travail sur le fichier `/etc/passwd` :

```
cut -d : -f1 /etc/passwd
cat /etc/passwd | cut -d : f1
```

Cependant, la deuxième forme utilise un processus (`cat`), ce qui charge inutilement la machine, une pratique découragée en programmation.

Le programme `awk` interprète le langage de programmation complet appelé AWK. Un des usages d’`awk` consiste à extraire un champ dans un fichier. Cette commande est plus flexible que `cut` (la commande qui suit a le même effet que les deux qui précèdent) :

```
cat /etc/passwd | awk -F : ’{print $1}’
```

Pour la même raison que dans l’exemple précédant, cette expression devrait d’être remplacée par :

```
awk -F : ’{print $1}’ /etc/passwd
```

Cette flexibilité se manifeste lorsque des espaces séparent les champs comme dans le premier exemple qui suit et qui permet d’isoler les 6, 7 et 8$^{\text{èmes}}$ champs de ce que retourne la commande `ls -al` (on obtient la date et l’heure) :

```
ls -al | awk ’{print $6 “” $7 “” $8}’
ls -al | awk ’{print $5 “ octets”}’
```

La dernière commande isole la taille des fichiers et répertoires (suivie du terme “octets”).

Vous pouvez obtenir les adresses IP non-locales à l'aide de la commande :

```
ifconfig | grep 'inet addr :' | fgrep -v '127.0.0.'| cut -d : -f2 \
                                    | cut -d' 'f1
```

Inversez l'ordre d'une adresse IP avec :

```
echo 192.168.2.3 | awk -F . '{print $4 "." $3 "." $2 "." $1}'
```

Affichez tous les noms d'utilisateurs communs (c'est-à-dire le nom des utilisateurs ayant une valeur d'UserID supérieure à 499 sur RedHat et 999 sur Debian) :

```
awk -F : '$3 >= 500 {print $1}' /etc/passwd
( awk -F : '$3 >= 1000 {print $1}' /etc/passwd )
```

21.7.8 Calculs avec bc.

Les scripts font usage de `bc` pour effectuer des calculs qu'`expr` ne peut manipuler. Par exemple, la conversion en format décimal se fait ainsi :

```
echo -e 'ibase=16;FFFF' | bc
```

La conversion en mode binaire est réalisée à l'aide de la commande :

```
echo -e 'obase=2;12345' | bc
```

Par ailleurs, traiter le sinus de 45° revient à utiliser la commande suivante :

```
pi=`echo "scale=10; 4*a(1)" | bc -l`
echo "scale=10; s(45*$pi/180)" | bc -l
```

21.7.9 Conversion des formats de graphiques sur de nombreux fichiers.

Le programme `convert` du paquet logiciel *ImageMagik* est une commande dont de nombreux utilisateurs de Windows aimeraient disposer. Il permet de convertir de nombreux fichiers d'un format donné vers un autre format. Le changement de l'extension de fichier sera réalisée avec `echo` *nom_de_fichier* `| sed -e 's/\.old$/.new/'`. La commande `convert` fait le reste :

```
for i in *.pcx ; do
    CMD="convert -quality 625 $i `echo $i | sed -e 's/\.pcx*/.png/'`"
# afficher la ligne de commande à l'utilisateur :
    ECHO $CMD
# exécute la ligne de commande :
    eval $CMD
done
```

Notez que le mécanisme de développement "recherche et remplacement" pourrait convenir pour remplacer les extensions de fichiers. L'expression `${i/%.pcx/.png}` produit le résultat désiré.

Incidemment, le code précédent compresse très bien les fichiers de haute résolution `pcx` –éventuellement, le résultat d'une opération de numérisation ou la compilation LaTeX en PostScript réalisée avec Ghostscript (c'est-à-dire : `gs -sDEVICE=pcx256 -sOutpoutFile='page%d.pcx' fichier.ps`).

21.7.10 Ecrasement en toute sécurité de fichiers.

La suppression d'un fichier avec la commande `rm` consiste à enlever le lien entre le nom d'un fichier et les données qu'il contient. Les blocs de fichiers restent sur le disque et ne sont "réclamés" qu'au moment où le système tente de réutiliser les données. Pour supprimer un fichier de manière correcte, il est nécessaire d'écrire des octets aléatoires sur les blocs du disque occupés par le fichier. La manipulation suivante permet d'écraser les fichiers du répertoire courant :

```
for i in * ; do
    dd if=/dev/urandom \
       of="$i"         \
       bs=1024         \
       count=`expr 1+  \
         \`stat "$i" | grep 'Size :' | awk '{print $2}'\` \
            / 1024
done
```

Vous pouvez alors utiliser `rm` de manière normale.

21.7.11 Processus persistant en tâche de fond.

Imaginons que nous voulions lancer le terminal `rxtv` en arrière-plan. Ceci peut être réalisé ainsi :

```
rxtv &
```

Cependant, une fois lancé, `rxtv` a encore sa sortie connectée au shell et reste un processus-fils du shell. Or, si le shell dont dépend `rxtv` meurt, il emporte ses processus-fils. Par ailleurs, `rxtv` peut aussi mourir de lui-même en tentant de lire ou d'écrire dans un terminal qui n'existe pas sans le shell-parent. Essayons donc à présent :

```
{rxtv >/dev/null 2>&1 </dev/null &} &
```

Cette technique est connue sous le nom de *forking twice*[14] ou *redirection d'un terminal vers dev null*. Le shell est donc informé en matière de processus-fils mais pas concernant le processus résultant du *forking twice*. Dès lors, nous pouvons créer un processus-démon propre avec la commande ci-dessus.

Par conséquent, il est possible de créer un démon qui redémarre au cas où il lui arriverait de mourir. Bien qu'une telle fonctionnalité soit plus aisée à réaliser en **C** (dont nous verrons un aperçu au chapitre 23), vous pouvez atteindre cet objectif en bash avec :

```
{ { while true ; do rxtv ; done ; } >/dev/null 2>&1 </dev/null & } &
```

Vous pouvez observer l'effet de ces astuces en lançant :

```
ps awwwxf
```

Par ailleurs, la commande `nohup` (voir `nohup(1)`) permet d'exécuter un programme en arrière-plan en le rendant insensible aux déconnexions. Ainsi, un utilisateur peut installer un programme appelé, par exemple, `alpha` dans son espace de travail avec la commande `make` :

```
nohup make alpha
```

et quitter sa session alors que `make` procède à la compilation/installation du programme `alpha` de l'utilisateur (la commande `make` est discutée à la section 23.4).

21.7.12 Traitement de la liste de processus.

La commande qui suit exploite l'option de format personnalisé pour imprimer chaque attribut concevable d'un processus :

```
ps awwwxo %cpu,%mem,alarm,args,blocked,bsdstart,bsdtime,c,caught,cmd,comm,\
command,cputime,drs,dsiz,egid,egroup,eip,esp,etime,euid,euser,f,fgid,fgroup,\
flag,flags,fname,fsgid,fsgroup,fsuid,fsuser,fuid,fuser,gid,group,ignored,\
intpri,lim,longtname,lstart,m_drs,m_trs,maj_flt,majflt,min_flt,minflt,ni,\
nice,nwchan,opri,pagein,pcpu,pending,pgid,pgrp,pid,pmem,ppid,pri,rgid,rgroup,\
rss,rssize,rsz,ruid,ruser,s,sess,session,sgi_p,sgi_rss,sgid,sgroup,sid,sig,\
sig_block,sig_catch,sig_ignore,sig_pend,sigcatch,sigignore,sigmask,stackup,\
start,start_stack,start_time,stat,state,stime,suid,suser,svgid,svgroup,svuid,\
svuser,sz,time,timeout,tmout,tname,tpgid,trs,trss,tsiz,tt,tty,tty4,tty8,ucomm,\
uid,uid_hack,uname,user,vsize,vsz,wchan
```

Il est préférable de créer un tube pour récupérer le résultat de la commande `ps`, et le visualiser dans un éditeur de texte brut. De manière plus intéressante,

[14] NdT : sous UNIX, le terme *forking* désigne une instruction ou une commande (parfois nommée *primitive*) créant un nouveau processus à partir d'un autre sans détruire celui-ci. On pourrait donc tenter de traduire *forking twice* par "dédoublement d'instruction(s)".

la commande awk peut afficher l'ID d'un processus avec :

```
ps awwx | grep 'htt[p]d' | awk '{print $1}'
```

Ainsi, tous les processus ayant httpd dans le nom de commande ou la ligne de commande seront affichés. Ce filtre est utile pour arrêter netscape :

```
kill -9 `ps awx | grep 'netsc[a]pe' | awk '{print $1}'`
```

(Remarquez que [a] est l'expression rationnelle empêchant grep de se trouver lui-même dans la liste des processus.

Parmi les options utiles de la commande ps, nous trouvons :

```
ps awwxf
ps awwxl
ps awwxv
ps awwxu
ps awwxs
```

L'option -f est la plus utile pour mettre en évidence les relations entre processus-père et processus-fils. Elle signifie forest, et affiche l'arborescence des processus principaux. Ainsi, dans l'exemple qui suit, un écran **X** avec deux fenêtres est utilisé :

```
PID        TTY      STAT   TIME     COMMAND
1          ?        S      0 :05    init[5]
2          ?        SW     0 :02    [kflushd]
3          ?        SW     0 :02    [kupdate]
4          ?        SW     0 :00    [kpiod]
5          ?        SW     0 :01    [kswapd]
6          ?        SW<    0 :00    [mdrecoveryd]
262        ?        S      0 :02    syslogd -m 0
272        ?        S      0 :00    klogd
341        ?        S      0 :00    xinetd -reuse -pidfile /var/run/xinetd.pid
447        ?        S      0 :00    crond
480        ?        S      0 :02    xfs -droppriv -daemon
506        tty1     S      0 :00    /sbin/mingetty tty1
507        tty2     S      0 :00    /sbin/mingetty tty2
508        tty3     S      0 :00    /sbin/mingetty tty3
509        ?        S      0 :00    /usr/bin/gdm -nodaemon
514        ?        S      7 :04    \_ /etc/X11/X -auth /var/gdm/ :0.Xauth :0
515        ?        S      0 :00    \_ /usr/bin/gdm -nodaemon
524        ?        S      0 :18       \_ /opt/icewm/bin/icewm
748        ?        S      0 :08          \_ rxtv -bg black -cr green -fg whi
749        pts/0    S      0 :00          |   \_ bash
5643       pts/0    S      0 :09          |      \_ mc
5645       pts/6    S      0 :02          |         \_ bash -rcfile .bashrc
25292      pts/6    R      0 :00          |             \_ ps awwxf
11780      ?        S      0 :16          \_ /usr/lib/netscape/netscape-commu
11814      ?        S      0 :00              \_ (dns helper)
15534      pts/6    S      3 :12    cooledit -I /root/.cedit/projects/Rute
15535      pts/6    S      6 :03    \_ aspell -a -a
```

L'option `u` indique le format d'utilisateur, et les autres options montrent la mémoire virtuelle, les signaux et le format long.

21.8 Initialisation du shell.

Dans cette section, nous allons voir ce qui se produit juste après la connexion dans une session et comment modifier les paramètres associés à cette étape.

Le shell interactif invoqué après la connexion est le shell specifié dans le dernier champ de l'entrée de l'utilisateur, tel qu'indiqué dans le fichier `/etc/passwd`. Une fois l'utilisateur authentifié, le programme `login` invoque le shell en plaçant un signe - devant le nom de la commande (ceci indique au shell quel est le shell de connexion). Ceci signifie que le shell lit et exécute plusieurs scripts pour initialiser son environnement. Dans le cas de `bash`, les fichiers sont lus dans cet ordre : `/etc/profile`, `~/.bash_profile`, `~/.bash_login` et `~/.profile`. De plus, un shell interactif qui n'est pas le shell de connexion lit également `~/.bashrc`. Notez que les shells traditionnels `sh` lisent seulement `/etc/profile` et `~/.profile`.

21.8.1 Personnalisation de PATH et LD_LIBRARY_PATH.

Les administrateurs ont la possibilité de personnaliser les variables d'environnement en modifiant les scripts de démarrage. Prenez, par exemple, le cas classique d'une arborescence sous /opt. Souvent, un paquet tel que /opt/OpenOffice.org ou /etc/oracle requerrons que les variables PATH et LD_LIBRARY_PATH soient ajustées de concert. Dans le cas de RedHat, un script tel que :

```
for i in /opt/*/bin /usr/local/bin ; do
   test -d $i || continue
   exho $PATH | grep -wq "$i" && continue
   PATH=$PATH :$i
   export PATH
done

if test `id -u` -eq 0 ; then
   for i in /opt/*/sbin /usr/bin/local/sbin ; do
      test - d $i || continue
      echo $PATH | grep -wq "$i" && continue
      PATH=$PATH :$i
      export PATH
   done
fi

for i in /opt/*/lib /usr/local/lib ; do
   test -d $i || continue
   echo $LD_LIBRARY_PATH | grep -wq "$i" && continue
   LD_LIBRARY_PATH=$LD_LIBRARY_PATH :$i
   export LD_LIBRARY_PATH
done
```

peut être placé dans /etc/profile.d/mes_locales.sh avec les droits d'exécution, ce qui fixera toutes les précautions à prendre vis-à-vis de tout ce qui est installé sous /opt/ ou /usr/local/. En ce qui concerne Debian, le script peut être insérer directement dans /etc/profile.

La section 24.3 contient des informations relatives à LD_LIBRARY_PATH.

Quoique cette dernière phrase ne soit pas directement liée à ce qui vient d'être discuté, pensez à éditer votre fichier /etc/man.config pour ajouter les chemins des pages de man qui apparaissent après toute installation sous /opt/.

21.9 Verrouillage de fichiers.

Régulièrement, il s'avère utile qu'un processus ait un *accès exclusif* à un fichier ; c'est-à-dire qu'un seul processus ne puisse accéder à ce fichier. Considérons un répertoire[15] contenant des courriels : si deux processus étaient capables d'y écrire simultanément, ce répertoire deviendrait corrompu. Par ailleurs, il arrive que nous voulions nous assurer qu'un programme ne puisse être exécuté à deux reprises en même temps. Pour cela, on pratique le "verrouillage" (ou

[15] NdT : le texte en anglais utilise le terme *folder*. Comme cela est indiqué au paragraphe 21.9.3, Paul Sheer fait la nuance entre *folder* et *directory*.

locking) de fichiers.

Dans le cas d'un répertoire de courriels, si un fichier est en cours d'écriture, il ne faut pas qu'un autre processus puisse lire ce fichier ou y écrire : il est donc nécessaire de mettre en place un *verrouillage en écriture* sur ce fichier. Cependant, si ce dernier est en cours de lecture, aucun autre processus ne devrait tenter de le lire : il est asticieux de ne mettre en place qu'un *verrouillage en lecture*. Les verrouillages en écriture sont appelés *verrouillages exclusifs* ; ceux en lecture *verrouillages partagés*. Pour des raisons de simplicité, le verrouillage exclusif est souvent préféré.

La mise en place d'un verrouillage consiste simplement à créer un fichier temporaire pour indiquer aux autres processus d'attendre avant de tenter un accès. UNIX présente également des fonctions internes plus sophistiquées.

21.9.1 Verrouiller un fichier de la boîte à messages.

Il existe quatre méthodes usuelles de verrouillage de fichiers [Les sources d'`exim` semblent indiquer une recherche approfondie dans ce domaine, ce que nous allons suivre] :

1. le verrouillage de fichier "*dot lock*". Dans ce cas, un fichier temporaire est créé avec le même nom que celui du répertoire de courriel et l'extension `.lock` est ajoutée. Aussi longtemps que ce fichier existe, aucun programme n'essaie d'accéder au répertoire. Il s'agit d'un cas de verrouillage exclusif. Il est aisé d'écrire un script de shell pour réaliser ce type de verrouillage de fichier.
2. le verrouillage "MBX". Il est analogue au premier, mais un fichier temporaire est créé dans `/tmp`. Il s'agit aussi d'un verrouillage exclusif.
3. le verrouillage `fnctl`. Les bases de données requièrent que des zones d'un fichier soient inaccessibles. `fnctl` est un appel système utilisable dans les programmes en **C**.
4. le verrouillage `flock` est similaire à `fnctl` mais il agit sur la totalité d'un fichier.

La fonction shell qui suit assure un verrouillage correct des fichiers d'un boîte de courriel :

```
function verrouillage_de_fichiers ()
{
     TEMPFILE="$1.$$"
     LOCKFILE="$1.lock"
     echo $$ > $TEMPFILE 2>/dev/null || {
           echo "You don't have permission to access `dirname $TEMPFILE`"
           return 1
     }
     ln $TEMPFILE $LOCKFILE 2>/dev/null && {
           rm -f $TEMPFILE
           return 0
     }
```

```
    STALE_PID= `< $LOCKFILE`
    test ``$STALE_PID'' -gt ``0'' >/dev/null || {
           return 1
    }
    kill -0 $STALE_PID 2>/dev/null && {
           rm -f TEMPFILE
           return 1
    }
    rm $LOCKFILE 2>/dev/null && {
           echo ``Removed stale lock file of process $STALE_PID''
    }
    ln $TEMPFILE $LOCKFILE 2>/dev/null && {
           rm -f $TEMPFILE
           return 0
    }
    rm -f $TEMFILE
    return 1
}
```

(Incidemment, remarquez que `< $LOCKFILE` a été avantageusement utilisé –pour sa rapidité– au lieu de `cat $LOCKFILE`)

Vous pouvez inclure la fonction qui est décrite ci-dessus dans divers scripts en vue d'effectuer du verrouillage de fichiers. Par exemple :

```
# wait for a lock
until verrouillage_de_fichiers /etc/passwd ; do
      sleep 1
done

# The body of the programme might go there
# [...]

# Then to remove the lock,
rm -f /etc/passwd.lock
```

Naturellement, ce script n'a d'intérêt qu'académique mais, il présente des caractéristiques intéressantes. Par exemple, la fonction `ln` est utilisée pour assurer le caractère exclusif du verrouillage. `ln` est une des fonctions d'UNIX qu'on appelle *atomiques* : dans le cas présent, cela signifie qu'un seul lien du même nom peut exister et que sa création exclut l'éventualité qu'un autre programme pense qu'il a réussi à créer le même lien. On pourrait croire naïvement que le programme que voici est suffisant pour effectuer un verrouillage de fichier :

```
function verrouillage_de_fichiers ()
{
     LOCKFILE="$1.lock"
     test -e $LOCKFILE && return 1
     touch $LOCKFILE
     return 0
```

```
}
```

Cependant, imaginons que deux programmes en cours d'exécution, exécutent la ligne 4 en même temps. Alors, les *deux* programmes –*simultanément*– penseront que le verrouillage n'existe pas et exécuteront la ligne 5. La conséquence sera que les deux programmes créeront le fichier verrouillé – ce qui bien sûr n'est pas souhaité.

La commande `kill` est alors utile pour vérifier si un processus est en cours. Le fait d'envoyer le signal `0` ne fera rien au processus ; toutefois le signal échouera si le processus n'existe pas. Cette technique peut être utilisée pour supprimer le verrouillage d'un processus qui a disparu avant le verrouillage lui-même : ce qu'on nomme en anglais sous le terme *stale lock*.

21.9.2 Verrouiller sur NFS.

Le script décrit dans la section qui précède *ne fonctionnera pas* si votre système de fichier est monté via NFS (*network file system* ; voir le chapitre 29). Il y a deux raisons : (i) le script exploite le PID du processus, qui n'est pas accessible lors d'un partage NFS ; (ii) la seconde raison -moins évidente- est que la fonction `ln` ne travaillera pas normalement via NFS. Il sera nécessaire d'utiliser la commande `stat` pour vérifier que le compteur de liens a augmenté d'une unité.[16]

Les commandes `lockfile` (du paquet `procmail`) et `muttdot_lock` (associée au client courriel `mutt`) effectue du verrouillage selon une méthode proche de celle discutée au paragraphe précédent. Cependant, elles ne tirent pas parti du PID. Donc, il ne leur est pas possible de détecter un *stale lock file*. Par exemple, pour effectuer une recherche dans votre boîte de courriel, vous pouvez exécuter :

```
lockfile /var/spool/mail/mary.lock
grep freddy /var/spool/mail/mary
rm -f /var/spool/mail/mary.lock
```

Cette séquence vous assure de réaliser une recherche dans une boîte non-corrompue même si `/var` est impliqué dans un partage NFS.

21.9.3 Verrouillage de fichiers ou de répertoires.

Le verrouillage de fichiers est un casse-tête pour les développeurs. Le problème avec UNIX est le suivant : nous pensons intuitivement en termes de verrouillage de *fichiers*, ce qui signifie en réalité "verrouillage des noms de fichiers dans un répertoire". Le verrouillage des fichiers *per se* ne devrait être opéré que sur des fichiers permanents tels que les fichiers d'une base de données. Concernant les fichiers de messageries et `passwd`, nous nous adressons à du verrouillage de répertoires [selon mon propre terme], ou dit autrement, à l'accès exclusif d'un processus à une entrée particulière d'un répertoire. L'absence d'une telle fonctionnalité constitue sans doute une lacune grave d'UNIX (étant donné que cela concerne à la fois le noyau, NFS, et les extensions de bibliothèques **C**, il est probable que la correction ne sera pas rapide).

[16] NdT : au paragraphe 21.7.10, nous avons vu comment extraire un champ en combinant dans un pipeline les commandes `stat`, `grep` et `awk`.

21.9.4 Verrouillage à l'intérieur de programmes en C.

Cette thématique est très probablement hors de propos dans ce document, si ce n'est que vous devriez consulter le code source de paquets réputés de bonne qualité plutôt que d'essayer de construire vos propres schémas de verrouillage.

Chapitre 22

Services du système et lpd.

Ce chapitre couvre une large gamme de notions relatives à la manière dont les services UNIX fonctionnent.

Chaque fonction d'un système UNIX est fournie par un paquet logiciel. Par exemple, le courriel est souvent traité par le serveur `sendmail` ou un paquet ayant des fonctions analogues (par exemple `postfix`) ; le web est traité par le serveur `apache`.

A présent, nous allons examiner comment obtenir, installer et configurer un paquet logiciel en prenant pour modèle `lpd`. Vous pourrez alors appliquer la connaissance acquise pour l'extrapoler à d'autres cas. Les notions vues dans le présent chapitre seront supposées acquises pour aborder les chapitres suivants. Ce qui est vu ici concerne uniquement l'installation et la gestion de l'impression.

22.1 Utilisation de `lpr`.

L'impression sous UNIX avec une machine correctement configurée se résume à la commande `lpr -Plp <nom_de_fichier>` (ou `cat <nom_de_fichier> | lpr -Plp`). Le terme "`lp`" dans `-Plp` désigne le nom de la file (*queue* en anglais) d'impression sur l'ordinateur local à partir duquel vous souhaitez imprimer. Vous pouvez omettre ce terme si vous effectuez une impression *via* la file par défaut (c'est-à-dire la première imprimante de votre liste). Une file (ou *queue*) est associée à une imprimante physique, de sorte que les utilisateurs savent toujours où le papier imprimé sort. Par convention, les files sont notés `lp`, `lp0`, `lp1`, et ainsi de suite. Grâce au réseau, tout fichier créé localement peut être redirigé vers la file d'impression d'une autre machine.

La commande `lprm` supprime les travaux en attente d'une file d'impression ; `lpq` affiche les travaux en cours, à l'écran.

Le service qui permet aux commandes d'être exécutées est `lpd`. Le programme utilisateur `lpr` envoye une requête au processus en tâche de fond `lpd` (éventuellement via le réseau) et lui envoye le travail à imprimer. Etant donné que plusieurs requêtes peuvent arriver simultanément sur la même machine, le document à imprimer est enregistré dans un fichier temporaire du répertoire de spoule (*spool*). `lpd` y met, filtre (il s'agit d'un formatage ayant lieu avant l'impression) et analyse le travail jusqu'à ce que ce dernier apparaisse dans la corbeille d'impression.

L'impression constitue un exemple du paradigme *client/serveur* d'UNIX. Le processus en arrière-plan `lpd` est le *serveur* et il est initialisé par le superutilisateur (*root*). Les commandes `lpr`, `lpq` et `lprm` sont des programmes clients exécutés par les *utilisateurs*.

22.2 Chargement et installation.

La discussion qui suit devrait dissiper les interrogations que voici : "où puis-je obtenir le service ou le paquet logiciel *xxx* ?" et "comment installer un service et l'activer ?". La gestion complète des paquets interviendra à la section 25.2, mais dans ce chapitre, nous voyons comment utiliser les gestionnaires de paquets en terme de services.

Supposons que nous ne sachions rien du service d'impression si ce n'est qu'il concerne un fichier `/usr/local/lpd`. Il nous faut avant tout utiliser le gestionnaire de paquets pour déterminer d'où vient ce paquet (la commande Debian est donnée entre parenthèses) :

```
rpm -qf /usr/sbin/lpd
( dpkg -S /usr/sbin/lpd )
```

Ceci devrait retourner `lpr-0.nn-n` (pour une RedHat 6.2, ou `LPRng-n.nn-n` pour RedHat 7.0, ou encore `lpr` sur Debian). Sur RedHat, vous pourriez être amené à essayer cette commande sur une autre machine car `rpm` ne permet pas de donner de renseignements sur les paquets non-installés. Si vous souhaitez savoir si un paquet dont le nom contient les lettres `lpd` est installé, il convient d'exécuter les commandes suivantes :

```
rpm -qa | grep -i lpd
( dpkg -l '*lpd*' )
```

Si le paquet n'est pas présent, le fichier se trouve sur le CD-ROM et peut aisément être installé avec (RedHat 7.0 entre parenthèses et Debian entre accolades) :

```
rpm -i lpr-0.50-4.i386.rpm
( rpm -i LPRng-3.6.24-2 )
{ dpkg -i lpr_0.48-1.deb }
```

Des informations supplémentaires sont données au chapitre 25.

La liste des fichiers associés au paquet `lpr` (aisément obtenue à l'aide la commande : `rpm -ql lpr` ou `dpkg -L lpr`) est approximativement celle-ci :

/etc/init.d/lpd	/usr/share/man1/lprm.1.gz
/etc/cron.weekly/lpr	/usr/share/man5/printcap.5.gz
/usr/sbin/lpf	/usr/share/man8/lpc.8.gz
/usr/sbin/lpc	/usr/share/man8/lpd.8.gz
/usr/sbin/lpd	/usr/share/man8/pac.8.gz
/usr/sbin/pac	/usr/share/man8/lpf.8.gz
/usr/bin/lpq	/usr/share/doc/lpr/README.Debian

/usr/bin/lpr	/usr/share/doc/lpr/copyright
/usr/bin/lprm	/usr/share/doc/lpr/examples/printcap
/usr/bin/lptest	/usr/share/doc/lpr/changelog.gz
/usr/share/man/man1/lpr.1.gz	/usr/share/doc/lpr/changelog.Debian.gz
/usr/share/man/man1/lptest.1.gz	/var/spool/lpd/lp
/usr/share/man/man1/lpq.1.gz	/var/spool/lpd/remote

22.3 LPRng et lpr-0.*nn*.

Depuis RedHat 7.0, le paquet LPRng remplace avantageusement le logiciel lpr-0.nn devenu obslolète, que Debian et d'autres distributions utilisent encore. LPRng est un logiciel plus moderne et plus complet. Il contient le même fichier /etc/printcap et des binaires indentiques à ceux de RedHat 6.2. La différence majeure se trouve dans les fichiers de contrôle créés dans les répertoires de spoule et dans un mécanisme de contrôle d'accès différent (voir ci-dessous). Notez que LPRng présente des exigences strictes en termes de droits sur les répertoires de spoule. Il n'est pas trivial d'en effectuer l'installation à partir des sources.

22.4 Eléments de paquet.

Les fichiers d'un paquet logiciel peuvent être vaguement regroupés selon leur fonctionnalité. Dans cette section, nous expliquons chacun des composants, en prenant lpr comme exemple. Reportez-vous à la liste des fichiers de la section 22.2.

22.4.1 Fichiers de documentation.

La documentation devrait être votre premier centre d'intérêt. Les pages de man ne seront probablement pas la seule documentation accessible. Ci-dessus, nous avons vu que lpr ne présente pas beaucoup d'information dans /usr/share/doc. Cependant, les autres paquets sont associés à un manuel d'utilisation très complet. Essayez par exemple, rpm -ql apache (voir /home/httpd/-html/manual ou /var/www/html/manual/) ou rpm -ql wu-ftpd qui devrait montrer des informations dans /usr/doc/wu-ftpd-?.?.?.

22.4.2 Pages de web, listes de diffusion, points de chargement.

Chaque paquet contient une information sur l'équipe qui maintient une page web s'y rapportant. Dans le cas de lpd, cependant, le code est assez ancien et les différents distributeurs de CDs en réalisent la maintenance eux-mêmes. LPRng constitue un meilleur exemple. Visitez la *page web de LPRng* http ://www.astart.com/LPRng.html avec votre navigateur web. Vous pourrez y voir la liste des développeurs, les listes de diffusion, et les points de téléchargement. Si un paquet donné vous intéresse tout particulièrement, vous devriez rapidement devenir familier avec ses ressources. Les pages de web bien tenues contiennent aussi de la documentation en cas de problèmes et une FAQ (*Foire aux questions* appelée en anglais *Frequently Asked Questions*). Certains sites

contiennent même des archives de leur liste de diffusion. Notez que certaines pages web sont orientées vers des revendeurs de CDs qui essayent de créer leur propre distribution et, par conséquent, elles ne présentent pas les paquets à télécharger que l'utilisateur débutant peut installer aisément.

22.4.3 Programmes utilisateurs.

Les programmes utilisateurs se trouvent dans un des répertoires `bin`. En l'occurrence, pour `lpq`, `lpr` et `lprm`, il s'agit de `/usr/bin` (voir le résultat des commandes `rpm -ql nom_de_paquet` ou `dpkg -L nom_de_paquet`).

22.4.4 Démons et programmes administrateurs.

La commande administrateur et le démon sont localisés dans un répertoire `sbin` comme l'indique le résultat de la commande `rpm -ql nom_de_paquet` (ou `dpkg -L nom_de_paquet`). Pratiquez de même pour localiser les pages de man. Notez que le seul démon parmi les programmes dont nous discutons est `lpd` lui-même (qui est exécuté en tâche de fond). Ce démon constitue le coeur du paquet d'impression.

22.4.5 Fichiers de configuration.

Le fichier `/etc/printcap` contrôle `lpd`. Pour la plupart, les services présentent un fichier de contrôle dans `/etc`. Remarquez que `printcap` est un fichier texte que lit `lpd` au démarrage. La configuration de tout service commence par la modification du fichier de configuration qui lui correspond. Plusieurs outils de configuration graphique sont disponibles afin d'éviter cette contrainte. C'est le cas de `linuxconf` –qui est général– et de `printtool` qui se rapporte spécifiquement à `printcap`. En fait, ces outils effectuent de manière "silencieuse" la modification des fichiers de configuration des services.

En raison du fait que l'impression est un constituant de base très général pour le système sur lequel vous travaillez, `printcap` n'est pas fourni par le paquet `lpr`. Essayez la commande `rpm -qf /etc/printcap` : cela vous retournera `setup-2.3.4-1` tandis que `dpkg -S /etc/printcap` montre que ce fichier n'est pas présent. Il est une partie intégrante de votre système Debian.

22.4.6 Fichiers d'initialisation de services.

Les fichiers de `/etc/rc.d/init.d` (ou `/etc/init.d`) sont les scripts de démarrage et d'arrêt permettant à `lpd` d'être lancé/arrêté au démarrage/arrêt de la machine. Vous pouvez démarrer `lpd` avec la commande suivante :

```
/usr/bin/lpd
```

mais il est préférable d'utiliser les scripts :

```
/etc/rc.d/init.d/lpd start
/etc/rc.d/init.d/lpd stop
```

(ou /etc/init.d/lpd). Le script a aussi d'autres usages :

```
/etc/rc.d/init.d/lpd status
/etc/rc.d/init.d/lpd restart
```

(ou /etc/init.d/lpd).

Pour s'assurer que lpd est bien lancé au démarrage, vous pouvez vérifier qu'il y a un lien symbolique sur le niveau d'exécution approprié. Les liens symboliques peuvent être visualisés en effectuant :

```
ls -al ‘find /etc -name ’*lpd*’‘
find /etc -name ’*lpd*’ -ls
```

ce qui donne :

```
-rw-r- -r- - 1 root    root 17335 Sep      25   2000       /etc/lpd.conf
-rw-r- -r- - 1 root    root 10620 Sep      25   2000       /etc/lpd.perms
-rwxr-xr-x   1 root    root  2277 Sep      25   2000       /etc/rc.d/init.d/lpd
lrwxrwxrwx   1 root    root    13 Mar      21 14 :03       /etc/rc.d/rc0.d/K60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root    root    13 MAr      21 14 :03       /etc/rc.d/rc1.d/K60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root    root    13 Mar      21 14 :03       /etc/rc.d/rc2.d/S60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root    root    13 Mar      21 14 :03       /etc/rc.d/rc3.d/S60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root    root    13 Mar      21 14 :03       /etc/rc.d/rc4.d/S60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root    root    13 Mar      21 14 :03       /etc/rc.d/rc5.d/S60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root    root    13 Mar      21 14 :03       /etc/rc.d/rc6.d/K60lpd -> ../init.d/lpd
```

Le “3” dans rc3.d est ce que nous recherchons. Le fait d'avoir S601lpd avec un lien symbolique à lpd sous le niveau d'exécution rc3.d signifie que lpd est lancé lorsque le système entre dans le niveau d'exécution 3 (ou *run level 3*), qui est l'état du système en temps normal (mode console).

Notez bien que sur RedHat, la commande setup a une option System Services. La liste des Services vous autorise à gérer les services qui doivent être initialisés lors de la procédure de démarrage de la machine ; c'est ainsi que les liens symboliques sont établis de manière automatique. Concernant Debian, vérifiez la page de man pour la commande update-rc.d.

Des informations supplémentaires sur la séquence de démarrage d'une machine sont présentées au chapitre 33.

22.4.7 Fichiers de spoule[1].

Les services comme lpd, innd, sendmail et uucp créent des fichiers au cours du traitement de chaque requête. Ceux-ci sont appelés fichiers de spoule (*spool files*) et ils sont stockés dans le répertoire /var/spool, usuellement pour être analysés et ensuite effacés de manière séquentielle. lpd a un répertoire de spoule /var/spool/lpd qui, sans doute, a été créé lors de l'installation. Vous pouvez créer des répertoires de spoule pour deux imprimantes dans l'exemple ci-dessous à l'aide de la commande :

[1] NdT :il faut comprendre ce terme comme un magasin de dépôt de données.

```
mkdir -p /var/spool/lpd/lp /var/spool/lpd/lp0
```

22.4.8 Fichiers de journalisation (*logs*).

UNIX est caractérisé par une politique très stricte consistant à ne pas rapporter les message d'erreur dans l'interface utilisateur étant donné que sur un système donné, il se pourrait qu'il n'y ait pas d'utilisateur connecté temporairement. Vu que les messages d'erreur sont envoyés à l'écran du terminal, ceux-ci ou les messages d'information produits par des commandes non-interactives sont basculés dans les fichiers du répertoire `/var/log`.

Un fichier de journalisation (ou fichier log) est un fichier contenant du texte, qui enregistre à la queue leu-leu et de manière continue des messages en provenance d'un démon. Le répertoire usuel des fichiers de journalisation est `/var/log`. Il contient les messages en provenance du noyau et des services primaires. Si un service est susceptible d'engendrer des fichiers de journalisation de grande taille (pensez au cas des accès au web, avec des milliers de tentatives par heure), il utilise son propre fichier de journalisation. Par exemple, `sendmail` emploie `/var/log/maillog`. En réalité, pour revenir à `lpd`, ce service n'a pas de fichier de journalisation en soi ; c'est un de ses défauts.

Pour visualiser le fichier de journalisation du système, utilisez la commande `tail` avec l'option `f` (pour follow) :

```
tail -f /var/log/messages
tail -f /var/log/syslog
```

Le redémarrage du service `lpd` donne un message tel que celui-ci :

```
Jun 27 16 :06 :43 cericon lpd : lpd shutdown succeeded
Jun 27 16 :06 :45 cericon lpd : lpd startup succeeded
```

22.4.9 Rotation des journaux.

Les fichiers de log sont traités tour-à-tour journalièrement ou hebdomadairement par le paquet `logrotate`. Son fichier de configuration est `/etc/logrotate.conf`. Pour chaque paquet qui produit un fichier de journalisation, il y a un fichier de configuration sous `/etc/logrotate.d/`. Il est facile d'écrire son propre fichier en s'aidant d'un modèle existant. Le terme *rotation* signifie que les fichiers de journalisation sont renommés avec une extension `.1` et sont tronqués à une taille nulle. Le service est averti par `logrotate`, parfois avec un signal `SIGHUP`. Votre répertoire `/var/log` contient vraisemblablement d'anciens fichiers de journalisation dont l'extension est `.2`, `.3`, etc. L'idée de la rotation sur les fichiers de journalisation est d'empêcher ceux-ci de s'allonger indéfiniment.

22.4.10 Variables d'environnement.

En majorité, les commandes associées aux services exploitent certaines variables d'environnement. Elles peuvent être définies dans les scripts de démarrage du shell, comme d'habitude. Concernant `lpr`, si aucune imprimante n'est spécifiée en ligne de commande, la variable d'environnement `PRINTER` détermine

la file d'impression par défaut. Par exemple, `export PRINTER=lp1` force l'usage de l'imprimante `lp1`.

22.5 Le fichier `printcap` en détail.

Le fichier `printcap` (*printer capabilities*) est similaire au fichier `termcap` (*terminal capabilities*). Il en est d'ailleurs inspiré. L'opération de configuration d'une imprimante consiste à ajouter ou à supprimer du texte dans ce fichier. `printcap` contient une liste d'entrée ligne par ligne, chacune concernant une imprimante. Les lignes peuvent être discontinuée par un `\` précédant une nouvelle ligne. Voici un exemple de fichier `printcap` dans le cas de deux imprimantes :

```
lp :\
      :sd=/var/spool/lpd/lp :\
      :mx#0 :\
      :sh :\
      :lp=/dev/lp0 :\
      :if=/var/spool/lpd/lp/filter :
lp0 :\
      :sd=/var/spool/lpd/lp0 :\
      :mx#0 :\
      :sh :\
      :rm=edison :\
      :rp :lp3 :\
      :if=/bin/cat :
```

Les imprimantes sont nommées dans le premier champ : dans le cas présent, `lp` est la première imprimante et `lp0` est la seconde. Chacune d'elles décrit un périphérique physique différent avec sa propre file. L'imprimante `lp` devrait toujours être citée en premier et devrait être la première utilisée si aucune n'autre n'est spécifiée. En l'occurrence, `lp` se rapport à la première imprimante locale sur le périphérique `/dev/lp0` (premier port parallèle). `lp0` se rapporte à une file d'impression distante sur la machine `edison`.

Le fichier `printcap` possède une page de `man` détaillée. Cependant, les champs suivants sont ceux dont vous aurez, de loin, le plus besoin :

sd le répertoire de spoule. Ce dernier contient les fichiers d'état et de spoule.

mx taille maximale de fichier. Dans l'exemple précédent : illimité.

sh supprime les en-têtes. L'en-tête consiste en une série de lignes d'information imprimées avant et après le travail d'impression. Cette option devrait toujours être désactivée.

lp périphérique d'impression en ligne.

if filtre d'entrée. Il s'agit d'un script exécutable dans lequel les données d'imprimante sont fournies à l'aide d'un tube. La sortie de ce script alimente directement l'imprimante ou la machine distante. Ce filtre traduit la sortie d'une application en un code interprétable par l'imprimante (code natif).

rm machine distante (`r`emote `m`achine). Si la file d'impression n'est pas locale, ceci est le nom de la machine.

rp nom de la file d'imprimante distante. La machine distante possède son propre fichier `printcap` avec éventuellement plusieurs imprimantes qui y sont définies. rp indique l'imprimante à utiliser.

22.6 PostScript et filtre d'impression.

Sur UNIX, le format de référence pour toutes les impressions est le PostScript. Les fichiers PostScript `.ps` sont de type graphique et représentent un texte, des lignes quelconques et/ou des images de manière à ce que leur taille soit ajustable. En réalité, PostScript est un langage de programmation destiné à tracer des éléments sur une page; donc, les filchiers `.ps` sont réellement des programmes Postscript. La dernière ligne de tout fichier Postscript est toujours une `showpage`, ce qui indique que les opérations de traçage sont terminées et que la page peut être affichée/imprimée. En conséquence, il est aisé de visualiser le nombre de pages se trouvant dans un fichier PostScript en effectuant un `grep` sur la chaîne de caractères `showpage`.

La méthode pour imprimer sur UNIX consiste toujours à convertir le document à imprimer en un fichier PostScript. Les fichers `.ps` peuvent être visualiser grâce à un émulateur PostScript tel que le programme `gv` (GhostView). Le programme appelé `gs` (GhostScript) est l'utilitaire permettant de convertir un fichier Postscript dans un format manipulable par l'imprimante. PostScript est un langage qui peut être facilement "passé" à tout imprimante. Il existe des imprimantes "PostScript" qui sont des périphériques interprétant directement les fichiers PostScript. Cependant, elles sont relativement coûteuses et la plupart des imprimantes comprennent seulement le dialecte PCL (printer control language) et quelques autres formats.

En résumé, les centaines de formats différents de graphiques et de textes possèdent un utilitaire qui leur permet d'être convertis en Postscript. Après cette opération, `gs` convertit le fichier résultant afin qu'il soit imprimable par les centaines de modèles d'imprimantes. [Au moment où ces lignes sont écrites, il faut signaler qu'il existe de nombreux types d'imprimantes qui ne gèrent pas `gs`. Ceci est dû au fait que les fabricants refusent d'émettre les spécifications de leurs protocoles de communication propriétaires]. Le filtre d'impression est le cheval de labeur réalisant l'ensemble de ces opérations.

En général, les applications émettent de manière convenable des fichiers PostScript lors des impressions. Par exemple, le menu de sélection de `netscape`

présente une fenêtre telle que :

qui envoye du PostScript vers l'entrée standard (stdin) de `lpr`. Toutes les applications n'ayant pas leurs propres pilotes d'impression procèdent de cette manière. Ceci signifie qu'en général, nous pouvons compter sur le fait que le filtre

d'impression reçoit du PostScript. En revanche, `gs` peut convertir du PostScript pour toutes les imprimantes, de telle sorte qu'il ne reste plus qu'à déterminer les options en ligne de commande.

Si vous avez choisi "Print To : File", vous pouvez visualiser le résultat avec le programme `gv` qui affiche une prévisualisation (*print preview*) à l'écran. Sur UNIX, la plupart des applications des bureaux n'ont pas leurs propres utilitaires de prévisualisation car les imprimantes PostScript sont elle-mêmes émulées par `gv`.

Remarquez que les programmes de filtre ne devraient pas être utilisés avec des filtres distants. Les files d'impression distantes peuvent envoyer leurs fichiers PostScript "tels quels" avec : `if=/bin/cat :` (comme dans l'exemple du fichier `printcap` vu ci-dessus). De cette manière, la machine connectée au périphérique doit être la seule configurée spécialement à cette fin.

Le programme de filtre que nous allons utiliser pour la file d'impression est un script de shell `/var/spool/lpd/lp/filter`. Créez le filtre de cette façon :

```
touch /var/spool/lpd/lp/filter
chmod a+x /var/spool/lpd/lp/filter
```

et modifiez le de manière à ce qu'il contienne le code suivant :

```
#!/bin/bash
cat | gs -sDEVICE=ljet4 -sOutputFile=- -sPAPERSIZE=a4 -r600x600 -q -
exit 0
```

L'option `-sDEVICE` décrit l'imprimante. Dans cet exemple, il s'agit d'une Hewlett Packard LaserJet 1100. Beaucoup d'imprimantes ont des formats similaires et compatibles. Donc, il y a beaucoup moins de types de `DEVICE` que de marques d'imprimantes. Pour obtenir une liste complète des périphériques d'impression supportés, utilisez la commande `gs -h` et consultez également un des fichiers suivants (cela dépendra de votre distribution) :

- `/usr/doc/ghostscript-?.??/devices.txt`
- `/usr/share/doc/ghostscript-?.??/Devices.htm`
- `/usr/share/doc/gs/devices.txt.gz`

Le terme `-sOutputFile=-` indique d'écrire dans stdout (comme cela est requis par le filtre). L'option `-sPAPERSIZE` peut être fixée à une des valeurs suivantes : `11x17`, `a3`, `a4`, `a5`, `b3`, `b4`, `b5`, `halfletter`, `ledger`, `legal`, `letter`, `note`. D'autres valeurs sont indiquées dans la page de `man`. Vous pouvez également utiliser `-g<largeur>x<hauteur>` pour attribuer la taille de page exacte en pixels. Par exemple, `-r600x600` fixe la résolution à 600 dpi (*dots per inch* ou points par pouce carré). L'option `-q` indique d'utiliser le mode silencieux, ce qui supprime les messages d'information qui autrement, corromperaient la sortie en PostScript. Le signe `-` signifie qu'il faut lire depuis stdin et non depuis le fichier.

La configuration de notre imprimante est à présent complète. Il faut maintenant lancer `lpd` et tester l'impression. Vous pouvez réaliser cette opération en ligne de commandes grâce au paquet `enscript`. `enscript` est un programme de conversion des fichiers-textes en pages PostScript joliment formatées. La page de man d'`enscript` décrit de très nombreuses d'options. Cependant, vous pouvez vous contenter d'essayer :

```
echo hello | enscript -p - | lpr
```

22.7 Contrôle d'accès.

Vous devriez être très prudent en exécutant la commande `lpd` sur des machines exposées à l'internet. `lpd` est l'objet de nombreuses alertes de sécurité [voir le chapitre 45] et ne devrait donc être utilisé que sur des réseaux LAN de confiance.

Pour empêcher que des machines distantes ne s'emparent de votre imprimante, `lpd` analyse d'abord le fichier `/etc/hosts.equiv`. Celui-ci contient une simple liste des machines autorisées à imprimer sur telle ou telle imprimante. Le fichier `/etc/hosts.equiv` de la machine de l'auteur contient l'information suivante :

```
192.168.3.8
192.168.3.9
192.168.3.10
192.168.3.11
```

Le fichier `/etc/hosts.lpd` présente les mêmes propriétés mais il ne donne pas à ces machines le contrôle des files d'impression. Notez qu'un autre service comme `sshd` ou `rshd` (ou `in.rshd`) analyse aussi le fichier `/etc/hosts.equiv` et considère toute machine listée comme `equiv`alente. Ceci signifie que le service leur fait pleine confiance et `rshd` ne requerra pas d'identification à la connexion entre machines authentifiées. Ce comportement est donc un défaut de sécurité grave. `sshd` requiert une identification et donc présente une fiabilité incomparablement plus grande.

`LPRng` sur RedHat 7.0 offre un contrôle d'accès différent. Ce paquet peut restreindre l'accès de diverses manières selon l'utilisateur distant et l'action entreprise (concernant l'utilisateur distant, il s'agira de déterminer qui est autorisé à manipuler des files). Le fichier `/etc/lpd.perms` contient la configuration. Le format du fichier est simple, bien que les capacités de LPRng soient plutôt complexes. Pour être bref, le fichier `hosts.equiv` devient `lpd.perms`.

```
ACCEPT SERVICE=* REMOTEIP=192.168.3.8
ACCEPT SERVICE=* REMOTEIP=192.168.3.9
ACCEPT SERVICE=* REMOTEIP=192.168.3.10
ACCEPT SERVICE=* REMOTEIP=192.168.3.11
DEFAULT REJECT
```

Les professionnels des grandes entreprises, ayant à faire à de nombreux utilisateurs auxquels on ne peut faire pleine confiance, devraient analyser de plus près le fichier `LPRng-HOWTO` de `/usr/share/doc/LPRng-n.n.nn`. Il explique comment limiter l'accès selon des modalités plus complexes que celles décrites ici.

22.8 Problèmes d'impression.

Voici un plan permettant de déterminer la nature des problèmes d'impression :

1. Vérifiez que l'imprimante est connectée et correctement alimentée. Toutes les imprimantes présentent la possibilité d'écrire une page de test. Lisez le manuel de votre imprimante.
2. Vérifiez le câblage.
3. Vérifiez les réglages CMOS de votre port parallèle (si vous utilisez une imprimante à port parallèle).
4. Vérifiez le câblage à nouveau.
5. Essayez `echo hello > /dev/lp0` pour tester que le port est en fonctionnement. L'imprimante devrait signaler que les données ont au moins été reçue. Le chapitre 43 explique comment installer le module associé au port parallèle.
6. Utilisez le programme `lpc` pour chercher le démon `lpd`. Essayez `help`, ensuite `status lp`, et ainsi de suite.
7. Vérifiez qu'il y a suffisamment d'espace dans les répertoires `/var` et `/tmp` pour tous les fichiers intermédiaires requis par le filtre d'imprimante. Un travail d'impression de grande taille nécessite plusieurs centaines de Mo. Il se peut que `lpd` ne retourne pas de message d'erreur lors d'un échec de fonctionnement du filtre d'impression : le travail d'impression disparaît sans retourner d'erreur. Si vous utilisez `lpr`, effectuez un rapport sur le site de votre distribution.
8. Concernant `lpr`, arrêtez le démon `lpd` et éliminez tous les fichiers susceptibles d'être appelés par `lpd` et qui se trouvent dans `/var/spool/lpd` ou tout sous-répertoire afférent [par le terme "fichier", il faut entendre tous les fichiers du programme en cours] (avec `LPRng`, il ne devrait pas être nécessaire de réaliser cette étape). Les fichiers non-désirés sont `.seq`, `lock`, `lpd.lock` et ceux se trouvant dans les fichiers du spoule qui ne peuvent être éliminés à l'aide de la commande `lprm` (ils sont reconnaissables à leurs noms très long qui comprennent le nom d'hôte et un nombre quelconque). Redémarrez `lpd`.
9. Pour les files d'impression distantes, vérifiez que vous pouvez effectuez des recherches directes et inverses sur chaque machine avec leurs noms d'hôte et les adresses IP correspondantes. Si ce n'est pas le cas, vous pourriez avoir des messages d'erreur du type `Hostname for your address (ipaddr) unknown` lorsque vous utilisez la commande `lpq`. Essayez les commandes `host <adresse_IP>` et également `host <nom_de_machine>`. Si aucune de ces commandes ne fonctionne, ajoutez de nouvelles entrées au fichier `/etc/hosts` de chaque machine (voir la section 28.3). Remarquez que la commande `host` peut échouer parce qu'elle ignore le fichier `/etc/hosts`. Le chapitre 41 explique comment configurer la recherche de nom.
10. Lancez le filtre d'impression manuellement pour s'assurer qu'il produit la sortie correcte. Par exemple : `echo hello | enscript -p - | /var/spool/lpd/lp/filter > /dev/lp0`.
11. Le démon `lpd` a quelques paquets "bizarres" – tenez en compte.

22.9 Programmes utiles.

22.9.1 printtool.

printtool est un programme d'installation en mode graphique qui vous permet d'installer lpd rapidement. Il produit un fichier printcap et un filtre, en vous évitant de vous plonger dans la configuration de lpd.

22.9.2 apsfilter.

apsfilter signifie *any to PostScript filter*. L'installation décrite plus haut requiert que tout soit converti en PostScript avant l'impression ; cependant un filtre pourrait auparavant utiliser la commande file pour déterminer le type de données et invoquer un programme destiné à convertir ces données en PostScript avant de l'envoyer via gs. Ceci permettrait aux fichiers texte, JPEG, GIF, DVI, et même aux fichiers HTML gzippés d'être imprimés directement étant donné que des convertisseurs Postscript ont été écrits pour chacun de ces types de fichiers. apsfilter est un de ces filtres appelés *filtres magiques* [ce terme provient du fait que la commande file utilise les nombres magiques, voir la section 5.9].

S'agit-il seulement d'une astuce ? Sans doute : la plupart du temps, les utilisateurs effectuent une prévisualisation de ce qu'ils veulent imprimer, ce qui nécessite une conversion en PostScript. Dans la majorité des cas, le filtre Postscript direct, décrit plus haut, fonctionne parfaitement bien pourvu que les utilisateurs utilisent enscript au lieu de lpr lors de l'impression de "texte plein".

22.9.3 mpage.

mpage est un utilitaire précieux lorsqu'il s'agit de sauver des arborescences. Il reformate les sorties PostScript de sorte que deux, quatre ou huit pages peuvent être regroupées sur une seule. La modification du filtre de sortie se fait ainsi :

```
#!/bin/bash
cat | mpage -4 | gs -sDEVICE=ljet4 -sOutputFile=- -sPAPERSIZE=a4 \
-r600x600 -q -
exit 0
```

22.9.4 psutils.

Le paquet psutils contient divers programmes de manipulation de PostScript en ligne de commandes. Ils sont d'un attrait majeur pour tous ceux qui exploitent les subtilités des filtres.

22.10 Imprimer sur autre chose qu'une imprimante.

Le fichier printcap contient la liste des imprimantes déclarées. Si nous substituons /dev/lp0 par /dev/null (voir l'exemple décrit à la section 22.5) et que

le filtre émet son résultat vers un autre périphérique, nous pouvons utiliser `lpd` pour rediriger tout travail d'impression sur n'importe quel service imaginable.

Voici ci-dessous, un script appelé `mon_filtre.sh` qui pourrait envoyer un travail d'impression *via* un partage d'impression SMB (Windows NT) en utilisant `smbclient` (voir le chapitre 40), vers un programme de prévisualisation ou vers un script qui transmettrait le travail par courriel :

```
lp1 :\
      :sd=/var/spool/lpd/lp1 :\
      :mx#0
      :sh :\
      :lp=/dev/null :\
      :if=/usr/local/bin/mon_filtre.sh
```

Au chapitre 34, nous verrons un exemple concret de redirection d'un travail d'impression sur un télécopieur.

Chapitre 23

Eléments de programmation C.

Le langage **C** a été inventé dans le but d'écrire un système d'exploitation qui pouvait être recompilé (donc porté) sur différentes plate-formes matérielles, c'est-à-dire différentes CPUs. Du fait que le système d'exploitation est écrit en **C**, ce langage est le premier auquel on pense pour écrire des applications susceptibles de communiquer efficacement avec le système d'exploitation.

De nombreuses personnes qui ne savent pas très bien programmer en **C** pense que celui-ci n'est qu'un langage parmi d'autres. Fixons les idées : le **C** est la base de toute la programmation dans le monde d'aujourd'hui. UNIX, Microsoft Windows, les suites "office", les navigateurs web et les pilotes sont tous écrits en **C**. Quatre-vingt dix pourcents du temps passé sur un ordinateur concernent des applications écrites en **C**. Environ 70% des logiciels libres sont écrits en **C**, et les 30% restant sont écrits dans des langages dont les compilateurs et interpréteurs sont écrits en **C**. [Le C++ est également très populaire. Cependant, il n'est pas aussi fondamental que le C, bien qu'il soit plus adapté dans de nombreuses situations].

Du reste, il n'y a pas de langage de substitution au **C**. Etant donné qu'il remplit ses fonctions sans défaillance, il n'a pas encore fallu lui trouver un remplaçant. *D'autres langages remplissent d'autres objectifs, mais le **C** est le langage qui convient le mieux aux fonctions auxquelles il est destiné.* Il est à peu près certain que les futurs systèmes d'exploitation seront écrits en **C**, et ce durant de nombreuses années, encore.

C'est pour cela que votre connaissance d'UNIX ne sera complète que si vous pouvez programmer en **C**. En revanche, le fait que vous puissiez écrire en **C** ne signifie pas que vous devrez l'utiliser pour modifier votre système d'exploitation proprement-dit. La bonne programmation en **C** est un art subtil qui échappe même à des personnes pratiquant la programmation depuis de nombreuses années. Il est essentiel de rejoindre un projet logiciel libre pour maîtriser de manière adéquate un bon style de programmation destiné au développement.

23.1 Les bases du C.

Nous commençons avec un programme très simple qui contient des éléments fondamentaux. Avant d'aller plus loin, vous voudrez peut-être revoir les fonctions `bash` de la section 8.7.

23.1.1 Le plus simple des programmes.

Voici un code assez élémentaire :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
   printf ("Hello World!\n");
   return 3;
}
```

Sauvez ce programme sous le nom `hello.c`. A présent, nous allons le compiler. [La compilation est le processus de traduction du code C en instructions en langage *assembler*. Ces dernières forment un code compréhensible par les CPUs qu'elles soient de type AMD, Pentium, 80?86, SPARC, Alpha ou RS6000. L'exécutable binaire qui résulte de la compilation est rapide car il est exécuté nativement par le processeur qui est la puce principale se trouvant sur votre carte-mère et qui va chercher `hello` dans la mémoire, octet par octet, de sorte à en exécuter chaque instruction. La vitesse d'exécution s'exprime en MIPS (*millions instructions per second*). Le terme mégahertz ou gigahertz caractérisant le matériel est *très* approximativement le nombre de MIPS. Les langages interprétés (comme les scripts de shell) sont beaucoup plus lents car le code est écrit dans un langage qui n'est pas directement interprétable par la CPU (code non-natif). `/bin/bash` est lui-même écrit en C. L'exécution de scripts en bash est plus lente de plusieurs ordres de grandeur que celle des langages compilés. Les scripts de shell ne doivent pas être compilés].

Exécutez la commande :

```
gcc -Wall -o hello hello.c
```

L'option `-o hello` indique à `gcc` [GNU **C** Compiler] de produire un fichier binaire nommé `hello` au lieu du fichier binaire par défaut `a.out` [ce nom a été attribué pour des raisons historiques]. L'option `-Wall` indique de faire un rapport de toutes les alertes (`all` *Warnings*) durant la compilation. Ceci n'est pas obligatoire mais cela s'avère très utile pour corriger d'éventuelles erreurs dans les programmes. Les options de compilation sont discutées au paragraphe 25.1.

Exécutez le programme :

```
./hello
```

Précédemment, nous nous sommes familiarisés avec les fonctions `bash`. En **C**, tout le code est contenu dans une ou plusieurs fonctions. La première qui est

appelée par le système d'exploitation est la fonction `main`. Tapez `echo $?` pour voir le code de retour du programme. Vous verrez qu'il s'agit de `3`, la valeur `return` de la fonction `main`.

D'autres points sont à noter : par exemple, le signe " entourant la chaîne qui doit être affichée à l'écran. Les guillemets entourent toujours une chaîne littérale. A l'intérieur d'une chaîne littérale, la séquence `\n` indique un retour de chariot. `ascii`(7) décrit en détail d'autres séquences. Vous noterez aussi la prolifération de `;` dans un programme en **C**. Chaque instruction doit se conclure par un `;` contrairement aux scripts de shell où le signe `;` est optionnel.

A présent, essayez :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    printf ("number %d, number %d\n", 1+2,10) ;
    exit (3) ;
}
```

`printf` peut être considéré comme la commande envoyant un message de sortie au terminal. Elle est connue sous le nom de *fonction de la bibliothèque standard du C* (*standard C library function*). En d'autres termes, toute implémentation[1] **C** devrait toujours présenter la fonction `printf` et elle devrait toujours se comporter de la même façon.

L'expression `%d` représente un nombre décimal avec un point à l'affichage. Le premier `%d` sera substitué par le premier argument de la fonction `printf` après la chaîne littérale, en l'occurence par le résultat de l'opération `1+2`. Le second `%d` sera remplacé par le second argument (ici, `10`). L'expression `%d` est un *descripteur de format*. Elle *convertit* un nombre entier en nombre décimal. Voir `printf`(3) pour plus de détails.

23.1.2 Variables et types.

Avec le `bash`, vous pouvez utiliser une variable n'importe où, à n'importe quel moment. La variable sera vide si aucune valeur ne lui est explicitement attribuée. En **C**, vous devez déclarer les variables explicitement et au préalable de tout usage dans le code. La déclaration de variables est effectuée de la manière suivante :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int x ;
    int y ;
```

[1] NdT : le mot est désormais consacré, il signifie "mise en oeuvre".

```
    x = 10 ;
    y = 2 ;
    printf (“number %d, number %d\n”, 1 + y, x) ;
    exit 3 ;
}
```

L'expression `int x` constitue une déclaration de variable. Elle indique au programme de réserver un emplacement en mémoire pour une variable de type entier (integer) à laquelle on se référera comme `x`. `int` est le type de la variable. L'expression `x = 10` attribue une valeur de 10 à la variable `x`. Il existe des types pour toutes les sortes de variables envisageables et des descripteurs de format pour en permettre l'affichage :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    char a ;
    short b ;
    int c ;
    long d ;
    float e ;
    double f ;
    long double g ;
    a = ’A’ ;
    b = 10 ;
    c = 10000000 ;
    d = 10000000 ;
    e = 3.14159 ;
    f = 10e300 ;
    g = 10e300 ;
    printf (“%c, %hd, %d, %ld, %f, %f, %Lf\n”, a, b, c, d, e, f, g) ;
    exit (3) ;
}
```

Vous remarquez que `%f` est utilisé aussi bien pour le type `float` que pour le type `double`. En fait, un type `float` est toujours converti en `double` avant d'être utilisé. Essayez aussi de remplacer `%f` avec `%e` pour afficher un nombre en notation exponentielle.

23.1.3 Fonctions.

Les fonctions sont implémentées de la manière suivante :

```
#include <stdlib.h>
#include <stdio.h>

void multiply_and_print (int x, int y)
{
```

```
    printf ("%d, %d = %d\n", x, y, x * y) ;
}


int main (int argc, char *argv[])
{
    multiply_and_print (30, 5) ;
    multiply_and_print (12, 3) ;

    exit (3) ;
}
```

Ici, nous avons une fonction "non-main" *appelée* par la fonction `main`. Elle est d'abord déclarée dans le code ainsi :

```
void multiply_and_print (int x, int y)
```

Cette déclaration indique la valeur retournée par la fonction (`void` signifie qu'il n'y a pas de valeur), le nom de la fonction (ici, `multiply_and_print`) et la liste des arguments qui lui sont passés. Les nombres passés à cette fonction sont décrits par leur propre nom, `x` et `y`. Ils sont convertis dans le type associé à `x` et à `y` avant d'être passés à la fonction `multiply_and_print`. Dans ce cas, les types sont `int` et `int`. Le code **C** qui décrit la fonction se trouve entre les accolades `{` et `}`. En d'autres termes, la fonction `multiply_and_print` est équivalente à ceci :

```
void multiply_and_print ()
{
    int x ;
    int y ;
    x = <premier_nombre_passe>
    y = <second_nombre_passe>
    printf ("%d * %d = %d\n", x, y, x * y) ;
}
```

23.1.4 Les déclarations `for`, `while`, `if` et `switch`.

Comme dans les scripts de shell, il existe des déclarations `for`, `while`, `if` et `switch` :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int x ;
    x = 10 ;

    if (x == 10) {
       printf ("x vaut 10\n") ;
```

```
        x++ ;
    } else if (x == 20) {
        printf ("x vaut 20\n") ;
    } else {
        printf ("x ne vaut ni 10 ni 20") ;
    }

    if (x > 10) {
        printf ("x vaut plus que 10\n") ;
    }

    while (x > 10) {
        printf ("x vaut %d\n", x) ;
        x = x - 1 ;
    }

    for (x = 0 ; x < 10 ; x++) {
        printf ("x vaut %d\n", x) ;
    }

switch (x) {
        case 9 :
            printf ("x vaut 9\n") ;
            break ;
        case 10 :
            printf ("x vaut 10\n") ;
            break ;
        case 11 :
            printf ("x vaut 11\n") ;
            break ;
        default :
            printf ("x ?") ;
            break ;
        }
return 0 ;
}
```

Il est facile de voir le format que ces instructions prennent, quoiqu'elles soient très différentes de celles associées aux scripts de shell. Le code **C** fonctionne par *blocs* de déclarations décrites entre accolades. Les scripts de shell, eux, pratiquent avec des couples `do` et `done`.

Remarquez qu'avec la plupart des langages de programmations, lorsque nous voulons ajouter 1 à une variable, nous écrivons une instruction du type : `x=x+1`. En **C**, l'abréviation `x++` est utilisable et signifie qu'il faut incrémenter la variable `x` d'une unité.

La boucle `for` contient trois déclarations entre `(...)` : la première pour initialiser `x`, un test et une incrémentation de la variable `x` à exécuter tant que la condition reste valide.

`switch` est similaire à l'instruction `case` des scripts de shell, et son argument est la portion de code qui se trouve dans les parenthèses `()` c'est-à-dire `x`. La valeur de cet argument décide de la ligne à laquelle le flux de programmation se poursuit. Dans notre exemple, le flux se rendra à l'instruction `printf ("x vaut`

10\n") ; puisque x vaut 10 à la sortie de la boucle for. La marque break signifie qu'une fois le "cas" rencontré et son code exécuté, le flux de programmation passe à la ligne se trouvant après l'instruction break associée à default.

Enfin, notons qu'en **C**, l'opérateur d'égalité est == *et non* =.

23.1.5 Chaînes, vecteurs et allocation mémoire.

Il vous est loisible de définir une liste de 10 nombres entiers par exemple, sous forme d'un tableau monodimensionnel (ou vecteur) :

```
int y[10] ;
```

En voici un usage simple :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int x ;
    int y[10] ;
    for (x = 0 ; x < 10 ; x++) {
        y[x] = x * 2 ;
    }
    for (x =0 ; x < 10 ; x++){
        printf ("item %d is %d\n", x, y[x])
    }
    return 0 ;
}
```

Si un tableau est de type charactère, on le désigne sous le terme de chaîne (*string*) :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int x ;
    char y[11] ;

    for (x = 0 ; x < 10 ; x++) {
      y[x] = 65 + x * 2 ;
    }
    for (x =0 ; x < 10 ; x++) {
      printf ("item %d is %d\n", x, y[x]) ;
    }
    y[10] = 0 ;
    printf ("string is s%\n", y) ;
    return 0 ;
```

```
}
```

Observez qu'il est nécessaire de terminer toute chaîne par un "zéro". Le code `y[10] = 0` définit le $11^{\text{ème}}$ caractère du tableau comme étant zéro. La conséquence est que la longueur d'une chaîne contient toujours une unité de plus que le nombre de caractères qui lui donne un sens linguistique. Remarquez aussi que le premier caractère est `y[0]` et non `y[1]`, une particularité retrouvée dans de nombreux langages de programmation.

Dans l'exemple qui précède, la ligne `char y[11]` réserve en mémoire 11 octets pour la chaîne. Que se passerait-il si nous voulions 100.000 octets ? Le **C** nous permet de demander de la mémoire au noyau. Cette méthode est appelée l'*allocation mémoire* (ou *allocation memory*). Tout programme un peu complexe nécessite de l'allocation mémoire pour lui-même et on ne peut obtenir de blocs mémoires de grande taille autrement qu'en pratiquant ainsi :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int x ;
    char *y ;
    y = malloc (11) ;
    printf ("%ld\n", y) ;
    for (x = 0 ; x < 10 ; x++) {
        y[x] = 65 + x * 2 ;
    }
    y[10] = 0 ;
    printf ("string is s%\n", y) ;
    free (y) ;
    return 0 ;
}
```

La déclaration `*y` signifie qu'on déclare une variable (un nombre) appelé `y` qui pointe vers une adresse de la mémoire RAM. Le symbole `*` veut dire *pointeur*. Disons que vous avez une machine avec 256 Mo de RAM + swap, `y` a potentiellement une taille de cette valeur-là. (La valeur numérique d'`y` est affichée avec `printf ("%ld\n", y) ;`, mais cela n'a pas d'intérêt immédiat pour le programmeur).

Lorsque vous en avez fini avec l'utilisation de la mémoire, vous devez rendre l'espace qui vous a été accordé, au noyau en utilisant `free`. Les programmes qui n'ont pas restitué toute la mémoire qui leur a été allouée sont dits à *fuite de mémoire*.

L'allocation de mémoire requiert très souvent de se livrer à un petit calcul préalable de manière à déterminer la quantité de mémoire requise. Dans le cas que nous venons de voir, nous avons alloué l'espace associé à 11 caractères (`char`s). Etant donné que chaque caractère compte pour un octet, cela ne pose pas de problème. Que se passerait-il si nous avions à faire à 11 entiers (`int`). Un `int` sur un PC vaut 32 bits soit 4 octets (4 adresses consécutives de la mémoire).

Pour déterminer la taille d'un type, nous pouvons utiliser le mot-clé `sizeof` :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int a ;
    int b ;
    int c ;
    int d ;
    int e ;
    int f ;
    int g ;
    a = sizeof (char) ;
    b = sizeof (short) ;
    c = sizeof (int) ;
    d = sizeof (long) ;
    e = sizeof (float) ;
    f = sizeof (double) ;
    g = sizeof (long double) ;
    printf ("%d, %d, %d, %d, %d, %d, %d\n", a, b, c, d, e, f, g) ;
    return 0 ;
}
```

Grâce à ce programme, vous pouvez déterminer le nombre d'octets associés à chaque type. A présent, vous pouvez aisément allouer de la mémoire pour des tableaux constitués d'autres types que des `char`.

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    int x ;
    int *y ;
    y = malloc (10 * sizeof (int)) ;
    printf ("%ld\n", y) ;

    for (x = 0 ; x < 10 ; x=++) {
        y[x] = 65 + x * 2 ;
    }
    for (x = 0 ; x < 10 ; x=++) {
        printf ("%dl\n", y[x]) ;
    }
    free (y) ;
    return 0 ;
}
```

Sur de nombreuses machines, un `int` compte pour 4 octets (32 bits), mais vous

ne devriez jamais postuler cela. *Utilisez toujours* `sizeof` *avant de pratiquer une allocation mémoire.*

23.1.6 Opérations sur les chaînes.

Les programmes en **C** manipulent des chaînes probablement plus que tout autre langage de programmation. Voici un code qui fragmente une phrase en mots :

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[])
{
    int length_of_word;
    int i;
    int length_of_sentence;
    char p[256];
    char *q;

    strcpy (p, "hello there, my name is fred.");
    length_of_sentence = strlen (p);
    length_of_word = 0;

    for (i = 0; i <= length_of_sentence; i++) {
        if (p[i] == ' ' || i == length_of_sentence) {
            q = malloc (length_of_word + 1);
            if (q == 0) {
                perror ("malloc failed");
                abort ();
            }

            strncpy (q, p + i - length_of_word, length_of_word) = 0;
            q[length_of_word] = 0;
            printf ("word : %s\n", q);
            free (q);
            length_of_word = 0;
        } else {
            length_of_word = length_of_word + 1;
        }
    }
    return 0;
}
```

Nous introduisons trois nouvelles fonctions de la *bibliothèque standard* **C** (`strcpy`, `strlen` et `strncpy`).

`strcpy` veut dire `str`*`ing`*`co`*`p`*`y`. Cette fonction copie les octets d'un endroit à un autre de manière séquentielle, jusqu'à atteindre l'octet zéro qui signale la fin d'une chaîne. La ligne 13 de ce programme copie le texte dans le tableau de

caractères p (character p array) qui est la cible de la fonction strcpy.

strlen signifie *string length*. Cette fonction permet de déterminer la longueur d'une chaîne de caractères, c'est-à-dire le nombre de caractères, y compris les espaces, qui la constituent et ce, jusqu'au caractère zéro de terminaison.

Il est nécessaire d'effectuer une boucle for sur la longueur de la phrase. La variable i sert de compteur indiquant la position courante dans la phrase. La ligne 20 qui commence par if, indique que si nous trouvons un caractère ASCII 32 (c'est-à-dire un espace, noté ' '), nous savons que nous avons atteint la fin d'un mot. Nous savons aussi que la fin de la phrase vaut une fin de mot. Le symbole || désigne un OU logique. Une fois la fin d'un mot atteinte, nous pouvons allouer de la mémoire pour le mot détecté et ensuite copier ce dernier dans l'emplacement mémoire réservé.[2]

La fonction strncpy permet d'effectuer cette dernière opération. Comme pour strcpy, le premier argument est la cible (q) et le second argument est l'objet à copier. Il reste à déterminer la position de début du *dernier* mot : p + i - length_of_word.[3]

Finalement, à la ligne 27, nous terminons la chaîne par zéro. On peut écrire q, et libérez la mémoire (free) pour recommencer sur le mot suivant.

Pour une liste complète des opérations associées aux chaînes, consultez string(3).

23.1.7 Opérations sur les fichiers.

Dans de nombreux langages de programmation, les opérations relatives aux fichiers impliquent trois étapes : *ouvrir*, *lire* (et/ou *écrire*), *fermer*. Il est possible d'utiliser la commande fopen pour indiquer au système d'exploitation que vous êtes prêt à commencer un travail sur un fichier. Le code suivant ouvre un fichier et envoye son contenu à l'écran :

---

[2] NdT : il faut toujours se souvenir d'ajouter une unité à la longueur d'une chaîne (ici d'un mot) pour le zéro de clôture ; d'où l'expression malloc (length_of_word + 1).

[3] NdT : nous pouvons pratiquer pas-à-pas sur le cas du mot is. Lorsque la lettre "i" du mot "is" est rencontrée, la valeur du compteur i est 21 puisque le compte a été commencé par i=0. Les deux "if" qui suivent l'entrée dans la boucle "for" sont passés car, concernant le premier, p[21] vaut 105, selon la table ASCII. Au tour précédent, lorsque la mémoire a été restituée, lentgh_of_word a été ré-initialisé à zéro (ligne 30). A la ligne 32, lentgh_of_word est incrémenté d'une unité. La boucle for continue avec i = 22. La lettre s est rencontrée et, passant les if on arrive à ce que lentgh_of_word devienne = 2. Cependant, au tour suivant, lorsque i = 23, p[23] est un caractère espace. On réserve alors de la mémoire sous forme d'un tableau q (il est à noter que tous les tableaux sont en fait des pointeurs puisque leur nom pointe vers la première adresse hexadécimale du premier octet du tableau). Pour terminer de manière correcte la chaîne de caractère is , on lui adjoint un caractère "nul", ce qui est fait en deux temps. A la ligne 21, la mémoire demandée via malloc est de length_of_word + 1 (soit 2+1). A la ligne 27, le dernier élément de q (c'est-à-dire q[lenght_of_word] ou encore q[3]) recevra un zéro de clôture de chaîne. Entre-temps, à la ligne 26, strncpy transfère is dans q. Il s'agit de réaliser une opération arithmétique sur le pointeur p. Celui-ci pointe toujours sur la première adresse hexadécimale du premier octet du tableau p. Lorsqu'on incrémente p d'une unité, le pointe se déplace (comme un curseur) d'une unité car il s'agit d'un type char. Si nous avions eu à faire à un tableau d'entiers, le pointeur eût effectué un déplacement de 4 octets [voir le programme relatif à sizeof dans la section 23.1.5]. Donc, dans la mémoire, le pointeur p s'est déplacé de l'adresse désignant la lettre h à +23 positions (p + i). Là, il pointe vers le caractère "espace" suivant le mot "is". Ensuite, on le fait se déplacer en arrière de length_of_word (soit 3 octets). Et, dans q , on sauve le mot compris entre ces positions, c'est-à-dire "is". Après affichage du mot contenu dans q, la mémoire occupée par q est libérée.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[])
{
    int c ;
    FILE *f ;

    f = fopen (“mytest.c”, “r”) ;
    if (f == 0) {
        perror (“fopen”) ;
        return 1 ;
    }
    for ( ; ;) {
        c = fgetc (f) ;
        if (c == -1)
            break ;
        printf (“%c”, c) ;
    }
    fclose (f)
    return 0 ;
}
```

`FILE *` constitue un nouveau type pour nous. Il s'agit d'une variable associée à des opérations, qui doit être initialisée avec `fopen` avant d'être utilisée. La fonction `fopen` prend deux arguments : le premier est le nom de fichier, le second est une chaîne expliquant le *mode* d'ouverture du fichier –en l'occurrence, `r` signifie *reading* (“en lecture depuis le début du fichier”). Une autre option courante est `w` pour *writing* (“en écriture”). D'autres options sont décrites dans `fopen`(3).

Si la valeur retournée par `fopen` est zéro, on dit que `fopen` a échoué. La fonction `perror` affiche alors un message littéral comme, par exemple, `No such file or directory`. Il est essentiel de vérifier la valeur de retour de tous les appels de bibliothèques selon cette méthode. Ces vérifications peuvent constituer jusqu'à environ un tiers de vos programmes en **C**.

La commande `fgetc` *extrait* un caractère du fichier. Elle récupère les octets consécutifs contenus dans le fichier jusqu'à ce qu'elle atteigne la fin de ce dernier. Elle renvoye alors le nombre `-1`. La déclaration `break` impose de terminer immédiatement la boucle `for`, pour reprendre le flot d'exécution à la ligne 21. Les déclarations `break` peuvent éventuellement apparaître à l'intérieur des boucles `while`.

Vous aurez noté que la déclaration `for` est vide ; ce qui implique qu'il n'y a pas de conditions de boucle et, par conséquent, que celle-ci est infinie. Ceci est tout-à-fait admis en **C**. En l'occurrence, on n'en sort que par le `break`.

D'autres fonctions associées aux fichiers sont `fread`, `fwrite`, `fputc`, `fprintf` et `fseek`. Consultez `fwrite`(3), `fputc`(3), `fprintf`(3) et `fseek`(3).

23.1.8 Lire des arguments en ligne de commande dans les programmes en C.

Jusqu'à présent, nous n'avons pas vu en quoi l'expression `(int argc, char *argv[])` était utile. Elle désigne les arguments en ligne de commandes passés au programme par le shell. `argc` est le nombre total d'arguments en ligne de commandes tandis qu'`argv` est une chaîne (c'est-à-dire un tableau) pour chaque argument. On peut les afficher à l'aide du code suivant :

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[])
{
    int i ;
    for (i = 0 ; i < argc ; i++) {
            printf ("argument %d is %s\n", i, argv[i]) ;
    }
    return 0 ;
}
```

23.1.9 Un exemple plus complexe.

Dans l'exemple qui suit, nous combinons les notions vues précédemment en un programme (appelons-le `wordsplit.c`) qui lit de nombreux fichiers et en extrait les mots. Nous y rencontrons la notation `!=` qui teste la *non-égalité* et qui est l'inverse de `==` (*signe d'égalité*). Nous y voyons aussi (1) `realloc` qui redéfinit la taille d'un bloc mémoire préalablement utilisé de sorte que tous les octets de l'ancien bloc sont conservés et (2) `\n` qui met à la ligne et `\t` qui effectue une tabulation (il s'agit respectivement des caractères ASCII 10 et 9 ; voir `ascii`(7)).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void word_dump (char *filename)
{
   int length_of_word ;
   int amount_allocated ;
   char *q ;
   FILE *f ;
   int c ;

   c = 0 ;

   f = fopen (filename, "r") ;
   if (f == 0) {
```

```
        exit (1);
        perror ("fopen failed");
    }

    length_of_word = 0

    amount_allocated = 256;
    q = malloc (amount_allocated);
    if (q == 0) {
        perror ("malloc failed");
        abort ();
    }

     while (c != -1) {
         if (length_of_word >= amount_allocated) {
            amount_allocated = amount_allocated * 2;
            q = realloc (q, amount_allocated);
            if (q == 0) {
                perror ("realloc failed");
                abort ();
            }
          }

     c = fgetc (f);
     q[length_of_word] = c;

     if (c == -1 || c == ' ' || c == '\n' || c == '\t') {
         if (length_of_word > 0) {
             q[length_of_word] = 0;
             printf ("%s\n", q);
          }
          amount_allocated = 256;
          q = realloc (q, amount_allocated);
          if (q == 0) {
              perror ("realloc failed");
              abort ();
          }         length_of_word = 0;
       } else {
          length_of_word = length_of_word + 1;
       }
     }
     fclose (f);
}
int main(int argc, char *argv[])
{
     int i;
     if (argc < 2) {
        printf ("Usage :\n\twordsplit <filename> ...\n");
        exit (1);
     }
```

```
    for (i = 1; i < argc; i++) {
        word_dump (argv[i]);
    }

    return 0;
}
```

Ce programme est plus compliqué qu'il n'y paraît à première vue. Lire un fichier pour lequel vous êtes sûr qu'aucun mot ne dépasse 30 caractères est élémentaire. Cependant, que se passe-t-il si des "mots" excèdent 100.000 caractères ? Les programmes GNU sont supposés se comporter correctement dans toutes les circonstances.

Pour faire face aux circonstances exceptionnelles autant que normales, nous allons d'abord supposer qu'un mot ne dépasse pas 256 caractères. S'il apparaît que cette hypothèse ne convient plus, nous réallouons (`realloc`*ate*) un espace mémoire double (lignes 32 et 33). Lorsque nous commençons avec un nouveau mot, nous pouvons libérer cette mémoire à nouveau de sorte à désallouer (`realloc`*ate back* ; lignes 48 et 49). De cette manière, nous utilisons toujours un minimum d'espace mémoire à tout instant.

Nous avons donc créer un programme qui peut travailler efficacement avec des fichiers de centaines de Go aussi bien qu'avec des fichers de l'ordre de la centaine d'octets. Ceci constitue *une partie de l'art de la programmation* **C**.

Les programmeurs expérimentés en **C** remarqueront de suite que ce programme n'est pas "minimalisé". En fait, c'est un excellent programme parce qu' :

- il est aisé à comprendre,
- il utilise un algorithme efficace (bien que non-optimisé),
- il n'y a pas de limites imposées à son usage qui, autrement, provoqueraient des comportements inattendus dans des conditions d'usage intensif,
- il n'utilise pas de fonctions **C** ou de notations non-standards qui empêcheraient sa compilation sur d'autres plate-formes ; il est donc *portable*.

La lisibilité d'un programme en ***C*** *est votre objectif majeur : il est impératif que ce que vous avez écrit soit évident à lire par une autre personne lisant le code.*

23.1.10 `#include` et les prototypes.

Au début de chaque programme, il y a des déclarations `#include`. Elles informent le compilateur de lire dans d'autres programmes en **C**. Ceci dit, le code "brut" ne présente pas de protection totale envers les erreurs : par exemple, la fonction `strcpy` pourrait tout aussi bien être utilisée avec un, deux, trois ou quatre arguments, ce qui n'empêcherait pas le programme d'être compilé. Cependant, l'exécution se passerait mal du côté de la mémoire s'il n'y avait pas deux arguments et, le programme planterait. Ces autres programmes **C** avec un suffixe `.h` sont des en-têtes ou *headers*. Ils contiennent des modèles (ou *templates*) sur base desquels les fonctions peuvent être appelées. Chaque fonction que vous utilisez est contenue dans un fichier "modèle". Ces modèles sont appelés des *prototypes de fonctions* (ou *function prototypes*). [Le C++ possède aussi des "templates", mais le terme utilisé en C++ désigne tout autre chose que ce dont nous discutons ici].

Le prototype d'une fonction est écrit comme la fonction elle-même, le code en moins. Un prototype de la fonction `word_dump` est :

```
void word_dump (char *filename) ;
```

La terminaison `;` est essentielle et permet de distinguer un prototype de fonction de la fonction elle-même.

Après qu'un prototype de fonction ait été défini, toute tentative d'utiliser cette fonction hors de son contexte d'exécution engendrera une erreur sans appel de la part de `gcc`. Travailler hors contexte d'exécution signifie passer trop ou trop peu d'arguments à cette fonction, ou des arguments non valides.

Vous noterez que `#include <string.h>` apparaît lorsque nous envisageons d'utiliser des opérations sur les chaînes (*strings*). La recompilation de ces programmes sans la ligne `#include <string.h>` engendrerait l'émission d'un avertissement :

```
mytest.c :21 : warning : implicit declaration of function 'strncpy'
```

Les prototypes de fonction donnent une définition claire de la manière dont chacune des fonctions est utilisée. En effet, on y trouve le nombre d'arguments et le type de ceux-ci mis en évidence.

23.1.11 Les commentaires.

Un commentaire en **C** est cité ainsi : `/* <ligne de commentaires> */` et il est possible de le faire tenir sur plusieurs lignes. Tout ce qui se trouve entre les signes `/*` et `*/` est ignoré par le compilateur. Chaque fonction devrait comprendre un commentaire ainsi que tout code non trivial. Il existe une bonne maxime qui dit que tout code mal écrit requiert du commentaire. Ne commentez jamais ce qui est évident. Expliquez *pourquoi* vous développez tel ou tel point mais pas *ce que* vous faites. Il n'est pas judicieux de réaliser de jolis graphiques entre les fonctions. Ainsi écrivez plutôt ceci :

```
/* returns -1 on eror, takes a positive integer */
int sqr (int x)
{
    < ... >
```

que cela :

```
/****************************----SQR----*******************************
*               x = argument to make the square of                    *
*               return value =                                        *
*                              -1 on error                            *
*                              square of x (on success)               *
*********************************************************************
int sqr (int x)
{
    < ... >
```

ce qui est très désagréable. En **C++**, l'addition de commentaires derrière `//` est

permise. Cette pratique autorisée par gcc ne peut pas être recommandée sauf si vous écrivez en **C++**. De plus, les programmeurs introduisent souvent des lignes de commentaires entre #if 0 et # endif ce qui produit le même résultat que décrit plus haut mais autorise à placer des commentaires à l'intérieur du commentaire principal (voir la section 23.1.12). Ainsi, par exemple, dans le code suivant :

```
  int x ;
  x = 10 ;
#if 0
     printf ("debug : x is %d\n", x) ;     /*print debug information */
#endif
  y = x + 10 ;
  < ... >
```

23.1.12 #define et #id – les macros en C.

Toute ligne qui commence par un # n'est pas vraiment du **C**, mais bel et bien une *directive de préprocesseur*. Un programme en **C** est, avant toute chose, traité par un préprocesseur qui élimine tout ce qui n'est pas les instructions proprement dites, comme par exemple les commentaires, les déclarations # include, et en général, les lignes débutant par #. Vous pouvez rendre vos programmes en **C** plus lisibles en définissant des macros au lieu des valeurs littérales. Par exemple, dans notre programme,

```
#define START_OF_BUFFER_SIZE 256
```

ce code définit (#defines) le texte START_OF_BUFFER_SIZE comme équivalent au texte 256. Par la suite, partout dans le code C où nous avons à faire à START_OF_BUFFER_SIZE, le compilateur y verra le texte 256. L'usage de l'expression START_OF_BUFFER_SIZE rend le code *plus clair* car, en cas de modifications, un seul changement à la ligne #define START_OF_BUFFER_SIZE 256 doit être fait. START_OF_BUFFER_SIZE est également plus facile à lire, à interpréter pour le programmeur.

Chaque fois que vous avez une *constante littérale* comme 256, vous devriez la remplacer par une macro définie au début du programme.

Vous pouvez aussi vérifier l'existence de macros avec les directives #ifdef et #ifndef. En soi, les directives # constituent un langage de programmation :

```
/* Set START_BUFFER_SIZE to fine-tune performance before compiling : */
#define START_BUFFER_SIZE 256
/* #define START_BUFFER_SIZE 128 */
/* #define START_BUFFER_SIZE 1024 */
/* #define START_BUFFER_SIZE 16384 */

#ifndef START_BUFFER_SIZE
#error This code did not define START_BUFFER_SIZE. Please edit
#endif
```

```
#if START_BUFFER_SIZE <= 0
#error Wooow START_BUFFER_SIZE must be greater than zero
#endif

#if START_BUFFER_SIZE < 16
#warning START_BUFFER_SIZE too small, progamme may be inefficient
#elif START_BUFFER_SIZE > 65536
#warning START_BUFFER_SIZE too large, programme may be inefficient
#else
/* START_BUFFER_SIZE is OK, do not report */
#endif

void word_dump (char *filename)
{
   < ... >
   amount_allocated = START_BUFFER_SIZE;
   q = malloc (amount_allocated);
   < ... >
```

23.2 Débogage avec gdb et strace.

Les erreurs de programmation (ou bogues ; en anglais, *bugs*) peuvent être trouvées en inspectant l'exécution du programme. Certains développeurs n'aiment pas pratiquer ainsi. Néanmoins, il est instructif d'apprendre le **C** en observant réellement le travail effectué par un programme.

23.2.1 gdb.

Le débogueur de GNU, `gdb`, remplace le débogueur `db` d'UNIX.[4] Le débogage d'un programme consiste à en réaliser l'exécution ligne par ligne, en vue de trouver les erreurs éventuelles. Utilisez la commande `gcc -Wall -g -O0 -o wordspilt wordsplit.c` pour recompiler le programme de la sous-section 23.1.9. L'option `-g` signifie "support de débogage" dans l'exécutable résultant et l'option `-O0` désactive toute optimisation du compilateur (qui, sinon, parfois provoque un comportement déroutant). Pour réaliser l'exemple qui suit, créez deux fichiers d'essai `readme.txt` et `readme2.txt` contenant du texte. A présent, vous pouvez lancer `gdb -q wordsplit`. L'invite de `gdb` apparaît pour indiquer le début d'une *session de débogage* :

```
(gdb)
```

Derrière cette invite, il est possible de passer de nombreuses commandes sous forme d'une seule lettre, de manière à contrôler l'exécution. La première de celle-ci est *run* qui exécute le programme comme s'il avait été démarré depuis un shell de manière classique :

[4] N.d.T : `ddd` est un autre programme très utile et graphique.

```
(gdb) r
Starting program : /home/src/wordsplit/wordsplit.c
Usage :
       wordsplit <filename> ...

Programme exited with code 01
```

Naturellement, nous pouvons passer des arguments en ligne de commande. Pour cela, on utilise la commande spéciale `set args` :

```
(gdb) set args readme1.txt readme2.txt
```

La commande *break* est utilisée comme `b [[<fichier> :]<ligne>[<fonction->]]`, et place un point d'arrêt (*breakpoint*) à un numéro de ligne ou au début d'une fonction :

```
(gdb) b main
Breakpoint 1 at 0x8048796 : file wordsplit.c, line 67.
```

Un point d'arrêt interrompt l'exécution du programme. Dans le cas présent, le programme peut s'arrêter quand il entre dans la fonction `main` (c'est-à-dire tout au début). A présent, vous pouvez relancer le programme :

```
(gdb) r
Starting program : /home/src/wordsplit/wordsplit readme.txt readme2.txt

Breakpoint 1, main (argc=3, argv=0xbffff804) at wordsplit.c :67
67          if (argc < 2) {
(gdb)
```

Linux

Comme indiqué, le programme s'arrête au début de `main` à la ligne 67.

Si vous vous intéressez à ce que contient une variable, vous pouvez faire usage de la commande *print* :

```
(gdb) p argc
$1 = 3
(gdb) p argv[1]
$2 = 0xbffff988 "readme.txt"
```

ce qui nous retourne la valeur d'`argc` et `argv[1]`. La commande *list* affiche les lignes voisines de la ligne courante :

```
(gdb) l
63   int main (int argc, char *argv[])
64   {
65       int i ;
66
67       if (argc < 2) {
68           printf ("Usage :\n\twordsplit <filename> ...\n") ;
69           exit (1) ;
70       }
```

La commande `list` peut aussi prendre un fichier en option et un numéro de ligne (ou même un nom de fonction) :

```
(gdb) l wordsplit.c :1
1        #include <stdlib.h>
2        #include <stdio.h>
3        #include <string.h>
4
5        void word_dump (char *filename)
6        {
7            int length_of_word ;
8            int amount_allocated ;
```

Ensuite, nous pouvons placer un point d'arrêt à un ligne donnée et utiliser la commande `continue` pour poursuivre l'exécution du programme :

```
(gdb) b wordsplit.c :48
Breakpoint 2 at 0x804873e : file wordsplit.c, line 48.
(gdb) c
Continuing.
Zaphod

Breakpoint 2, word_dump (filename=0xbffff988 "readme.txt") at wordsplit.c :48
48                  amount_allocated = 256 ;
```

L'exécution s'arrête à la ligne 48. A ce stade, il est utile de lancer un `backtrace`. Ceci affiche la *pile* des fonctions qui ont été appelées pour atteindre la ligne courante, ce qui vous permet de tracer l'historique de l'exécution.

```
(gdb) bt
#0   word_dump (filename=0xbffff988 "readme.txt") at wordsplit.c : 48
#1   0x80487e0 in main (argc=3, argv=0xbffff814) at wordsplit.c :73
#2   0x4003db65 in __libc_start_main (main=0x8048790 <main>, argc=3,
ubp_av=0xbffff814, init=0x08048420 <_init>,
     fini=0x804883c <_fini>, rtld_fini=0x4000df24 <dl_fini>,
stack_end=0xbffff80c) at ../sysdeps/generic/libc-start.c :111
```

La commande `clear` supprime les points d'arrêts de la ligne courante :

```
(gdb) clear
Deleted breakpoint 2
```

Les commandes les plus importantes pour le débogage sont n*ext* et s*tep*. La commande n exécute une seule ligne de code **C** à la fois :

```
(gdb) n
49          q = realloc (q, amount_allocated);
(gdb) n
50          if (q == 0) {
(gdb) n
50          length_of_word = 0;
```

Cette activité est appelée le s*teeping*. La commande s est similaire à n mais elle traverse les fonctions sans s'arrêter ligne après ligne dans le code qu'elles contiennent. Pour apprécier la différence de comportement entre s et n, essayez ceci :

```
(gdb) set args readme.txt readme2.txt
(gdb) b main
Breakpoint 1 at 0x8048796 : file wordsplit.c, line 67.
(gdb) r
Starting program : /home/src/wordsplit/wordsplit readme.txt readme2.txt

Breakpoint 1, main (argc=3, argv=0xbffff814) at wordsplit.c :67
67          if (argc < 2) {
(gdb) n
72          for (i = 1; i < argc; i++) {
(gdb) n
73          word_dump (argv[i]);
(gdb) n
Zaphod
has
two
heads
72          for (i = 1; i < argc; i++) {
```

```
(gdb) s
73          word_dump (argv[i]);
(gdb) s
word_dump (filename=0xbffff993 "readme2.txt") at wordsplit.c :13
13          c = 0
(gdb) s
15          f = fopen (filename, "r");
(gdb)
```

Un trait intéressant de gdb est sa capacité à être attaché à des programmes en cours. Essayez la séquence de commandes suivantes :

```
[root@cericon]# lpd
[root@cericon]# ps awx | grep lpd
28157 ?          S          0 :00 lpd Waiting
28160 pts/6      S          0 :00 grep lpd
```

```
[root@cericon]# gdb -q /usr/bin/lpd
(no debugging symbols found)...
(gdb) attach 28157
Attaching to program : /usr/sbin/lpd, Pid 28157
0x40178bfe in __select () from /lib/libc.so.6
(gdb)
```

Le démon `lpd` n'a pas été compilé avec le support de débogage, mais vous pouvez arrêter et déboguer *n'importe quel* processus de votre système. Essayez `bt` pour voir. A présent, libérez le processus avec :

```
(gdb) detach
Detaching from program : /usr/sbin/lpd, Pid 28157
```

Le débogueur fournit une importante quantité d'aide en ligne. La commande `help` peut être lancée à cette fin. Les pages d'info associées à la commande `info gdb` présente également un grand nombre de données qui ne seront pas couvertes ici.

23.2.2 Examiner les fichiers `core`.

Si votre programme présente une erreur de segmentation (*segmentation violation* ou "segfault"), un fichier `core` sera écrit dans le répertoire courant. On l'appelle en anglais un *core dump* (vidange du contenu mémoire). Ce dernier est produit par une bogue du programme en réponse à un signal `SIGSEGV` envoyé au programme parce qu'il tente d'accéder à une partie de la mémoire en dehors de la zone autorisée. Ces fichiers peuvent être examinés en utilisant `gdb` pour révéler (ce qui est la pratique usuelle) où un problème s'est produit. Exécutez simplement `gdb <nom_exécutable> ./core` et alors tapez `bt` (ou tout autre commande `gdb`) derrière l'invite de `gdb`. Si vous saisissez `file ./core`, vous obtiendrez un message similaire à ceci :

```
/root/core : ELF 32-bit LSB core file of '<executable>' (signal 11), Intel 80386, version
1
```

23.2.3 `strace`.

La commande `strace` affiche chaque appel système réalisé par un programme. Un appel système est un appel à une fonction fait par une fonction de la bibliothèque **C** à l'attention du noyau LINUX. Essayez :

```
strace ls
strace ./wordsplit
```

Si un programme n'a pas été compilé avec le support de débogage, la seule

manière d'inspecter son exécution est d'utiliser la commande `strace`. En tout cas, cette commande fournit une information précieuse sur les endroits où le programme échoue et s'avère utile pour diagnostiquer les erreurs.

23.3 Bibliothèques du C.

Nous avons déjà mentionné la bibliothèque standard du **C**. Le langage **C** en soi ne fait pas grand-chose ; tout ce qui est utile se trouve dans des fonctions externes. Celles–ci sont groupées en bibliothèques. La bibliothèque standard du **C** est le fichier `/lib/libc.so.6`. Pour afficher toutes les fonctions de cette bibliothèque, lancez :

```
nm /lib/libc.so.6
nm /lib/libc.so.6 | grep ' T ' | cut -f3 -d' ' | grep -v '^_' | sort
-u \
| less
```

De nombreuses fonctions ont une page de `man` mais, certaines ne présentent pas de documentation si bien qu'il est nécessaire de lire les explications fournies dans les fichiers d'en-têtes. Il vaut mieux ne pas utiliser de fonctions qui ne seraient pas *standards* (c'est-à-dire qui ne sont pas usuelles sur d'autres systèmes).

La création de sa propre bibliothèque est simple. Supposons que nous ayons deux fichiers contenant plusieurs fonctions que nous souhaitons compiler pour les placer dans une bibliothèque. Les fichiers s'appellent `simple_math_sqrt.c` et `simple_math_pow.c` :

```
#include <stdlib.h>
#include <stdio.h>

static int abs_error (int a, int b)
{
   if (a < b) {
      return a - b;
   return b - a;
}
```

```
int simple_math_isqrt (int x)
{
   int result;
   if (x < 0) {
       fprintf (stderr,
       "simple_math_sqrt : taking the sqrt of a negative number\n");
      abort ();
   }
    result = 2;
    while (abs_error (result * result, x) > 1) {
       result = (x / result + result) / 2;
    }
    return result;
}
```

Et pour `simple_math_pow.c`, nous avons :

```
#include <stdlib.h>
#include <stdio.h>

int simple_math_ipow (int x, int y)
{
   int result;
   if (x == 1 || y == 0)
      return 1;
   if (x == 0 && y < 0) {
      fprint (stderr,
       "simple_math_pow : raising zeo to a negative power\n");
      abort ();
   }
   if (y < 0)
      return 0;
   result = 1;
   while (y > 0) {
      result = result * x;
      y = y - 1;
   }
   return result;
}
```

Nous allons appeler notre bibliothèque `simple_math`. Il sera judicieux de nommer toutes les fonctions qu'elle regroupera sous la terminologie `simple_math.-?????`. La fonction `abs_error` ne sera pas utilisée hors du fichier `simple_math/sqrt.c` et, par conséquent, nous plaçons le terme `static` devant sa déclaration, ce qui signifie qu'il s'agit d'une fonction *locale*.

Nous pouvons à présent compiler le code de cette manière :

```
gcc -Wall -c simple_math_sqrt.c
gcc -Wall -c simple_math_pow.c
```

L'option `-c` vaut pour *compile only* (compiler seulement). Le code ne sera pas

converti en exécutable. Les fichiers produits seront caractérisés par une extension `.o`. On les appelle des *fichiers objets*.

Maintenant, il faut *archiver*ces fichiers dans la bibliothèque. A cette fin, on fait usage de la commande `ar` (précurseur de `tar`) :

```
ar r libsimple_math.a simple_math.sqrt.o simple_math_pow.o
ranlib libsimple_math.a
```

Remarquons que le nom de la bibliothèque est précédé de "lib" et que la commande `ranlib` permet d'indicer l'archive.

La bibliothèque est maintenant utilisable. Créons un fichier appelé `mon_test.c` :

```
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    printf ("%d\n", simple_math_ipow (4, 3));
    printf ("%d\n", simple_math_isqrt (50));
    return 0;
}
```

et exécutez :

```
gcc -Wall -c mon_test.c
gcc -L. -o mon_test mon_test.o -lsimple_math
```

La première commande compile le fichier `mon_test.c` en `mon_test.o`. Ensuite, on procède à l'étape d'*édition de liens* qui combine `mon_test.o` et les bibliothèques pour produire un exécutable unique. L'option `L.` signifie qu'il faut regarder dans le répertoire courant pour toute bibliothèque. L'option `-lsimple_math` signifie qu'il faut prendre en compte la bibliothèque `libsimple_math.a` (`lib` et `.a` sont ajoutés automatiquement). Cette opération d'édition de liens est statique [cela n'a rien à voir avec le mot-clé `static` utilisé dans le programme simple_math_sqrt.c] parce qu'elle a lieu avant que le programme ne soit lancé et qu'elle inclut tous les fichiers objets dans un exécutable.

Notons encore qu'il arrive souvent que de nombreuses bibliothèques statiques soient liées à un même programme. Ici, l'ordre est important : la bibliothèque ayant le moins de dépendances devra être citée en dernier ou alors, vous obtiendrez des *erreurs de référencement de symboles* (*symbol referencing errors*).

Nous pouvons aussi créer un fichier d'en-tête `simple_math.h` pour tirer parti de la bibliothèque :

```
/* calculates the integer square root, aborts on error */
int simple_math_isqrt (int x);

/* calculates the integer power, aborts on error */
int simple_math_ipow (int x, int y);
```

Ajoutez la ligne `#include "simple_math.h"` au début du fichier `mon_test.c` :

```
#include <stdlib.h>
#include <stdio.h>
#include "simple_math.h"
```

Cette formulation permet de se débarrasser des messages d'alerte associés à la `déclaration implicite de fonction` (`implicit declaration of function`). D'habitude, on utilise une forme du type `#include <simple_math.h>`, mais en l'occurrence, il s'agit d'un fichier d'en-tête d'un répertoire courant (notre *propre* fichier d'en-tête) et, c'est le cas précisément où il faut utiliser `"simple_math.h"` au lieu de `<simple_math.h>`.

23.4 Les projets en C – `Makefiles`.

Supposons que nous ne fassions qu'un changement sur un seul des fichiers sources (comme cela se fait souvent au cours du développement). Faut-il tout recompiler à chaque fois ? On pourrait penser à un script effectuant la compilation et l'édition de liens. Cependant, ce script réaliserait l'ensemble de toutes les opérations sans agir sélectivement sur le fichier modifié. En réalité, nous avons besoin d'un utilaire qui ne recompile que les sources modifiées. Cet utilitaire existe et s'appelle `make`.

`make` est un programme qui analyse un fichier `Makefile` dans le répertoire courant et effectue les opérations de compilation et d'édition de liens. Les fichiers `Makefile` contiennent une liste de règles et des *dépendances* décrivant comment construire un programme.

A l'intérieur d'un fichier `Makefile`, vous devez déclarer d'une part une liste de dépendances que `make` peut exploiter, et d'autre part, des commandes de shell.

23.4.1 Compléter notre exemple par un `Makefile`.

Notre première *dépendance* dans le processus pour compléter la compilation de `mon_test` concerne à la fois la bibliothèque `libsimple_math.a` et le fichier objet `mon_test.o`. Dans les termes propres à `make`, nous créons une ligne de `Makefile` comme ceci :

```
mon_test :    libsimple_math.a mon_test.o
```

ce qui signifie simplement que les fichiers `libsimple_math.a` et `mon_test.o` doivent exister et être mis-à-jour avant `mon_test`. L'expression `mon_test :` est appelée la *cible* de `make` (*make target*). Au-dessous de cette ligne, nous devrons aussi déclarer comment construire `mon_test` :

```
	gcc -Wall -o $@ mon_test.o -L. -lsimple_math
```

Le signe `$@` désigne le nom de la cible elle-même, et donc il sera substituée par `mon_test`. *Il est impératif de noter que l'espace avant `gcc` est une tabulation (et* ***non*** *une suite de huit espaces).*

Ensuite, on sait que `libsimple_math.a` a une dépendance vis-à-vis de `simple_math_sqrt.o` et de `simple_math_pow.o`. Une fois de plus, nous avons une dépendance qui peut-être décrite avec un script de shell pour construire la cible :

```
libsimple_math.a : simple_math_sqrt.o simple_math_pow.o
        rm -f $@
        ar rc $@ simple_math_sqrt.o simple_math_pow.o
        ranlib $@
```

Notons que la marge à gauche est une seule tabulation (et non une suite de huit espaces).

La dernière dépendance est celle des fichiers `simple_math_sqrt.o` et `simple_math_pow.o` qui dépendent de `simple_math_sqrt.c` et `simple_math.pow_c`, respectivement. Ceci requiert deux règles de `make`. Cependant, make présente un raccourci qui fonctionne dans le cas de nombreux fichiers sources **C** :

```
.c.o :
        gcc -Wall -c -o $*.o $<
```

Ceci signifie que tous les fichiers `.o` doivent être construits à partir des fichiers `.c` du même nom au moyen de la commande `gcc -Wall -c -o $*.o $<`, où `$*.o` désigne le nom du fichier object et `$<` le nom du fichier dont dépend `$*.o` et ce, séquentiellement, pour tous les fichiers impliqués.[5]

23.4.2 Combiner le tout.

En fait, les fichiers `Makefile` peuvent avoir leurs règles placées dans un ordre arbitraire, mais il est astucieux d'établir les règles les plus évidentes au début pour des raisons de lisibilité.

Il existe aussi une règle qui devrait toujours valoir dès le début :

```
all :   libsimple_math.a mon_test
```

La cible `all :` constitue la règle à laquelle `make` essaie de satisfaire quand il est lancé sans arguments en ligne de commande. Cela signifie juste que `libsimple_math.a` et `mon_test` sont les deux derniers fichiers à être construits. Ils sont donc au sommet de la série des dépendances.

Les `Makefile`s ont aussi leur propre forme de variables d'environnements, à l'instar des scripts de shells. Vous pouvez constater que nous avons utilisé le texte `simple_math` dans trois de nos règles. Il est donc pertinent de définir une *macro*.

Notre `Makefile` final est donc :

[5] NdT : cette notation `.c.o` est aujourd'hui remplacée par `%.o : %.c` (où l'espacement entre le signe `:` et `%` est une tabulation)

```
# Comments start with a # (hash) character like shell scripts.
# Makefile to build libsimple_math.a and mon_test program.
# Paul Sheer <psheer@icon.co.za> Sun Mar 19 15 :56 :08 2000

OBJS    = simple_math_sqrt.o simple_math_pow.o
LIBNAME = simple_math
CFLAGS  = -Wall

all :    lib$(LIBNAME).a mon_test

mon_test :   lib$(LIBNAME).a mon_test.o
         gcc $(CFLAGS) -o $@ mon_test.o -L. -l${LIBNAME}

lib$(LIBNAME).a : $(OBJS)
        rm -f $@
        ar rc $@ $(OBJS)


        ranlib $@

.c.o :
       gcc $(CFLAGS) -c -o $*.o *<

clean :
       rm -f *.o *.a mon_test
```

A présent nous pouvons taper :

```
make
```

dans le répertoire courant pour lancer la construction des différents fichiers.

Vous pouvez voir que nous avons ajouter une autre cible appelée `clean :`. Les cibles peuvent être exécutées explicitement en ligne de commande comme suit :

```
make clean
```

ce qui supprimera tous les fichiers construits.

Les fichiers `Makefile` ont bien d'autres usages que la construction de programmes **C**. Tout ce qui doit être construit à partir de sources peut s'appuyer sur un `Makefile` pour faciliter la mise en place.[6]

[6] NdT : notons qu'il existe un excellent tutoriel complémentaire à l'information présente. On le trouve à l'adresse : `http ://gl.developpez.com/makefile`

Chapitre 24

Bibliothèques partagées.

Ce chapitre est un prolongement direct de l'élaboration des bibliothèques statiques à extension `.a` comme cela a été vu au Chapitre 23. Nous y discutons la création et l'installation de *bibliothèques liées dynamiquement* (DLLs ou *Dynamically Linked Libraries*). Ainsi, nous allons avoir une vue d'ensemble de bon niveau technique à propos du fonctionnement des DLLs sur UNIX. Dans ce qui est discuté ci-après, vous ne devrez pas oublier `ldconfig` et `LD_LIBRARY_PATH`.

Les fichiers `.a` des bibliothèques liées statiquement sont utiles pour créer des fonctions que de nombreux programmes incorporent. Cette manière de procéder est connue sous le terme de *réutilisabilité du code* (*code reuse*). Cependant, si on analyse un peu la manière dont un fichier est lié (en réalité, incorporé) dans un exécutable comme `mon_test` (au chapitre 23), on s'aperçoit vite que la taille de ce dernier fichier a augmenté d'exactement la taille de `libsimple_math.a`. Si des centaines de programmes utilisent le même fichier `.a`, le code est dupliqué à chaque utilisation. Une telle inefficacité a été considérée comme rédhibitoire bien avant la naissance de LINUX. Des fichiers de bibliothèque ont été inventés, qui se lient seulement lorsqu'un programme qui y fait appel est exécuté. Cette manière de procéder est basée sur le *chargement dynamique*. Au lieu d'avoir à faire à des fichiers `.a`, on utilise dans ce cas des fichiers `.so` (*shared object*) se trouvant dans `/lib` et `/usr/lib`. Le chargement se fait de manière automatique lorsqu'un programme est exécuté.

24.1 Création de fichiers DLL.`so`.

La création d'une DLL requiert quelques modifications dans le fichier `Makefile` vu à la sous-section 23.4.2 :

```
OBJS      = simple_math_sqrt.o simple_math_pow.o
LIBNAME   = simple_math
SONAME    = libsimple_math.so.1.0.0
SOVERSION = libsimple_math.so.1.0
CFLAGS    = -Wall

all :    libs$(LIBNAME).so mon_test
```

```
mon_test : libs$(LIBNAME).so mon_test.o
          gcc $(CFLAGS) -o $@ mon_test.o -L. -l${LIBNAME}

lib$(LIBNAME).so : $(OBJS)
        gcc -shared $(CFLAGS) $(OBJS) -lc -Wl,-soname -Wl,$(SOVERSION) \
                                  -o $(SONAME) && \
        ln -sf $(SONAME) $(SOVERSION) && \
        ln -sf $(SONAME) lib$(LIBNAME).so

.c.o :
       gcc -fPIC -DPIC $(CFLAGS) -c -o $*.o *<

clean :
       rm -f *.o *.a *.so mon_test
```

L'option `-shared` de `gcc` construit notre bibliothèque partagée. Les options `-W` sont des options de l'éditeur de liens qui fixent le numéro de version de la bibliothèque que les programmes chargent lors de l'exécution. Les termes `-fPIC` et `-DPIC` indiquent qu'il faut produire un code indépendant de la position (*position-independant code*), c'est-à-dire un code capable de réaliser du chargement dynamique.

Après l'exécution de `make`, nous obtenons :

```
lrwxrwxrwx    1 root     root           23 Sep 17 22 :02    libsimple_math.so -> libsimple_math.so.1.0.0
lrwxrwxrwx    1 root     root           23 Sep 17 22 :02    libsimple_math.so.1.0 -> libsimple_math.so.1.0.0
-rwxr-wr-w    1 root     root         6046 sep 17 22 :02    libsimple_math.so1.0.0
-rwxr-xr-x    1 root     root        13677 Sep 17 22 :02    mon_test
```

24.2 Versions de DLL .

Vous avez remarqué que nos trois fichiers `.so` sont similaires à de nombreux fichiers de `/lib` et de `/usr/lib`. Ce système complexe d'édition de liens et de création de liens symboliques est une partie du processus de *définition des versions des bibliothèques* (*library versioning*). Bien que la création d'une bibliothèque partagée soit hors du cadre des tâches usuelles d'administration, la définition des versions des bibliothèques est importante à comprendre.

Les DLLs présentent un problème. Supposons qu'une DLL soit boguée ou obsolète. Si le fichier DLL est écrasé par un fichier à jour, l'ensemble des applications ayant recours à cette DLL sera affecté. Si ces applications dépendent de certains comportements du code contenu dans cette DLL, elles planteront très certainement à l'appel de la nouvelle version. UNIX a résolu le problème de manière élégante en permettant à de multiple versions des DLLs d'être présentes simultanément. Les programmes possèdent eux-mêmes le numéro de version à utiliser. Essayez :

```
ldd mon_test
```

Ceci indique les fichiers DLL que `mon_test` peut lier (il est programmé pour cela) :

```
libsimple_math.so.1.0.0 => ./libsimple_math.so.1.0.0 (0x40018000)
libc.so.6 => /lib/libc.so.6 (0x40022000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Pour l'instant, nous sommes intéressé par `libsimple_math.so.1.0`. Observez bien la manière dont est établie la correspondance de la variable `SOVERSION` dans `Makefile`. Remarquez aussi comment nous avons choisi notre lien symbolique. Nous permettons effectivement à `mon_test` de se lier à toute future bibliothèque `libsimple_math.so.1.0.?` (ce qui fait que la bibliothèque `simple_math` peut être mise à jour sous forme d'une nouvelle version). Ceci a pu être réalisé grâce au lien symbolique. Cependant, le lien ne se produira pas avec une version du type `libsimple_math.so.1.1.?` par exemple. Comme développeurs de `libsimple_math`, nous avons décidé que les bibliothèques ayant un numéro de version mineur différent seront incompatibles, tandis que les bibliothèques de niveau de patch différent seront compatibles [dans notre exemple, nous avons considéré que le nom d'un bibliothèque était du type `lib`*name*`.so.`*majeur.mineur.patch*].

Nous pourrions changer `SOVERSION` en `libsimple_math.so.1`. Ceci reviendrait à dire que les futures bibliothèques de numéros de version mineure différents sont compatibles ; seule une modification du numéro majeur engendrerait l'incompatibilité.

24.3 Installation de fichiers DLL.so.

Si vous lancez `./mon_test`, vous serez accueillis par un message tel qu'`error loading shared libraries`. La raison en est que l'éditeur de liens dynamiques ne recherche pas les fichiers `.so` dans le répertoire courant. Pour exécuter le programme, vous devrez installer votre bibliothèque :

```
mkdir -p /usr/local/lib
install -m 0755 libsimple_math.so libsimple_math.so.1.0 \
             libsimple_math.so.1.0.0 /usr/local/lib
```

Editez alors le fichier `/etc/ld.so.conf` et ajoutez une ligne :

```
/usr/local/lib
```

Reconfigurez vos bibliothèques :

```
ldconfig
```

Enfin, lancez votre programme avec :

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH :/usr/local/lib"
./mon_test
```

`ldconfig` configure toutes les bibliothèques du système. Elle recrée les liens symboliques de manière appropriée (comme nous l'avons fait manuellement) et

reconstruit un cache qui sera consulté ultérieurement. Les répertoires de bibliothèques qu'elle prend en compte sont `/lib`, `/usr/lib` et ceux repris dans le fichier `/etc/ls.so.config`. La commande `ldconfig` devrait être exécutée automatiquement lorsque le système est réamorcé et manuellement, chaque fois que des bibliothèques sont installées et mises à jour.

La variable d'environnement `LD_LIBRARY_PATH` est appropriée pour chaque exécutable du système et similaire à la variable d'environnement `PATH`. `LD_LIBRARY_PATH` indique quels répertoires seront recherchés pour les fichiers des bibliothèques. Notez que même si `LD_LIBRARY_PATH` est désactivée, la recherche se fera sur base de `/lib` et `/usr/lib`.

Chapitre 25

Paquets sources et binaires.

Dans ce chapitre, nous allons apprendre à installer des paquets logiciels à partir des sources, en nous appuyant sur notre connaissance des `Makefile` vus au chapitre 23. Cependant, en majorité, les paquets se présentent sous la forme de fichiers .rpm (RedHat) ou .deb (Debian).

25.1 Construction de paquets sources GNU.

Presque tous les paquets proviennent de sources **C** (archivées par la commande `tar` et compressées) disponibles à partir d'un des sites FTP publiques comme `metalab.unc.edu`. Les développeurs attentifs ont préparé des paquets conformes aux normes de GNU (*GNU standards compliant*). Ceci veut dire que le désarchivage du paquet produit les fichiers suivants dans un répertoire principal (le plus souvent de même nom que celui du paquet) :

INSTALL C'est le document de référence qui commence par la ligne "`These are generic installation instructions.`" Etant donné que tous les paquets GNU sont installés de la même manière, ce fichier devrait toujours être le même d'un paquet à l'autre.

NEWS Ce sont les commentaires (relatifs aux nouveautés) à l'attention des utilisateurs.

README Toute information essentielle y est contenue. Il s'agit traditionnellement des explications à propos de ce que réalise le programme, le matériel promotionnel et toute information nécessaire à l'installation du paquet.

COPYING Ce fichier contient la licence GNU-GPL (GNU General Public Licence).

AUTHORS Ce fichier contient la liste des personnes contribuant au projet.

ChangeLog Il s'agit d'une liste de format spécial contenant un historique de tous les changements faits sur le paquet, les noms de personnes l'ayant modifié et la date des modifications. Ce fichier est utilisé pour tracer les modifications effectuées au cours du temps.

Le fait qu'un paquet obéisse au standard de GNU signifie aussi qu'il doit pouvoir être installé à l'aide des trois commandes :

```
./configure
make
make install
```

Cela implique également que les paquets doivent pouvoir être compilés sur n'importe quel système UNIX. Par conséquent, cette section va vous apprendre à utiliser des paquets LINUX pouvant être installés et pouvant fontionner sur des systèmes non-LINUX.

Nous allons choisir un exemple à titre d'illustration. Commençons par télécharger `cooledit` à partir du site `metalab.unc.edu`. dans le répertoire `/pub/Linux/apps/editors/X/cooledit`, en utilisant `ftp`.[1] Maintenant, entrez :

```
cd /opt/src
tar -zxvf cooledit-3.17.5.tar.gz
cd cooledit-3.17.5
```

Vous noterez que le nom de la plupart des sources est du type *package-major.minor.patch*`.tar.gz`. Le nombre majeur n'est modifié que lorsque les développeurs ont réalisé des changements substantiels ou lorsqu'une version présente des incompatibilités vis-à-vis des versions précédentes. Le nombre mineur est mis à jour lors de modifications peu importantes. Le numéro de correctif (*patch*, aussi connu sous le terme *niveau* de correctif) correspond à une mise à jour vers toute nouvelle version (*release*) et usuellement, dans ce cas, il y a des corrections de bogues.

A ce stade, vous pouvez appliquer au besoin tous les patches dont vous disposez. Référez-vous à la section 21.7.3.

Vous pouvez alors lancer la commande `./configure`. Le script `./configure` est produit par `autoconf`, un paquet utilisé par les développeurs pour créer un code source **C** qui compilera le paquet sur tout type d'UNIX. Le paquet `autoconf` contient également les *GNU Coding Standards* auxquels tous logiciels devraient se conformer [`autoconf` est le fruit du travail remarquable de David MacKenzie. J'ai souvent entendu le mythe selon lequel les systèmes Unix avaient tellement divergé au cours du temps qu'ils n'étaient plus compatibles. Le fait qu'un logiciel évolué comme `cooledit` (et de nombreux autres) soit compilable sur tous les UNIX devrait dissiper ce non-sens. Il y a aussi un exagération à propos des developpeurs "portant" des logiciels UNIX commerciaux sous LINUX. Si pour commencer, ils avaient écrits leur logiciel proprement, il n'y aurait pas de "portage" à réaliser. En bref, *tous* les logiciels LINUX fonctionnent sur tous les UNIX. Les seules exceptions sont des paquets qui exploitent des caractéristiques propres au noyau LINUX].

```
./configure --prefix=/opt/cooledit
```

En l'occurrence, `--prefix` indique le répertoire-parent sous lequel le paquet sera installé (voir la section 18.2). Essayez ensuite :

```
./configure --help
```

[1] NdT : par exemple en lançant `ftp`, puis en entrant `open metalab.unc.edu`.

pour obtenir les options spécifiques au paquet.

Une autre astuce consiste à fixer les options de compilation :

```
CFLAGS='-O2 -fomit-frame-pointer -s -pipe' ./configure --prefix=/opt/cooledit
```

Ces options ont la signification suivante :

-O2 fixe les options de compilation pour "rendre le paquet compilé aussi rapide que possible sans en augmenter la taille" (`-O3` ne produit pratiquement jamais d'amélioration).

-fomit-frame-pointer permet au compilateur d'utiliser un registre supplémentaire qui, normalement, est dévolu au débogage. N'utilisez cette option que lorsque vous êtes absolument sûr que vous ne devrez pas analyser des problèmes associés à l'installation d'un paquet.

-s élimine le code objet. Ceci réduit la taille du code objet en supprimant toute donnée de débogage.

-pipe indique de ne pas utiliser de fichiers temporaires. En fait, cette option permet d'utiliser des tubes pour distribuer le code parmi les différents étages de compilation. Elle s'avère utile pour accélérer la compilation.

A présent, il faut compiler le code, ce qui peut prendre jusqu'à plusieurs heures en fonction de la quantité de code et de la puissance de la CPU [`cooledit` est compilé en moins de 10 minutes sur toute machine de base existant au moment de la rédaction de la version originale anglaise de cet ouvrage (NdT : 2001)].

```
make
```

Vous pouvez aussi lancer :

```
make CFLAGS='-O0 -g'
```

si vous décidez de compiler avec l'aide de débogage. L'installation se poursuit avec :

```
make install
```

Voici une jolie astuce [qui n'est pas toujours supportée] pour installer le paquet dans un sous-répertoire différent :

```
mkdir /tmp/cooledit
make install prefix=/tmp/cooledit
```

Vous pouvez utiliser ces commandes pour empaqueter la construction ainsi réalisée afin de la désarchiver sur un système différent. Cependant, vous ne devrez pas exécuter le programme à partir d'un répertoire différent de celui pour lequel il a été `--prefix`é durant l'étape d'installation. En effet, en majorité, les paquets *sont compilés à* cet endroit et accèdent aux données installées en fonction de cela.

L'utilisation d'un paquet source est souvent le meilleur moyen de pratiquer une installation quand vous voulez qu'un paquet fonctionne de la manière ima-

ginée à l'origine par ses développeurs. Il vous sera aussi souvent nécessaire de trouver de la documentation car il se peut que les distributeurs aient oublié d'inclure certains fichiers.

25.2 Paquets binaires RedHat et Debian.

Dans cette section, nous plaçons les exemples associées à Debian entre parenthèses, `(...)`. Etant donné que les exemples sont utilisés sur des systèmes réels, il se peut qu'il n'y ait pas de correspondance stricte avec le vôtre.

25.2.1 Version des paquets.

La numération des versions de paquets logiciels de RedHat et Debian est souvent réalisée comme suit (quoique ceci soit loin d'être une règle) :

```
nom_de_paquet>-<version_du_source>-<version_du_paquet>.<plateforme_materielle>.rpm
( <nom_du_paquet>_<version_du_source>-<version_du_paquetage>.deb )
```

Par exemple,

```
bash-1.14.7-22.i386.rpm
( bash_2.03-6.deb )
```

est le Bourne Shell Again que vous utilisez, version majeure 1, version mineure 14, patch 7, version de paquet 22, compilé sur un processeur Intel 386. Parfois, les paquets Debian présenteront une indication sur l'architecture, ajoutée au numéro de la version. Il se peut qu'on ait, par exemple, `bash_2.03-6_i386.deb`.

Le terme `<version_du_source>` est la version du fichier original `.tar` (comme ci-dessus). Le terme `<version_du_paquet>`, également appelé version (ou *release*), se rapporte au fichier `.rpm` lui-même ; dans ce cas,`bash-1.14.7-22.i386.rpm` a été empaqueté pour la 8$^{\text{ième}}$ fois, probablement avec des améliorations mineures. Le terme i386 décrit l'architecture. Il pourrait s'agir de *SPARC* [un type de processeur utilisé par les stations de travail Sun Microsystems], `ppc` pour *PowerPC* [un autre type de station de travail non-Intel], `alpha` pour une station *DEC Alpha* [un type de station de travail/serveur à 64 bits, de haut-niveau], ou tout autre architecture.

25.2.2 Installation, mise-à-jour, suppression.

Pour installer un paquet, exécutez la commande suivante sur le fichier `.rpm` ou `.deb` :[2]

```
rpm -i mirrodir-0.10.48-1.i386.rpm
( dpkg -i mirrodir_0.10.48-2.deb )
```

[2] NdT : concernant la distribution Mandriva (`.rpm`), il existe une commande `urpmi` qui permet l'installation et la désintallation via le nom de fichier sans numéro de version, ni d'extension `.rpm`. Pour les paquets de la distribution Debian, les commandes modernes sont présentées à la section 25.2.8.

La mise-à-jour peut être réalisée à l'aide des commandes suivantes (Debian choisit une mise-à-jour automatiquement si le paquet est déjà présent sur le système) :

```
rpm -U mirrodir-0.10.48-1.i386.rpm
( dpkg -i mirrodir_0.10.49-1.deb )
```

Ensuite, l'installation est complétée avec :

```
rpm -e mirrodir
( dpkg --purge mirrodir )
```

Dans le cas de Debian, une procédure d'élimination de paquet (*package removal*) ne supprime pas les fichiers de configuration, ce qui autorise donc la réversibilité d'une réinstallation future.

```
dpkg -r mirrodir
```

Si vous devez réinstaller un paquet (peut-être en raison d'un fichier corrompu), utilisez :

```
rpm -i --force python-1.6-2.i386.rpm
```

Debian réalise la réinstallation automatiquement si le paquet est présent sur le système.

25.2.3 Dépendances.

Les paquets requièrent souvent d'autres paquets qui doivent être installés au préalable. La base de données des paquets conserve une trace de ces *dépendances* (*dependencies*). Il se peut aussi que vous obteniez une erreur du type : `error : failed dependencies :` (ou `dependency problems` dans le cas de Debian) alors que vous tentez une installation. Ceci signifie que d'autres paquets doivent être installés au préalable. Le même problème survient lorsque vous essayez de supprimer des paquets. Si deux paquets au moins ont besoin l'un de l'autre, vous devez les placer tous les deux sur la même ligne de commande lors de l'étape d'installation. Parfois, il arrivera qu'un paquet requière quelque chose qui n'est pas essentiel ou qui est déjà fourni par un paquet logiciel équivalent. Par exemple, un programme pourra requérir l'installation de `sendmail` alors qu'`exim` constitue un substitut adéquat. Dans de tels cas, l'option `--nodeps` permet de passer la vérification des dépendances.

```
rpm -i --nodeps <fichier_rpm>
( dpkg -i --ignore-depends=<paquet-requis> <fichier-deb> )
```

Notez que Debian est beaucoup plus tatillon en terme de dépendances. Ne les écrasez donc pas à tort et à travers.

25.2.4 Examen de paquets.

Les paquets `.rpm` et `.deb` constituent bien plus qu'une simple méthode d'archivage des fichiers. Autrement, vous pourriez tout aussi bien n'utiliser que les fichiers sous leur forme `.tar`. Chaque paquet possède une liste enregistrée auprès d'une base de données et, qui peut être consultée. Ce qui suit indique les principales vérifications susceptibles d'être utilisées sur des *paquets installés* :

```
rpm -qa
( dpkg -l '*' )
```

Pour rechercher le nom d'un paquet :

```
rpm -qa | grep <expression_reguliere>
( dpkg -l <expression_globale> )
```

Essayez :

```
rpm -qa | grep util
( dpkg -l '*util*' )
```

Pour vérifier l'existence d'un paquet, par exemple `textutils` (query, list), tapez :

```
rpm -q textutils
( dpkg -l textutils )
```

ce qui retournera le nom et la version :

```
textutils-2.0e-7
( ii textutils     2.0-2     The GNU text file processing utilities. )
```

Pour obtenir des informations sur un paquet :

```
rpm -qi <paquet>
( dpkg -s <paquet> )
```

Pour lister les bibliothèques et les autres paquets nécessaires au logiciel :

```
rpm -qR <logiciel>
( dpkg -s <logiciel> | grep Depends )
```

Pour afficher quels paquets requièrent un paquet donné (notez qu'avec Debian, nous pouvons vérifier cela en essayant d'enlever le "paquet donné" avec l'option `--no-act` pour signifier "tester seulement"),

```
rpm -q --whatrequires <paquet_donne>
( dpkg --purge --no-act <paquet_donne> )
```

25.2.5 Liste et examen de fichiers.

Pour obtenir une liste des fichiers contenus dans un paquet [à nouveau, ceci ne vaut que pour des fichiers intallés] :

```
rpm -ql <paquet>
( dpkg -L <paquet> )
```

Les listes de fichiers associés aux paquets sont très utiles pour trouver les commandes et la documentation associées à un logiciel. Les utilisateurs sont souvent frustrés par un paquet pour lequel "ils ne savent pas à quoi s'attendre". Un bon point de départ consiste donc à afficher les fichiers constituant le paquet.

Pour déterminer à quel paquet appartient un fichier :

```
rpm -qf <nom_de_fichier>
( dpkg -S <nom_de_fichier> )
```

Par exemple, `rpm -qf /etc/rc.d/init.d/httpd` (ou `rpm -qf /etc/init.d/httpd`) retournera par exemple `apache-mod_ssl-1.3.12.2.6.6-1`, et `rpm -ql fileutils-4.0w-3 | grep bin` fournit une liste de toutes les commandes associées à `fileutils`. Voici une astuce pour trouver tous les fichiers apparentés à une commande de votre `PATH` :

```
rpm -ql `rpm -qf \ `which --skip-alias <commande> \`
( dpkg -L `dpkg -S \`which <commande> \` | cut -fl -d :` )
```

25.2.6 Vérification des paquets.

Il arrivera bien que vous vouliez rechercher les fichiers d'un paquet qui ont été modifiés depuis l'installation (éventuellement par un programmeur ou un administrateur incompétent). La vérification de tous les paquets est très consommatrice de temps, mais elle donne des informations très précieuses :

```
rpm -V `rpm -qa`
( debsums -a )
```

Cependant, il n'existe pas encore de méthode pour affirmer que le paquet installé est le vrai paquet (voir la section 45.3.2). Pour procéder à une telle vérification, vous devez obtenir le vrai paquet `.deb` ou `.rpm` et opérer la vérification de cette manière :

```
rpm -Vp openssh-2.1.1p4-1.i386.rpm
( debsums openssh_2.1.1p4-1_i386.deb )
```

Finalement, même si vous avez le paquet sous la main, comment pouvez-vous être sûr qu'il s'agit bien du paquet d'origine des développeurs et qu'aucun cheval de Troie (en anglais *trojan*) n'y est incorporé ? Utilisez la commande `md5sum` :

```
md5sum openssh-2.2.2p4-1.i386.rpm
( md5sum openssh-2.2.2p4-1.i386.deb )
```

La commande `md5sum` utilise un algorithme mathématique (*MD5*) pour calculer une valeur numérique basée sur le contenu du fichier, en l'occurrence `8e8d8e95db7fde99c09e1398e4dd3468`. Cette méthode est semblable à celle utilisée pour le cryptage des mots de passe (voir la section 12.3). Il n'existe pas de méthode permettant de forger un paquet truqué de même MD5 qu'un paquet original. Donc, les développeurs publient les résultats de leurs sommes md5 (`md5sum`) sur leurs pages web et il vous suffit de faire une comparaison entre le nombre qu'ils donnent et celui que vous retourne la commande `md5sum`.

25.2.7 Examens spéciaux.

Si vous voulez examiner un fichier particulier dans un paquet qui n'a pas encore été installé, utilisez par exemple :

```
rpm -qp -qf '[%{VERSION}\n]' <fichier-rpm>
( dpkg -f <fichier-debian> Version )
```

En l'espèce, `VERSION` est un mot-clé (*tag*) applicable aux fichiers `.rpm`. Voici une liste d'autres mots-clés qui peuvent être utilisés :

BUILDHOST	OBSOLETES	RPMTAG_PREUN
BUILDTIME	OS	RPMVERSION
CHANGELOG	PACKAGER	SERIAL
CHANGELOGTEXT	PROVIDES	SIZE
CHANGELOGTIME	RELEASE	SOURCERPM
COPYRIGHT	REQUIREFLAGS	SUMMARY
DESCRIPTION	REQUIRENAME	VENDOR
DISTRIBUTION	REQUIREVERSION	VERIFYSCRIPT
GROUP	RPMTAG_POSTIN	VERSION
LICENSE	RPMTAG_POSTUN	
NAME	RPMTAG_PREIN	

Pour Debian, `Version` est un champ de contrôle. Les autres sont :

Conffiles	Maintainer	Replaces
Conflicts	Package	Section
Depends	Pre-Depends	Source
Description	Priority	Status
Essential	Provides	Suggests
Installed-Size	Recommends	Version

Par ailleurs, il est possible d'extraire tous les scripts `config` et les fichiers de contrôle d'un paquet `.deb` à l'aide de :

```
dpkg -e <fichier-debian> <repertoire-resultat>
```

Cette commande crée un répertoire `<repertoire-resultat>` et y place les fichiers extraits du paquet. Vous pouvez aussi pratiquez de cette manière :

```
dpkg --fsys-tarfile <fichier_debian>
```

ou, dans le cas d'un fichier `.rpm` :

```
rpm2cpio <fichier_rpm>
```

Enfin, la liste des fichiers d'un paquet peut être sondée avec :

```
rpm -qip <fichier-rpm>
( dpkg -I <fichier-deb> )
rpm -qlp <fichier-rpm>
( dpkg -c <fichier-deb> )
```

ce qui est analogue aux recherches effectuées sur les paquets intallés.

25.2.8 `dpkg`/`apt` et `rpm`.

Jusqu'à présent, nous n'avons donné qu'une petite idée de la gestion des paquets Debian. Debian possède deux outils très performants : *APT* (*Advanced Package Tool*, qui comprend les commandes `apt-cache`, `apt-cdrom`, `apt-config` et `apt-get`) et `dselect` qui est un sélecteur intéractif de paquets, fonctionnant en mode texte. Après votre première installation de Debian, il se peut que vous ayez supposé qu'il fallait lancer `dselect` (il existe des interfaces graphiques [*front-ends*] que vous pouvez rechercher sur *Fresh Meat* `http ://freashmeat.net`) et que vous ayez installé et configuré des logiciels que vous aviez passés durant l'installation proprement-dite de la distribution. Entre ces deux types d'installation, vous pouvez entreprendre diverses actions avancées qui vous permettent d'économiser du temps, comme, par exemple, résoudre récursivement les dépendances via des téléchargements automatisés. *APT* vous permet de faire cela en trouvant les paquets, les dépendances et en installant les paquets proprement. Consultez `apt`(8), `sources.list`(5) et `apt.conf`(5) pour davantage d'information.

Il existe aussi de nombreuses applications graphiques interactives permettant la gestion des paquets *RPM*. La plupart sont purement "cosmétiques".

L'expérience vous montrera clairement la supériorité des paquets Debian sur la plupart des autres.[3] Vous noterez aussi que les distributions inspirées de RedHat ont choisi une sélection des paquets que leurs contributeurs pensent que *vous* trouverez utiles. Debian a des centaines de mainteneurs volontaires sélectionnant ce qu'*ils* trouvent utiles. Ceci dit, presque tous les paquets libres de type UNIX sur l'internet ont déjà été inclus dans Debian.

25.3 paquets sources.

La vie des paquets binaires de RedHat et Debian commence sous forme de fichiers sources à partir desquels les versions binaires sont ultérieurement compi-

[3] NdT : on notera que la distribution Gentoo possède un très beau système de gestion des paquets basé sur `Portage`, qui combine les avantages de Debian et de BSD. La principale commande d'installation et de gestion des paquets est `emerge`. Voir `http ://www.gentoo.org`

lées. Les paquets sources de RedHat ont pour extension `.src.rpm`. Les sources de Debian apparaissent dans l'arborescence des sources de la distribution. Le `HOWTO-RPM` détaille la construction des paquets sources de RedHat. Les paquets `dpkg-dev` et `packaging-manual` de Debian contiennent une référence complète des standards et des méthodes utilisés par cette distribution (essayez `dpkg - L dpkg-dev` et `dpkg -L packaging-manual`).

La construction de paquets sources pour les distributions RedHat et Debian n'est pas couverte dans cet ouvrage.[4]

[4] NdT : concernant Gentoo, le lecteur se référera à la section de développement des paquets `.e-build` à l'adresse `http ://www.gentoo.org/doc/fr`

Chapitre 26

Introduction à IP.

IP est l'acronyme d'*Internet Protocol*. Il s'agit d'un protocole de transmission de données sur l'internet (c'est-à-dire un ensemble de règles à respecter pour pouvoir transmettre ou recevoir des données).

26.1 Communication internet.

Au niveau matériel, les cartes-réseau ont la capacité de transmettre des *paquets* de données (également appelés *datagrammes*) d'une machine à une autre. Un paquet contient un petit bloc de données de l'ordre d'un kilo-octet. Ceci contraste avec les lignes série qui effectuent des transmissions continuement. Toute communication internet repose sur la transmission de paquets, voyageant de manière intacte, même entre deux machines se situant aux antipodes de la planète.

Chaque paquet contient un en-tête de 24 octets (1 octet = 8 bits) ou plus qui précède les données. Donc, la quantité de données qui transite sur le câble a une taille légèrement plus grande qu'un Ko. Pour qu'un paquet soit transmis, l'en-tête doit bien évidemment contenir l'adresse de la machine destinataire. Chaque machine est donc caractérisée par une adresse IP unique qui est un nombre de 32 bits. Il n'existe pas de machine sur l'internet qui ne possède d'adresse IP.

Les octets de l'en-tête sont repris dans le tableau 26.1.

La **version** du protocole est actuellement 4, bien que la *prochaine génération d'IP* (ou version 6) soit en cours de déploiement. Le terme **IHL** désigne la longueur de l'en-tête divisée par 4. Le terme **TOS** (*Type of Service*) représente un champ quelque peu ésotérique relatif à l'ajustement des performances. Il ne sera pas détaillé ici. Le champ intitulé **Length** représente la taille en octets du paquet dans son entiéreté en ce compris l'en-tête. Le terme **Source** désigne l'adresse IP dont provient le paquet et **Destination** est l'adresse IP *vers* laquelle le paquet est envoyé.

La description que nous venons de faire représente la vision qu'une machine a de l'internet. Cependant, l'internet consiste physiquement en un ensemble de petits réseaux à haute vitesse appelés *Local Area Networks* ou *LANs*. Ceux-ci sont interconnectés par des liens physiques portant à de longues distances ; ils sont donc plus lents. Sur un LAN, le moyen de transmission n'est pas un paquet mais une trame (*frame*). Les trames sont similaires aux paquets (les deux pos-

TAB. 26.1 – Octets des en-têtes IP.

Octets	Description
0	Bits de 0 à 3 : Version, bits de 4 à 7 : Internet Header Length (IHL)
1	Type de service (TOS)
2-3	Longueur
4-5	Identification
6-7	Bits 0 -3 : indicateurs (*flags*) ; bits 4-15 : décalage ou offset
8	Temps de vie (*Time To Live* ou TTL)
9	Type
10-11	Somme de contrôle (*checksum*)
12-15	Adresse IP source
16-19	Adresse IP de destination
20-IHL*4-1	Options + remplissage pour grouper jusqu'à 4 octets
	Les données débutent à IHL*4 et finissent à la longueur -1

sèdent un en-tête et un corps de données) mais leur taille est adaptée par souci d'efficacité aux particularités du matériel. Les paquets IP sont encapsulés dans des trames : chacun d'eux est intégré dans la partie **Data** de la trame. Cependant, une trame peut présenter une taille insuffisante pour contenir un paquet entier. Dans ce cas, le paquet IP est morcelé en sous-paquets. Ce groupe de sous-paquets IP reçoit un numéro d'identification. Chaque sous-paquet présente un champ d'**Identification** avec ce numéro ainsi qu'un champ d'**Offset** indiquant la position relative du sous-paquet dans le paquet principal. A l'autre bout de la connexion, la machine destinataire reconstruit le paquet principal à partir des sous-paquets en fonction du nombre contenu dans leur champ d'**Identification** et du numéro présent dans leur champ d'**Offset**.

La convention d'écriture d'une adresse IP, sous une forme qui nous est lisible, consiste à utiliser une notation avec des points séparateurs de nombres décimaux comme dans `152.2.254.81`. Chaque nombre séparé par les points est un octet et, par conséquent, la gamme va de 0 à 255. Donc, l'espace d'adresses s'étend de `0.0.0.0` à `255.255.255.255`. Pour organiser l'attribution des adresses, chaque adresse à 32 bits est divisée en deux parties : une partie réseau (*network*) et une partie hôte (*host*), tel qu'indiqué dans la figure 26.1.

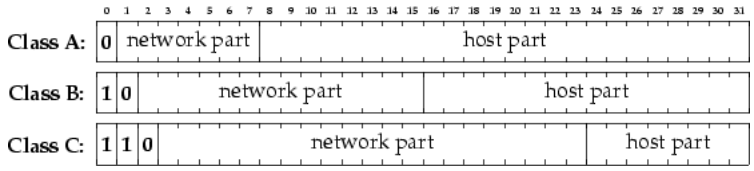


FIG. 26.1 – Types principaux de réseaux.

La partie réseau de l'adresse IP désigne le LAN tandis que la partie hôte désigne une machine donnée du LAN. Du fait qu'à l'époque de la mise en place de ces spécifications, on ne pouvait prédire s'il y aurait davantage de LANs que de machines par LAN, trois classes différentes d'adresses ont été créées.

Les adresses de *classe A* commencent avec 0 pour premier bit de la partie

réseau. Donc, un réseau de classe A présente un nombre avant le premier point de l'adresse, inférieur à 128. Les 7 bits suivants précise l'identité du LAN et les 24 autres, l'identité d'une machine physique de ce LAN. Une adresse de *classe B* débute avec 1 suivi de 0 pour les deux premiers bits. De ce fait, le premier nombre en mode décimal précédant le premier point est compris entre 128 et 191. Les 14 bits suivants donnent l'identité du LAN, et les 16 autres bits, celle de toute machine physique appartenant à ce LAN. La plupart des universités appartiennent à un réseau de classe B. Enfin, les adresses de *classe C* commencent avec la suite de bits 110. Le premier nombre en mode décimal est compris entre 192 et 223. Les 21 bits qui suivent concernent l'identité du LAN et les 8 bits restants définissent une machine physique. Le plus souvent , les petites sociétés utilisent des adresses de classes C.

En pratique, peu d'organismes nécessitent des adresses de classes A. Une université ou une entreprise de grande taille peut utiliser un réseau de classe B. Elles auront leurs propres sous-réseaux de sorte que le troisième nombre en notation décimale de l'adresse d'un département (les bits de 16 à 23) soit propre à ce dernier et que les machines de ce département soient caractérisées par le dernier champ en notation décimale (bits de 24 à 31). Dans ce cas, le LAN devient un micro-internet en soi et il est appelé "réseau". Les départements possèdent leur sous-réseau (*subnet*).

26.2 Adresses IP spéciales.

Certaines adresses IP spéciales ne sont pas utilisées sur l'Internet. Celles comprises entre `192.168.0.0` et `192.168.255.255` sont privées et susceptibles d'être utilisées dans un réseau LAN local. Elles ne communiquent pas directement avec l'internet. Les adresses allant de `127.0.0.0` à `127.255.255.255` sont employées pour des communications internes à la machine elle-même (appelée *localhost*). D'ordinaire, `127.0.0.1` est une adresse pointant vers la machine elle-même. En outre, la gamme s'étendant de `172.16.0.0` à `127.31.255.255` contient des adresses privées supplémentaires pour les très grands réseaux, et la gamme `10.0.0.0` à `10.255.255.255` est associée à de plus grands réseaux, encore.

26.3 Masques de réseau et adresses.

Considérons le cas d'une université avec un réseau de classe B. Elle pourrait avoir une gamme d'adresses allant de `137.158.0.0` à `137.158.255.255`. Supposons qu'il ait été décidé d'attribuer 512 adresses IP au département d'astronomie, de `137.158.26.0` à `137.158.27.255`. Le département d'astronomie a pour adresse réseau `137.158.26.0`. Toutes les machines de ce réseau ont pour masque réseau (*network mask*) `255.255.254.0`. Une machine de ce département pourrait avoir pour adresse IP `137.158.27.158`. Cette terminologie sera utilisée ultérieurement. La figure 26.2 illustre cet exemple.

	Dotted IP	Binary
Netmask	255 . 255 . 254 . 0	1111 1111 1111 1111 1111 1110 0000 0000
Network address	137 . 158 . 26 . 0	1000 1001 1001 1110 0001 1010 0000 0000
IP address	137 . 158 . 27 . 158	1000 1001 1001 1110 0001 1011 1001 1110
Host part	0 . 0 . 1 . 158	0000 0000 0000 0000 0000 0001 1001 1110

FIG. 26.2 – Répartition d'une adresse en portions "réseau" et "hôte".

26.4 Ordinateurs sur réseau LAN.

Dans cette section, nous utilisons le terme LAN pour désigner un réseau d'ordinateurs tous connectés par des câbles Ethernet, ce qui est très usuel dans le cas d'entreprises avec une cinquantaine de machines. Chaque machine possède sa carte Ethernet à laquelle on se réfère par `eth0` lorsqu'on utilise la ligne de commande. Quand une machine possède plusieurs cartes Ethernet, celles-ci sont nommées `eth0`, `eth1`, `eth2`, etc. Chacune de ces cartes est aussi appelées *interface réseau* (*network interface*), parfois simplement *interface* ou encore *port Ethernet* (*Ethernet port* en anglais).

Un LAN fonctionne de la manière suivante. Les cartes réseau transmettent une trame au LAN et les autres cartes réseau lisent cette dernière dans le LAN. Si une seule carte émet une trame, toutes les autres cartes peuvent lire cette dernière. Si une carte commence à transmettre une trame alors qu'une autre carte est déjà en train d'en émettre une, une *collision* (*clash*) a lieu. La carte qui a commencé à émettre en second lieu réessaye après une durée aléatoirement choisie. Chaque carte réseau a une adresse physique de 48 bits, appelée adresse matérielle, qui lui est attribuée au moment de sa fabrication dans son en-tête. C'est par elle que sont adressées les trames, si bien que les cartes réseau ignorent les trames qui ne leur sont pas adressées.

Etant donné que la transmission des trames est régie par les cartes réseau, l'adresse matérielle de destination doit être déterminée à partir de l'adresse IP de destination avant que le paquet ne soit envoyé à une machine donnée. Cet objectif est atteint grâce à l'*Address Resolution Protocol* (*ARP*). Une machine transmet un paquet spécial qui demande : "quelle est l'adresse matérielle associée à cette adresse IP ?". La machine destinataire répond et la machine transmettrice enregistre le résultat. Naturellement, si vous modifiez soudainement les cartes réseau, les autres machines du LAN recevront une information erronée. Aussi, ARP est-il un protocole basé sur des pauses et des requêtes répétées.

Essayez la commande `arp` de manière à obtenir une liste des adresses matérielles correspondant aux adresses IP.

26.5 Configuration d'interfaces.

La plupart des distributions ont une méthode générique pour configurer les interfaces. Cependant, nous allons réaliser la configuration complète d'un réseau en utilisant la ligne de commande.

D'abord, nous créons une interface `lo`, c'est-à-dire le périphérique loopback (notez qu'il n'a rien à voir avec les périphériques-blocs loopback qui sont les

fichiers /dev/loop ?). Le périphérique loopback est une carte réseau imaginaire utilisée pour faire communiquer la machine avec elle-même. Par exemple, si vous utilisez telnet sur votre machine locale, vous êtes réellement connecté *via* le périphérique loopback. La commande ifconfig (*interface configuration*) est utilisée pour effectuer des opérations de configuration des interfaces. Exécutez :

```
/sbin/ifconfig lo down
/sbin/ifconfig eth0 down
```

de manière à supprimer toutes les interfaces existantes. Ensuite, lancez :

```
/sbin/ifconfig lo 127.0.0.1
```

ce qui crée l'interface loopback.

A présent, créons une interface Ethernet :

```
/sbin/ifconfig eth0 192.168.3.0 broadcast 192.168.3.255 netmask 255.255.255.0
```

L'adresse de broadcast est spéciale car toutes les machines y répondent. Usuellement, il s'agit de la première ou de la dernière adresse d'un réseau particulier.

Poursuivons :

```
/sbin/ifconfig
```

Ceci permet d'afficher les caractéristiques des interfaces. La sortie devrait être :

```
eth0          Link encap :Ethernet HWaddr 00 :00 :E8 :3B :2D :A2
              inet addr :192.168.3.9 Bcast :192.168.3.255 Mask :255.255.255.0
              UP BROADCAST RUNNING MULTICAST MTU :1500 Metric :1
              RX packets :1359 errors :0 dropped :0 overruns :0 frame :0
              TX packets :1356 errors :0 dropped :0 overruns :0 carrier :0
              collisions :0 txqueulen :100
              Interrupt :11 Base address :0xe400

lo            Link encap :Local Loopback
              inet addr :127.0.0.1 Mask 255.0.0.0
              UP LOOKBACK RUNNING MTU :3924 Metric :1
              RX packets :53175 errors :0 dropped :0 overruns :0 frame :0
              TX packets :53175 errors :0 dropped :0 overruns :0 carrier :0
              collisions :0 txqueuelen :0
```

Cet affichage révèle diverses informations intéressantes comme, par exemple, l'adresse matérielle à 48-bits de la carte réseau (représentation en octets hex : 00 :00 :E8 :3B :2D :A2).

26.6 Configuration du routage.

Désormais, les interfaces sont actives. Cependant, rien n'indique encore au noyau la (ou les) destination(s) de tel ou tel paquet, même si nous pourrions nous attendre à un tel comportement à priori. Avec UNIX, il faut indiquer explicitement au noyau d'envoyer tel paquet particulier vers telle interface particulière.

Tout paquet arrivant à une interface est versé dans un "bassin de traitement" (*pool*) par le noyau. Le noyau analyse alors chaque adresse de destination du paquet et décide, en fonction de la destination, où le paquet est envoyé. La provenance du paquet n'a pas d'importance. Une fois le noyau en possession du paquet, c'est la destination qui compte. Au premier chef, il incombe au réseau d'assurer que le paquet n'arrive pas à une mauvaise interface.

Nous savons que tout paquet destiné à l'adresse réseau `127.???.???.???` doit être dirigé vers l'interface loopback (ceci est une convention en quelque sorte). La commande

```
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo
```

ajoute une *route* au réseau `127.0.0.0`, quoique celle-ci soit virtuelle.
Le périphérique `eth0` peut être routé de la manière suivante :

```
/sbin/route add -net 192.168.3.0 netmask 255.255.255.0 eth0
```

La commande permettant de visualiser la table de routage est :

```
/sbin/route -n
```

L'option `-n` agit sur `route` afin de ne pas afficher les adresses IP sous forme de nom d'hôte, ce qui donne :

```
Kernel IP routing table
Destination  Gateway    Genmask         Flags  Metric  Ref    Use    Iface
127.0.0.0    0.0.0.0    255.0.0.0       U      0       0      0      lo
192.168.3.0  0.0.0.0    255.255.255.0   U      0       0      0      eth0
```

La table de routage indique que les paquets dont l'adresse de destination est `127.0.0.0/255.0.0.0` [la notation *network/mask* est souvent utilisée pour désigner les gammes d'adresses IP] doivent être envoyés au périphérique `lo`opback. De plus, les paquets dont l'adresse de destination est `192.168.3.0/255.255.255.0` doivent être envoyés à `eth0`. L'adresse de la passerelle ou `Gateway` est zéro, ce qui veut dire qu'elle n'est pas configurée (voir les commandes qui suivent).

La table de routage permet donc d'effectuer le routage des paquets vers `127.` et `192.168.3`. A présent, nous devons avoir une route pour les autres adresses IP éventuelles. UNIX peut admettre une route permettant d'envoyer des paquets avec des adresses de destination IP particulières, vers une autre machine du LAN. C'est à partir de cette machine (parfois appelée *passerelle* ou *gateway*) que ces paquets pourront être envoyés ailleurs. La commande établissant ce routage est :

```
/sbin/route add -net <adresse-reseau> netmask <masque-reseau> gw \
<adresse-IP-passerelle> <interface-passerelle-vers-LAN>
```

Cette commande est la plus générale, mais usuellement il est plus facile d'utiliser :

```
/sbin/route add default gw <adresse-reseau> netmask <adresse-IP-passerelle> \
<interface-passerelle-vers-LAN>
```

lorsque nous voulons ajouter une route qui s'applique à tout autre paquet que ceux définis plus haut (par exemple). Cette route est appelée la *passerelle par défaut* (*default gateway*). `default` signifie "tous les paquets" ; cela équivaut à :

```
/sbin/route add -net 0.0.0.0 netmask 0.0.0.0 gw <adresse-IP-passerelle> \
<interface-passerelle-vers-LAN>
```

mais, étant donné que les routes sont établies en fonction du `netmask`, *les routes les plus spécifiques sont utilisées préférentiellement à celles qui le sont moins.*

Finalement, vous pouvez fixer le nom d'hôte (le nom de votre machine) ainsi :

```
hostname cericon.cranzgot.co.za
```

Les commandes que nous venons d'utiliser sont résumées ici :

```
/sbin/ifconfig lo down
/sbin/ifconfig eth0 down
/sbin/ifconfig lo 127.0.0.1
/sbin/ifconfig eth0 192.168.3.9 broadcast 192.168.3.255 netmask 255.255.255.0
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo
/sbin/route add -net 192.168.3.0 netmask 255.255.255.0 eth0
/sbin/route add default gw 192.168.3.254 eth0
hostname cericon.cranzgot.co.za
```

Bien que ces 7 commandes vous permettent de faire une configuration manuelle complète, la configuration de scripts de démarrage vous évite cette procédure fastidieuse (voir la section 26.7).

26.7 Configuration de scripts de démarrage.

Pour la plupart, les distributions possèdent un système plus ou moins évolué de scripts modulaires qui initialisent le réseau.

26.7.1 Scripts réseau de RedHat.

Les systèmes RedHat contiennent un répertoire `/etc/sysconfig` incluant les fichiers de configuration pour initialiser automatiquement le réseau.

Le fichier `/etc/sysconfig/network-scripts/ifcfg-eth0` présente les lignes :

```
DEVICE=eth0
IPADDR=192.168.3.9
NETMASK=255.255.255.0

NETWORK=192.168.3.0
BROADCAST=192.168.3.255
ONBOOT=yes
```

Le fichier /etc/sysconfig/network contient :

```
NETWORKING=yes
HOSTNAME=cericon.cranzgot.co.za
GATEWAY=192.168.3.254
```

Ces deux fichiers sont équivalents à ce que réalise la configuration manuelle donnée à la section 26.6. Ces deux fichiers acceptent un nombre considérable d'options relatives à divers protocoles autres que IP, mais ce que nous venons de voir constitue la configuration la plus courante.

Le fichier /etc/sysconfig/network-scripts/ifcfg-lo associé au périphérique loopback est automatiquement configuré à l'installation. Vous ne devriez pas le modifier.

Les commandes de démarrage ou d'arrêt du réseau (celles entre parenthèses sont des alternatives) sont respectivement :

```
/etc/init.d/network start
( /etc/rc.d/init.d/network start )
/etc/init.d/network stop
( /etc/rc.d/init.d/network stop )
```

Ces commandes lisent indirectement vos fichiers /etc/sysconfig.

Il vous est loisible d'ajouter des fichiers supplémentaires, par exemple, ifcfg-eth1 dans /etc/sysconfig/network-scripts pour une seconde carte Ethernet. Ainsi, ifcfg-eth1 pourrait contenir :

```
DEVICE=eth1
IPADDR=192.168.4.1
NETMASK=255.255.255.0
NETWORK=192.168.4.0
BROADCAST=192.168.4.255
ONBOOT=yes
```

De manière à ce que le noyau puisse envoyer les paquets entre ces deux cartes, exécutez :

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

26.7.2 Scripts réseau de Debian.

[Selon son habitude, Debian adopte une méthode plus nette]. Debian possède un répertoire /etc/network contenant un fichier /etc/network/-interfaces (voir interfaces(5)). Si on veut obtenir une configuration identique à celle de RedHat (ci-dessus), le fichier /etc/network de Debian contiendra :

```
iface lo inet loopback
iface eth0 inet static
   address 192.168.3.9
   netmask 255.255.255.0
   gateway 192.168.3.254
```

Le fichier /etc/network/options contiendra les options de transmission suivantes (il en existe d'autres) :

```
ip_forward=no
spoofprotect=yes
syncookies=no
```

Pour arrêter et redémarrer le réseau (c'est-à-dire activer les interfaces et le routage), exécutez :

```
/etc/init.d/networking stop
/etc/init.d/networking stop
```

A nouveau, ces commandes lisent indirectement votre fichier /etc/network/interfaces.

En réalité, le script /etc/init.d/networking exécute simplement les commandes ifup et ifdown. Voir ifup(8). Vous pouvez utiliser ces commandes directement pour obtenir un contrôle plus fin.

Pour ajouter une interface supplémentaire (et obtenir le même résultat que dans l'exemple relatif à RedHat ci-dessus), il faut éditer /etc/network/interfaces. La méthode Debian est :

```
iface lo inet loopback
iface eth0 inet static
   address 192.168.3.9
   netmask 255.255.255.0
   gateway 192.168.3.254
iface eth1 inet static
   address 192.168.4.1
   netmask 255.255.255.0
```

Ensuite, il faut éditer /etc/network/options pour y écrire ip_forward=yes.

Finalement, alors que RedHat permet de retenir le nom d'hôte dans la ligne HOSTNAME=... du fichier /etc/sysconfig/network, Debian présente un fichier /etc/hostname qui, dans le cas présent, ne contiendra que :

```
cericon.cranzgot.co.za
```

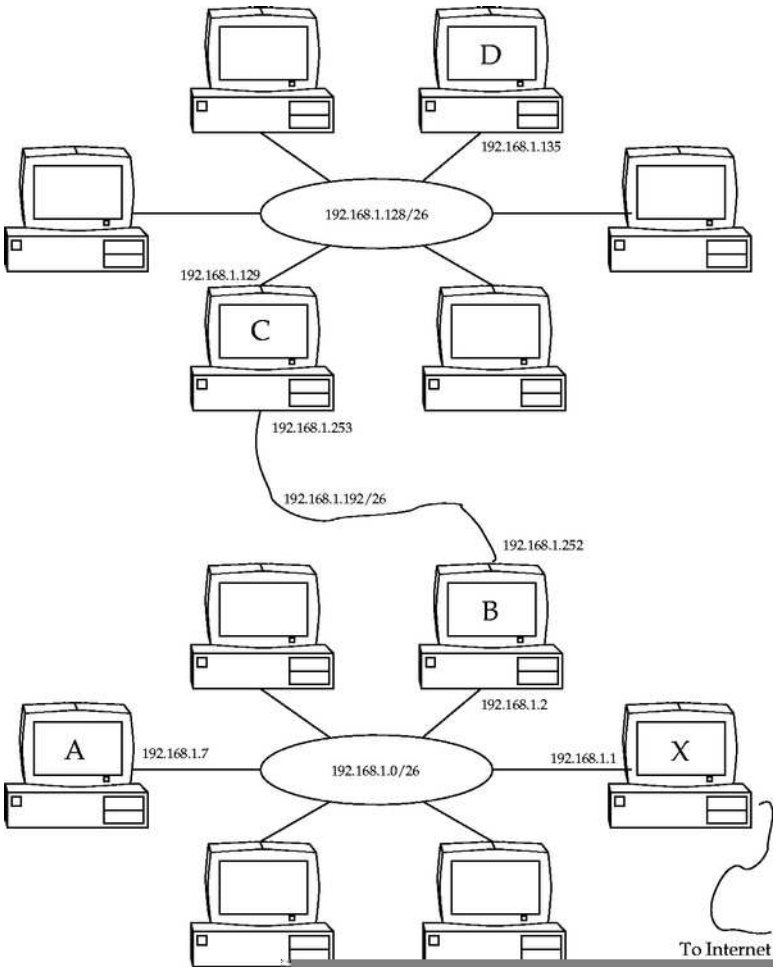


FIG. 26.3 – Cas de deux réseaux interconnectés (B–C) et communiquant avec l'internet par X.

26.8 Routage complexe – Exemple "à sauts multiples".

Considérons deux réseaux LAN qui doivent communiquer entre eux. Deux machines (une dans chaque LAN) jouent le rôle de relai. En l'occurence, elles sont connectées par une ligne série permanente (voir la figure 26.3).
Ce montage peut être réduit à cinq machines "clés" **X**, **A**, **B**, **C** et **D**. Les machines **X**, **A** et **B** forment le réseau **LAN 1** `192.168.1.0/26`. Les machines **C** et **D** constituent le réseau **LAN 2** `192.168.1.128/26`. Remarquez qu'en attribuant 26 bits sur les 32 associés à l'adressage (c'est la notation "`/26`" que nous venons d'utiliser), on fait de **LAN 1** et de **LAN 2** des réseaux de classe C. Les 6 autres bits sont utilisés pour les adresses IP des machines. Ceci implique donc qu'il ne peut y avoir au plus que $2^6 = 64$ adresses IP pour chaque LAN. Le lien série de communication entre **LAN 1** et **LAN 2** doit être établi entre

les machines **B** et **C**.

La machine **X** a pour adresse IP (dans **LAN 1**) : 192.168.1.1. Cette machine est une passerelle vers l'internet. Le port Ethernet de la machine **B** est simplement configuré avec une adresse IP 192.168.1.2 et la passerelle par défaut (pour pouvoir sortir vers l'internet via **X**) est 192.168.1.1. Observez que l'adresse de broadcast est 192.168.1.63 (les 6 derniers bits valent 1).

Le port Ethernet de la machine **C** (**LAN 2**) est configuré avec une adresse IP égale à 192.168.1.129. Aucune passerelle par défaut ne devrait être déclarée avant que la connexion série ne soit configurée.

Nous devons établir le réseau entre **B** et le sous-réseau 192.168.1.128/26 via la machine **C**. Il s'agit d'un sous-réseau en soi, même s'il n'y a que deux machines interconnectées. Nous allons attribuer les adresses 192.168.1.252 à la machine **B** et 192.168.1.253 à la machine **C** pour ce qui concerne les ports se faisant face. Ceci constitue un exemple réel avec un lien série non fiable. Sont requis, pppd[1] pour conserver le lien actif et, un script shell pour redémarrer le lien s'il disparaissait. Le script sur la machine **B** sera :

```
#/!bin/sh
while true;do
   pppd lock local mru 296 mtu 296 nodetach nocrtscts nocdtrcts \
   192.168.1.252 :192.168.1.253 /dev/ttyS0 115200 noauth \
   lcp-echo-interval 1 lcp-echo-failure 2 lcp-max-terminate 1 lcp-restart 1
done
```

Notez que si le lien était de type Ethernet (sur une seconde carte Ethernet de la machine **B**) et que nous avions à faire à un véritable LAN entre les machines **B** et **C** (avec un sous-réseau 192.168.1.252/26), le script équivalent serait :

```
/sbin/ifconfig eth1 192.168.1.252 broadcast 192.168.1.255 netmask 255.255.255.192
```

Dans ce cas, tous les ppp0 seraient changés en eth1 dans les scripts qui suivent.

Le routage sur la machine **B** est réalisé avec le script suivant, pourvu que le lien soit activé. Le script doit être exécuté chaque fois que pppd vient de négocier la connexion et il peut donc être placé dans le fichier /etc/pppd/ip-up. Ce script sera exécuté par pppd automatiquement dès que l'interface ppp0 est disponible :

```
/sbin/route del default
/sbin/route add -net 192.168.1.192 netmask 255.255.255.192 dev ppp0
/sbin/route add -net 192.168.1.128 netmask 255.255.255.192 gw 192.168.1.253
/sbin/route add default gw 192.168.1.1

echo 1 > /proc/sys/ipv4/ip_forward
```

Notre table de routage complète et la liste des interfaces pour la machine **B** ressemblent à ceci [RedHat 6 préfère ajouter des routes explicites

[1] NdT : le démon pppd est associé au *Point-to-Point Protocol* qui fournit une méthode de transmission des datagrammes via des liens séries en mode point-à-point. Voir pppd(8).

(redondantes) pour chaque périphérique, ce qui n'est pas nécessaire sur votre système] :

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   Metric  Ref    Use  Iface
192.168.1.2     0.0.0.0         255.255.255.255 UH      0       0      0    eth0
192.168.1.253   Gateway         Genmask         Flags   Metric  Ref    Use  Iface
192.168.1.0     0.0.0.0         255.255.255.255 UH      0       0      0    eth0
192.168.1.192   0.0.0.0         255.255.255.255 UH      0       0      0    ppp0
192.168.1.128   0.0.0.0         255.255.255.192 U       0       0      0    eth0
127.0.0.0       0.0.0.0         255.255.255.192 U       0       0      0    ppp0
0.0.0.0         192.168.1.253   255.255.255.192 UG      0       0      0    ppp0
                0.0.0.0         255.0.0.0       U       0       0      0    lo
                192.168.1.1     0.0.0.0         UG      0       0      0    eth0



eth0    Link  encap :Ethernet  HWaddr  00 :A0 :24 :75 :3B :69
        inet addr :192.168.1.2 Bcast :192.168.1.63 Mask :255.255.255.192
lo      Link encap :Local Loopback
        inet addr :127.0.0.1 Mask :255.0.0.0
ppp0    Link encap :Point-to-Point Protocol
        inet addr :192.168.1.252 P-t-P :192.168.1.253 Mask :255.255.255.255
```

Sur la machine C, nous pouvons exécuter le script suivant :

```
#/!bin/sh
while true; do
   pppd lock local mru 296 mtu 296 nodetach nocrtscts nocdtrcts \
   192.168.1.253 :192.168.1.252 /dev/ttyS0 115200 noauth \
   lcp-echo-interval 1 lcp-echo-failure 2 lcp-max-terminate 1 lcp-restart 1
done
```

et, à présent, créer les routes avec les commandes :

```
/sbin/route del default
/sbin/route add -net 192.168.1.192 netmask 255.255.255.192 dev ppp0
/sbin/route add default gw 192.168.1.252

echo 1 > /proc/sys/net/ipv4/ip_forward
```

La table de routage complète pour la machine C devrait ressembler à ceci :

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   Metric  Ref    Use  Iface
192.168.1.129   0.0.0.0         255.255.255.255 UH      0       0      0    eth0
192.168.1.252   0.0.0.0         255.255.255.255 UH      0       0      0    ppp0
192.168.1.192   0.0.0.0         255.255.255.192 U       0       0      0    ppp0
192.168.1.128   0.0.0.0         255.255.255.192 U       0       0      0    eth0
```

```
127.0.0.0       0.0.0.0         255.0.0.0       U         0          0          0          lo
0.0.0.0         192.168.1.252   0.0.0.0         UG        0          0          0          ppp0

eth0    Link encap :Ethernet HWaddr 00 :A0 :CC :D5 :D8 :A7
        inet addr :192.168.1.129 Bcast :192.168.1.191 Mask :255.255.255.192
lo      Link encap :Local Loopback
        inet addr :127.0.0.1 Mask :255.0.0.0
ppp0    Link encap :Point-to-Point Protocol
        inet addr :192.168.1.253 P-t-P :192.168.1.252 Mask :255.255.255.255
```

La machine **D** peut être configurée comme toute machine ordinaire d'un LAN. Il faut lui indiquer que la passerelle par défaut est `192.168.1.129`. En revanche, la machine **A** doit être capable d'envoyer des paquets destinés au sous-réseau `192.168.1.128/26` via la machine **B**. Sa table de routage doit donc présenter une ligne pour le LAN `192.168.1.128/26`. La table de routage complète de la machine **A** est :

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags     Metric     Ref        Use        Iface
192.168.1.0     0.0.0.0         255.255.255.192 U         0          0          0          eth0
192.168.1.128   102.168.1.2     255.255.255.192 UG        0          0          0          eth0
127.0.0         0.0.0.0         255.255.0.0     U         0          0          0          lo
0.0.0.0         192.168.1.1     0.0.0.0         UG        0          0          0          eth0
```

De manière à éviter de devoir ajouter cette route supplémentaire sur la machine **A**, vous pouvez décrire la même route sur la machine **X**. Ceci peut sembler curieux, mais cela signifie que tous les paquets provenant de **A** et destinés au réseau **LAN 2** doivent d'abord essayer de passer par **X** (vu qu'**A** ne possède qu'une route), et ensuite, ils seront redirigés *par* **X** pour passer via **B**.

La configuration qui précède permet aux machines d'envoyer correctement les paquets entre les machines A et D mais également sur l'internet. Un avertissement toutefois : parfois la commande `ping` (voir la sous-section 26.10.1) ne fonctionne pas alors que `telnet` est opérationnel. Ceci peut être une particularité de certaines versions du noyau que nous utilisons, ***shrug*** [sic].

26.9 *Aliasing* d'interfaces - Multiples IPs sur une seule carte physique.

Le fichier `/usr/src/linux/Documentation/networking/alias.txt` contient la documentation du noyau à sujet.

Vous pouvez très bien utiliser une carte réseau avec plusieurs adresses IP différentes. Appelez l'interface `eth0 :`*n* où *n* désigne un nombre entier positif ou nul. Vous pouvez utiliser `ifconfig` autant de fois que vous voulez sur la même carte réseau :

```
/sbin/ifconfig eth0 :0 192.168.4.1 broadcast 192.168.4.255 netmask 255.255.255.0
/sbin/ifconfig eth0 :1 192.168.5.1 broadcast 192.168.5.255 netmask 255.255.255.0
/sbin/ifconfig eth0 :2 192.168.6.1 broadcast 192.168.6.255 netmask 255.255.255.0
```

et cela, en supplément de votre périphérique ordinaire `eth0`. En l'occurence, la même interface peut communiquer avec trois réseaux LAN `192.168.4.0`, `192.168.5.0` et `192.168.6.0`. N'oubliez pas de définir le routage pour ces réseaux.

26.10 Utilitaires de diagnostic.

Il est essentiel de pouvoir inspecter et tester votre réseau pour résoudre des problèmes. Nous décrirons donc les utilitaires UNIX standard.

26.10.1 `ping`.

La commande `ping` est l'utilitaire réseau le plus répandu. Les paquets IP véhiculés sur l'Internet sont de trois types : *TCP*, *UDP* ou *ICMP*. Les deux premiers –discutés au chapitre 27– constituent deux moyens fondamentaux de communications entre deux programmes en cours d'exécution sur deux machines. *ICMP* est l'acronyme d'*Internet Control Message Protocol*. Il s'agit d'un protocole permettant de gérer les informations relatives aux erreurs sur des machines en connexion.[2]

Essayez :

```
ping metalab.unc.edu
```

ou spécifiez n'importe quel hôte bien connu. Vous devriez obtenir le résultat suivant :

```
PING metalab.unc.edu (152.19.254.81)from 192.168.3.9 : 56(84) bytes of data
64 bytes from 152.19.254.81 :icmp_seq=0 ttl=238 time=1059.1 ms
64 bytes from 152.19.254.81 :icmp_seq=1 ttl=238 time=764.9 ms
64 bytes from 152.19.254.81 :icmp_seq=2 ttl=238 time=858.8 ms
64 bytes from 152.19.254.81 :icmp_seq=3 ttl=238 time=1179.9 ms
64 bytes from 152.19.254.81 :icmp_seq=4 ttl=238 time=986.6 ms
64 bytes from 152.19.254.81 :icmp_seq=5 ttl=238 time=1274.3 ms
64 bytes from 152.19.254.81 :icmp_seq=6 ttl=238 time=930.7 ms
```

Dans les faits, `ping` envoie des paquets ICMP à `metalab.unc.edu`, qui répond automatiquement avec un paquet ICMP de retour. "Pinguer" est souvent le test-clé pour déterminer si une interface réseau est correctement configurée et fonctionnelle. Notez que certains sites filtrent les paquets ICMP : par exemple, `ping cnn.com` ne fonctionnera pas.

[2] NdT : en fait, le protocole IP effectue peu de contrôles et ne permet pas de corriger des erreurs associées au transport de données. En revanche, il fait part de ces erreurs aux couches réseau voisines. ICMP est utilisé par les routeurs pour signaler une erreur (*Delivery Problem*). ICMP peut aussi être sujet aux erreurs. Pour éviter l'effet boule de neige, il n'y a pas de rapport lorsqu'ICMP commet lui-même une erreur.

`ping` envoie un paquet toutes les secondes et mesure le temps qu'il faut pour recevoir le paquet de retour –exactement comme le "ping" du sonar d'un sous-marin. Sur l'internet, il est possible d'obtenir des durées excédant les 2 secondes si le site sondé est très éloigné. Sur un réseau local LAN, cette durée peut être inférieure à la milliseconde.

Si `ping` ne parvient même pas à produire la ligne `PING metalab.unc.edu...`, cela signifie que `ping` ne peut pas résoudre le nom d'hôte. Vous devrez alors vérifier que votre DNS est fonctionnel (voir le chapitre 28). Si `ping` émet cette ligne mais rien d'autre, les paquets ne sont pas émis ou retournés. Dans tous les autres cas, `ping` engendrera un message d'erreur signalant l'absence de routes ou d'interfaces.

26.10.2 `traceroute`.

`traceroute` est un utilitaire étonnant qui indique la route empruntée par un paquet pour atteindre sa destination. Cette commande utilise des paquets UDP ou, lorsque l'option `-I` est utilisée, des paquets ICMP pour déterminer l'itinéraire suivi. Par exemple de `cericon.cranzgot.co.za` à `metalab.unc.edu`,

```
traceroute metalab.unc.edu
```

donne :

```
traceroute to metalab.unc.edu (152.19.254.81), 30 hops max, 38 byte packets
1  192.168.3.254 (192.168.3.254) 1.197 ms 1.085 ms 1.050 ms
2  192.168.254.5 (192.168.254.5) 45.165 ms 45.314 ms 45.164 ms
3  cranzgate (192.168.2.254) 48.205 ms 48.170 ms 48.074 ms
4  cranzposix (160.124.182.254) 46.117 ms 46.064 ms 45.999 ms
5  cismpjhb.posix.co.za (160.124.255.193) 451.886 ms 71.549 ms 173.321 ms
6  cisapl.posic.co.za (160.124.112.1) 274.834 ms 147.251 ms 400.654 ms
7  saix.posix.co.za (160.124.255.6) 187.402ms 325.030 ms 628.576 ms
8  ndf-core1.gt.saix.net (196.25.253.1) 252.558 ms 186.256 ms 255.805 ms
9  ny-core.saix.net (196.25.0.238) 497.273 ms 454.531 ms 639.795 ms
10 bordercore6-serial5-0-0-26.WestOrange.cw.net (166.48.144.105) 595.755 ms 595.174 ms *
11 corerouter1.WestOrange.cw.net (204.70.9.138) 490.845 ms 689.483 ms 1029.369 ms
12 core6.Washington.cw.net (204.70.4.113) 580.971 ms 893.481 ms 730.608 ms
13 204.70.10.182 (204.70.10.182) 644.070 ms 726.363 ms 639.942 ms
14 mae-brdr-01.inet.qwest.net (205.171.4.201) 767.783 ms * *
15 * * *
16 * wdc-core-03.inet.qwest.net (205.171.24.69) 779.546 ms 898.371 ms
17 atl-core-02.inet.qwest.net (205.171.5.243) 894.553 ms 689.472 ms
18 atl-edge-05.inet.qwest.net (205.171.21.54) 735.810 ms 784.461 ms 789.592 ms
19 * * *
20 * * unc-gw.ncren.net (128.109.190.2) 889.257 ms
21 unc-gw.ncren.net (128.109.190.2) 646.569 ms 780.000 ms *
22 * helios.oit.unc.edu (152.2.22.3) 600.558 ms 839.135 ms
```

Vous constatez qu'il y a vingt-deux machines (et par conséquent 22 sauts ou *hops*) entre cericon.cranzgot.co.za et `metalab.unc.edu` [cette observation constitue un bon argument pour expliquer pourquoi les serveurs web "*enterprise*" ne

sont pas utilisés sur les marchés non-américains : il n'y a pas assez de vitesse réseau pour charger de tels serveurs, ce qui rend superflue toute comparaison de vitesses de serveurs].

26.10.3 `tcpdump`.

`tcpdump` scrute tout le trafic passant sur une interface donnée, c'est-à-dire tout le trafic de toutes les machines connectées sur le même concentrateur (*hub*) (aussi appelé *segment réseau* ou *segment*). De manière usuelle, une carte réseau saisit les trames qui lui sont destinées. Cependant, `tcpdump` bascule la carte en mode promiscueux (*promiscuous mode*), ce qui veut dire que la carte recherche toutes les trames indépendamment de leur adresse matérielle de destination. Essayez :

```
tcpdump -n -N -f -i eth0
```

La commande `tcpdump` sera discutée à la section 42.6. Le déchiffrement de la sortie de `tcpdump` est laissé à titre d'exercice. Nous verrons davantage de détails sur la partie *tcp* de `tcpdump` au chapitre 27.

Chapitre 27

TCP et UDP.

Au cours du chapitre précédent, nous avons abordé la communication entre machines en nous intéressant à la couche IP. Ce protocole effectue peu de contrôles et ne permet pas de corriger des erreurs associées au transport de données. En particulier, il ne vérifie pas le bon acheminement des paquets. Aussi, lorsque deux applications fonctionnent de part et d'autre de l'Océan Atlantique (par exemple), le fait d'être capable d'envoyer un paquet qui peut (ou peut ne pas) atteindre le côté opposé ne suffit-il pas. Nous avons besoin de communications *fiables* donc d'un protocole supplémentaire.

Idéalement, un programme utilisateur établit un lien sur une machine distante et lui transmet des octets tout en étant sûr que ceux-ci atteignent leur destination "point-à-point" (et ce, de part et d'autre de la ligne de communication). Une telle opération est appelée *communication à flux fiable*.

Vous pourriez imaginer envoyer des paquets séparés et attendre que pour chaque paquet, le machine distante confirme leur réception. Cette approche est inefficace car les paquets mettent un temps assez long pour atteindre leur destination ; il en va de même pour la confirmation de la réception associée à chaque paquet. L'idée est plutôt d'envoyer autant de paquets que possible en une fois et de disposer d'un moyen de négocier avec la machine distante de manière à renvoyer les paquets qui n'auraient pas été reçus. C'est précisément ce que fait *TCP* (*Transmission Control Protocol*) en envoyant des paquets de données en un coup et en analysant les *paquets de contrôle* (*acknowledgment packets*) de sorte à indiquer les parties du flux de données qui ont été correctement reçues sur la machine de destination.

Nous voyons donc que TCP est mis en oeuvre à un niveau supérieur à IP. Ainsi, la communication sur l'internet est-elle souvent appelée "communication par TCP/IP".

La communication TCP se fait en trois étapes : la *négociation*, le *transfert* et la *déconnexion* (*detachment*) [Ceci est la terminologie de l'auteur du texte original en anglais. Il s'agit d'une représentation quelque peu schématique].

Négociation L'application cliente (disons votre navigateur) initie tout d'abord la connexion en utilisant la fonction **C** `connect()` (voir `connect`(2)). Ceci force le noyau à envoyer un paquet *SYN* (de *SYN*chronisation) au serveur TCP distant (dans le cas présent, un serveur web). Le serveur web répond en envoyant un paquet *SYN-ACK* (*ACK*nowlegde : pour *accusé de ré-*

ception). Finalement, le client répond avec un dernier paquet SYN. Cette négociation se fait de manière transparente pour l'utilisateur.

Transfert L'application emploie les appels de fonction **C** `send ()` et `recv ()` (voir `send`(2) et `recv`(2)) pour envoyer et recevoir un flux de données réelles. Le flux est morcelé en paquets et ceux-ci sont envoyés individuellement à l'application distante. Dans le cas du serveur web, les premiers octets envoyés devraient être `GET /index.html HTTP/1.0<CR><NL><CR><NL>`. De l'autre côté de la communication, les paquets de réponse (appelés *ACK*) sont retournés aussitôt que les données arrivent, ce qui permet d'indiquer quelles parties du flux manquent et de demander une nouvelle transmission. La communication se passe dans les deux sens en *full-duplex* (le flux se produit simultanément dans les deux directions, comme dans le cas des communications téléphoniques) : les données et les paquets ACK se croisent.

Déconnexion L'application utilise les fonctions **C** `shutdown ()` et `close ()` (voir `shutdown ()` et `close(2)`) afin de clore la connexion. Un paquet *FIN* est transmis et la communication TCP cesse.

27.1 L'en-tête TCP.

Les paquets TCP sont *encapsulés* dans les paquets IP. Ils se trouvent dans la partie **Data begins at ...** du paquet IP. Un paquet TCP se compose d'un en-tête et d'une partie contenant les données. Cette dernière peut parfois être vide; c'est le cas durant la phase de négociation. Le tableau 27.1 répertorie l'en-tête complet TCP/IP.

Au minimum, l'en-tête TCP/IP combiné représente 40 octets.

Avec les machines fonctionnant sur l'internet, diverses applications communiquent souvent simultanément. Les champs intitulés **Source Port** et **Destination Port** permettent d'identifier et de distinguer chaque flux individuellement. Dans le cas d'une communication web, le port de destination (du point de vue des programmes clients) est le port 80. Donc, tout trafic sortant sera caractérisé par le nombre 80 dans ce champ. Le port source (du point de vue des programmes clients) est choisi au hasard parmi les ports inutilisés de nombre supérieur à 1024 avant que toute connexion ne soit négociée. Le numéro de port est aussi indiqué dans les paquets sortant. Deux flux ne peuvent avoir la même combinaison de numéros de port source et port destination. Le noyau utilise les numéros de port contenus dans les paquets entrant pour déterminer quelle application a requis tel ou tel paquet. La même méthode est suivie sur la machine distante.

Le numéro de séquence (**Sequence Number**) est le décalage de position qui caractérise un paquet particulier dans le flux de données. Le numéro d'accusé de réception (**Acknowledge Number**) est le point jusqu'où les données ont été reçues. Le descripteur **Control** consiste en divers bits indicateurs. Le descripteur **Window** indique la quantité maximale de données que le programme de réception est préparé à accueillir. La somme de contrôle (**Checksum**) permet de vérifier l'intégrité des données, et le pointeur de données urgentes (**Urgent pointer**) est utilisé pour interrompre le flux. Les données requises par les extensions du protocole sont ajoutées en tant qu'options à la suite de l'en-tête.

TAB. 27.1 – En-têtes associées à IP et à TCP.

Octets (IP)	**Description**
0	Bits 0-3 : Version, bits 4-7 : Internet Header Length (IHL)
1	Type de Service (TOS)
2-3	Longueur
4-5	Identification
6-7	Bits 0-3 : Indicateurs (*flags*), bits 4-15 : décalage ou *offset*
8	Temps de vie (*Time To Live* ou TTL)
9	Type
10-11	Somme de contrôle (*checksum*)
12-15	Adresses IP source
16-19	Adresses IP de destination
20-IHL*4-1	Options + remplissage pour grouper jusqu'à 4 octets
Octets (TCP)	**Description**
0-1	Port source
2-3	Port destination
4-7	Numéro de séquence
8-11	Numéro d'accusé de réception
12	Bits 0-3 :nombres d'octets des options supplémentaires de TCP / 4
13	Contrôle
14-15	Fenêtre
16-17	Somme de contrôle (*checksum*)
18-19	Pointeur de données urgentes (*urgent pointer*)
20-(20 +options * 4)	Options + remplissage pour grouper jusqu'à 4 octets
Les données TCP commencent à IHL * 4 + 20 + options * 4 et finissent à Lentgh -1	

27.2 Un exemple de session TCP.

Il est aisé de visualiser la manière dont TCP fonctionne en utilisant `telnet`. Vous avez certainement l'habitude d'utiliser `telnet` pour vous connecter à des systèmes distants. En réalité, `telnet` est un programme générique pour établir toute "connexion" (ou *socket*) TCP. En l'occurrence, nous allons tenter une connexion à la page web de `cnn.com`.

Nous devons d'abord obtenir l'adresse IP de `cnn.com` : [1]

```
[root@cericon]# host cnn.com
cnn.com has address 207.25.71.20
```

A présent, nous pouvons exécuter dans une autre fenêtre :

[1] NdT : il se peut que la commande soit `hostx cnn.com`; ce qui retournera `cnn.com A adresse_IP_decimale`.

```
[root@cericon]# tcpdump \
'( src 192.168.3.9 and dst 207.25.72.20 ) or ( src 207.25.71.20 and dst 192.168.3.9 )'
Kernel filter, protocol ALL, datagram packet socket
tcpdump :listening on all devices
```

ce qui indique à la machine d'afficher tous les paquets ayant pour sources (`src`) et destinations (`dst`) notre adresse et celle de CNN.

A présent, nous pouvons utiliser le protocole HTTP pour capter la page principale. Tapez la commande HTTP `GET /HTTP/1.0` et pressez deux fois la touche Enter (comme cela est requis dans le protocole HTTP). Les quelques premières et dernières lignes sont données ci-dessous :

```
[root@cericon]# telnet 207.25.71.20 80
Trying 207.25.71.20...
Connected to 207.25.71.20.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 200 OK
Server : Netscape-Enterprise/2.01
Date : Tue, 18 Apr 2000 10 :55 :14 GMT
Set-cookie :CNNid=cf19472c-23286-956055314-2 ; expires=Wednesday, 30-Dec-2037 16 :00 :00
                                        GMT ;path=/ ; domain=.cnn.com

Last-modified : Tue, 18 Apr 2000 10 :55 :14 GMT
Content-type : text/html

<HTML>
<HEAD>
       <TITLE>CNN.com</TITLE>
       <META http-equiv="REFRESH" conntent="1800">

       <!--CSSDATA :956055234-->
       <SCRIPT src="/virtual/2000/code/main.js" language="javascript"></SCRIPT>
       <LINK rel="stylesheet" href="/virtual/2000/style/main.css" type="text/css">
       <SCRIPT language="javascript" type="text/javascript">
              <!--//
              if ((navigator.platform=='MacPPC')&&(navigator.ap

..............
..............

</BOBY>
</HTML>
connexion closed by foreign host.
```

Les commandes décrites ci-dessus affichent la première page du site web de CNN en format HTML brut. Il est facile de la copier dans un fichier puis de la consulter hors-ligne.

Dans l'autre fenêtre, `tcpdump` est en train de nous montrer quels paquets

sont échangés. `tcpdump` facilite la lecture en affichant les noms d'hôte au lieu des adresses IP et aussi les lettres `www` plutôt que le numéro de port 80. Le port choisi aléatoirement (au-delà de 1024) est en l'occurrence `4064`.

```
[root@cericon]# tcpdump \
'(src 192.168.3.9 and dst 207.25.71.20) or (src 207.25.71.20 and dst 192.168.1.3)'
Kernel filter, protocol ALL, datagram packet socket
tcpdump : listening on all devices
12.52.35.467121 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   S 2463192134 :2463192134(0) win 32120 <mss 1460,sackOK,timestamp 154031689 0,nop,wscale 0
12.52.35.964703 eth0 < www1.cnn.com> cericon.cranzgot.co.za.4064 :
   S 4182178234 :4182178234(0) ack 2463192135 win 10136 <nop,nop,timestamp 1075172823 154031
12 :52 :35.964791 eth0 > cericon.cranzgot.co.za4064 > www1.cnn.com.www :
   . 1 :1(0) ack 1 win 32120 <nop,nop,timestamp 154031739 1075172823> (DF)
12 :52 :46.413043 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   P 1 :17(16) ack 1 win 32120 <nop,nop,timestamp 154032784 1075172823> (DF)
12 :52 :46.908156 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 1 :1(0) ack 17 win 10136 <nop,nop,timestamp 1075173916 154032784>
12 :52 :49.259870 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   P 17 :39(2) ack 1 win 32120 <nop,nop,timestamp 154033068 1075173916> (DF)
12 :52 :49.886846 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   P 1 :278(77) ack 19 win 10136 <nop,nop,timestamp 1075174200 154033068>
12 :52 :49.887039 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   . 19 :19(0) ack 278 win 31856 <nop,nop,timestamp 154033131 1075174200> (DF)
12 :52 :50.053628 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 278 :1176(898) ack 19 win 10136 <nop,nop,timestamp 1075174202 154033068>
12 :52 :50.160740 eth0 < www1.cnn.com .www > cericon.cranzgot.co.za.4064 :
   P1176 :1972(796) ack19 win 10136 <nop,nop,timestamp 1075174202 154033068>
12 :52 :50.220067 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   . 19 :19(0) ack 1972 win 31856 <nop,nop,timestamp 154033156 1075174202> (DF)
12 :52 :50.824143 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 1972 :3420(1448) ack 19 win 10136 <nop,nop,timestamp 1075174262 154033131>
12 :52 :51.021465 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 3420 :4868(1448) ack 19 win 10136 <nop,nop,timestamp 1075174295 154033165>

.............
.............

12 :53 :13.856919 eth0 > cericon.cranzgot.co.za.4064 > www.cnn.com.www :
   . 19 :19(0) ack 53204 win 30408 <nop,nop,timestamp 154035528 1075176560> (DF)
12 :53 :14.722584 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
  ; 53204 :54652(1448) ack 19 win 10136 <nop,nop,timestamp 1075176659 154035528>
12 :53 :14722738 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   . 19 :19(0) ack 54652 win 30408 <nop,nop,timestamp 154035615 1075176659> (DF)
12 :53 :14.912561 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 54652 :56100(1448) ack 19 win 10136 <nop,nop,timestamp 1075176659 154035528>
12 :53 :14.912706 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   . 19 :19(0) ack 58500 win 30408 <nop,nop,timestamp 154035634 1075176659> (DF)
12 :53 :15.706463 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
```

```
   . 58500 :59948(1448) ack 19 win 10136 <nop,nop,timestamp 1075176765 154035634>
12 :53 :15.896639 eth0 < www1.cnn.com.www > cerincon.cranzgot.co.za.4064 :
   . 59948 :61396(1448) ack 19 win 10136 <nop,nop,timestamp 1075176765 154035634>
12 :53 :15.896791 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   .19 :19(0) ack 61396 win 31856 <nop,nop,timestamp 154035732 1075176765> (DF)
12 :53 :16.678439 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 61396 :62844(1448) ack 19 win 10136 <nop,nop,timestamp 1075176864 154035732>
12 :53 :16.867963 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 62844 :64292(1448) ack 19 win 10136 <nop,nop,timestamp 1075176864 154035732>
12 :53 :16.868095 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   . 19 :19(0) ack 64292 win 31856 <nop,nop,timestamp 154035829 1075176864> (DF)
12 :53 :17.521019 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   FP 64292 :65200(908) ack 19 win 10136 <nop,nop,timestamp 1075176960 154035829>
12 :53 :17.521154 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   . 19 :19(0) ack 65201 win 31856 <nop,nop,timestamp 154035895 1075176960> (DF)
12 :53 :17.523243 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   F 19 :19(0) ack 65201 win 31856 <nop,nop,timestamp 154035895 1075176960> (DF)
12 :53 :20.410092 eth0 > cericon.cranzgot.co.za.4064 > www1.cnn.com.www :
   F 19 :19(0) ack 65201 win 31856 <nop,nop,timestamp 154036184 1075176960> (DF)
12 :53 :20.940833 eth0 < www1.cnn.com.www > cericon.cranzgot.co.za.4064 :
   . 65201 :65201(0) ack 20 win 10136 <nop,nop,timestamp 1075177315 154035895>

103 packets received by filter
```

Le résultat qui précède requiert quelques explications. Les lignes 5, 7 et 9 (sachant que la ligne 1 est [root@cericon]#...) correspondent à l'étape de négociation. `tcpdump` utilise le format `<numéro de séquence> :<numéro de séquence + longueur de données>(<longueur de données>)` sur chaque ligne pour indiquer le contexte du paquet à l'intérieur du flux. Le numéro de séquence, **Sequence number**, est choisi aléatoirement au départ. Ainsi, `tcpdump` affiche le numéro de séquence relatif après les deux premiers paquets pour donner la position réelle dans le flux. La ligne 11 est celle où la touche **Enter** a été pressée pour la première fois et la ligne 15 correspond au second **Enter**. Les "ack 19" indiquent le moment de réception des données entrantes pour le serveur web de CNN. Dans le cas qui nous occupe, nous n'avons saisi au clavier que 19 octets. Par conséquent, le serveur web place cette valeur dans chacun de ses paquets sortants alors que, la plupart du temps, nos propres paquets sortants sont vides de données.

Les lignes 61 et 63 correspondent à l'étape de déconnexion.

Des informations plus détaillées à propos de `tcpdump` peuvent être obtenues dans `tcpdump`(8), en particulier dans la section relative aux **paquets TCP**.

27.3 Protocole datagramme utilisateur (UDP).

Il n'est pas toujours nécessaire de disposer de communications totalement *fiables*, soit pour des raisons d'efficacité, soit qu'il importe peu que des paquets soient altérés. C'est le cas des transmissions vocales où la réduction des délais de transmission est plus importante que l'intégrité des données, ou des communications de serveurs de noms pour lesquelles des transmissions de paquets uniques

sont souhaitées. Un autre cas concerne NFS (*Network File System*[2]) qui utilise UDP pour exclusivement mettre en oeuvre *via* des programmes (en anglais, *to implement*) des transferts de données à large bande passante. Avec UDP,[3], l'utilitaire client envoie et reçoit des paquets individuels qui sont encapsulés dans IP. Les ports sont utilisés de la même manière qu'avec TCP, mais ceux-ci ne sont que des identificateurs. Par conséquent, la notion de flux n'existe pas avec UDP. L'en-tête complet UDP/IP est donnée dans le tableau 27.2.

TAB. 27.2 – En-tête combiné d'IP et d'UDP.

Octets (IP)	**Description**
0	Bits 0-3 : Version, bits 4-7 : Internet Header Length (IHL)
1	Type de Service (TOS)
2-3	Longueur
4-5	Identification
6-7	Bits 0-3 : Indicateurs (*flags*), bits 4-15 : décalage ou *offset*
8	Temps de vie (*Time To Live* ou TTL)
9	Type
10-11	Somme de contrôle (*checksum*)
12-15	Adresses IP source
16-19	Adresses IP de destination
20-(IHL *4 - 1)	Options + remplissage pour grouper jusqu'à 4 octets
Octets (UDP)	**Description**
0-1	Port source
2-3	Port destination
4-5	Longueur
6-7	Somme de contrôle (*checksum*)
Les données UDP commencent à IHL * 4 + 8 et finissent à Length -1	

27.4 Le fichier `/etc/services`.

Les divers numéros de ports standards correspondent univoquement à des types particuliers de services. Comme nous l'avons vu précédemment, le port 80 est le port de destination web pour un programme client. Les ports allant de 1 à 1023 sont réservés aux services standards et un nom sous forme de texte est attribué à chacun d'eux pour rendre leur identification commode. Tous les services sont définis aussi bien pour TCP que pour UDP, même s'il n'y a pas d'accès FTP d'UDP, par exemple. Les numéros de port inférieurs à 1024 sont exclusivement réservés aux programmes dont l'**UID** est `root` tels que le courriel, DNS et les services web. Les programmes des utilisateurs ordinaires n'ont pas l'autorisation de *se lier* aux ports inférieurs à 1024. [La liaison (ou *port binding*) est une opération au cours de laquelle un programme réserve un port de manière à écouter les connexions entrantes, comme le font tous les services réseau]. Ces ports sont définis dans le fichier `/etc/services` qui constitue une

[2] NdT : NFS est un système de partage de fichiers sur réseau (voir le chapitre 29).

[3] NdT : UDP ne garantit pas la bonne réception des données, ni le contrôle de la séquence des paquets. Il n'apporte pas de fonctionnalités supplémentaires à IP mais permet de désigner les numéros de port correspondant aux applications avec des temps de réponses courts.

table essentiellement descriptive : les programmes peuvent y rechercher les numéros de port (et vice-versa) .
Le fichier /etc/services n'a rien à voir avec la disponibilité d'un service. Voici un extrait du fichier `/etc/services`.

```
tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/dcp
daytime         13/tcp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timeserver
time            37/udp          timeserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          #deprecated
bootps          67/tcp                          # BOOTIP server
bootps          67/udp
bootpc          68/tcp                          # BOOTIP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
www             80/udp                          # HyperText Transfer Protocol
```

27.5 Cryptage et redirection TCP.

Le flux de données TCP peut être aisément reconstruit par toute personne qui écoute sur le support physique (le câble, par exemple) et qui arrive à visualiser votre trafic réseau. En fait, TCP est connu pour être un service peu sûr en soi. Le but est donc de chiffrer (ou crypter) les données de manière à ce que

tout message capté par un tiers soit illisible. Quelles doivent être les propriétés d'un flux chiffré ?

1. Il devrait garantir que le client se connectant effectue *vraiment* cette opération sur le serveur en question. En d'autres termes, il devrait authentifier le serveur pour assurer que ce dernier n'est pas un cheval de Troie (ou *Trojan*, en anglais).
2. Il devrait éviter que des parties des données soient lisibles par un pirate. Toutes données lues par une tierce personne devraient apparaître sous forme cryptée.
3. Il devrait être impossible pour quelqu'un qui a capté le message de modifier le trafic des données sans se faire repérer.

Deux paquets permettent de rencontrer aisément ces objectifs. Supposons que nous voulions utiliser POP3 (*Post Office Protocol* permet à la station de travail d'un utilisateur d'accéder au courrier depuis un serveur de courrier) pour rechercher du courriel provenant d'une autre machine. Premièrement, nous pouvons vérifier que POP3 est en cours d'exécution en se connectant au serveur POP3 lui-même. Lancez un `telnet` sur le port 110 (qui est le service POP3) :

```
telnet localhost 110
Connected to localhost.localdomain.
Escape character is '^]'
+OK POP3 localhost.localdomain v7.64 server ready
QUIT
+OK Sayonara
connexion closed by foreign host
```

Pour rencontrer la première contrainte, nous allons utiliser le paquet OpenSSH. Nous pouvons initialiser et exécuter le démon `sshd` (*Secure Shell daemon*) s'il ne l'est pas déjà. Les commandes suivantes peuvent être appliquées au serveur POP3 :

```
ssh-keygen -b 1024 -f /etc/ssh/ssh_host_key -q -N ''
ssh-keygen -d -f /etc/ssh/ssh_host_dsa_key -q -N ''
sshd
```

Pour créer un canal crypté comme indiqué à la figure 27.1, nous utiliserons le programme de connexion client `ssh` d'une manière un peu spéciale. Nous souhaitons le faire écouter sur un port TCP particulier et ensuite, crypter puis rediriger tout le trafic vers le port TCP distant du serveur.

Cette technique est connue sous le nom de *redirection de port crypté* (*encrypted port forwarding*). Sur la machine cliente, nous sélectionnons un port quelconque mais inutilisé. Dans le cas présent, il s'agit de `12345` :

```
ssh -C -c arcfour -N -n -2 -L 12345 :<pop3-server.doma.in> :110 \
<pop3-server.doma.in> -l <user> -v
```

où `<user>` est le nom d'un compte shell courant sur le serveur POP3. Finalement, toujours sur la machine cliente, nous effectuons :

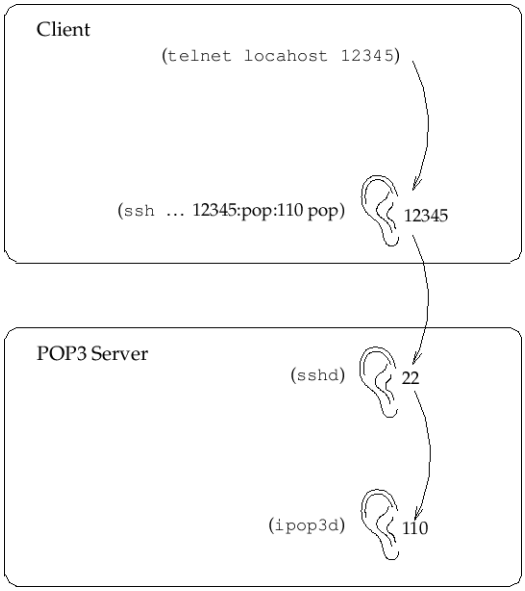


FIG. 27.1 – Redirection entre deux machines.

```
telnet localhost 12345
Connected to localhost.localdomain
Escape character is '^]'
+OK POP3 localhost.localdomain v7.64 server ready
QUIT
+OK Sayonara
connexion closed by foreign host
```

Nous obtenons un résultat identique à celui qui précède. En effet, en ce qui concerne le serveur, la connexion POP3 provient d'un client sur la machine serveur elle-même (qui ne sait pas que cette connexion a pour origine `sshd`) qui est en train d'effectuer une redirection à partit d'un client `ssh`. L'option `-C` compresse toutes les données (ce qui est utile dans le cas des connexions à bas débit). Notez également qu'en général, vous ne devriez utiliser aucun cryptage avec `arcfour` et le protocole SSH 2 (option `-2`).

La seconde méthode repose sur le programme `forward` du paquet `mirrordir`. Ce programme possède un protocole de cryptage unique qui agit similairement à OpenSSH bien qu'il n'ait pas été testé de manière approfondie par la communauté (il ne devrait donc être utilisé qu'avec précaution). Sur une machine agissant comme serveur, tapez simplement `secure-mcserv`. Sur la machine cliente :

```
forward <user>@<pop3-server.doma.in> <pop3-server.dom.in> :110 \
                                        12345 --secure -z -K 1024
```

et ensuite lancez `telnet` pour effectuer un test comme précédemment.

Avec la redirection, vous pouvez utiliser tout client POP3 comme vous le feriez normalement. Cependant, veillez à attribuer `localhost` à votre machine hôte et `12345` comme adresse de port pour votre client POP3.

Cet exemple peut, naturellement, être transposé à presque tous les services. Avec certains de ceux-ci qui effectuent des opérations spéciales comme la création de connexion de retour TCP sur un client (c'est le cas avec FTP, par exemple), cela ne fonctionnera pas.

Chapitre 28

DNS et résolution de noms.

Chaque ordinateur sur l'internet possède sa propre adresse IP. Bien que cette dernière soit suffisante pour identifier une machine lorsqu'il s'agit de transmettre des paquets, il ne nous est pas particulièrement commode de l'utiliser. Il nous est plus aisé d'identifier une machine par un nom, au cas où celle-ci serait déplacée par exemple.

Par conséquent, une machine reçoit un nom qui s'avère plus descriptif que l'adresse IP. Le nom d'une machine est appelé [selon ma terminologie] *le nom d'hôte non-qualifié* et il est usuellement constitué de moins de huit caractères sous forme de minuscules ou de chiffres (sans caractères "point"). Il arrive fréquemment que des groupes d'ordinateurs possèdent *un nom de domaine*. Le nom complet d'une machine est alors *nom_hote_non-qualifie.nom_domaine* ce qui constitue le *nom d'hôte pleinement qualifié* (*fully qualified host name*) [terminologie standard] ou le *nom d'hôte qualifié* [selon ma terminologie]. A titre d'exemple, mon ordinateur s'appelle `cericon`. Le nom de domaine de mon entreprise est `cranzgot.co.za`; donc, le nom d'hôte (pleinement) qualifié de mon ordinateur est `cericon.cranzgot.co.za`, l'adresse IP étant `160.123.76.9`.

Souvent, le mot *domaine* (*domain* en anglais) est synonyme de *nom de domaine* (*domain name*), et le *nom d'hôte* (en anglais *hostname*) signifie souvent en soi le nom d'hôte (c'est-à-dire le nom de la machine) qualifié ou non-qualifié.

La méthode permettant de nommer les ordinateurs et de faire correspondre, adresses IP et noms d'hôte est appelée *Domain Name System* (*DNS*).

28.1 Domaines de haut-niveau (TLDs).

Dans le titre, l'acronyme TLDs signifie *Top-Levels Domains* (domaines de hauts niveaux). Les domaines sont souvent désignés dans une adresse littérale sous forme d'un suffixe standard comme ceux de la liste qui suit :

.com une entreprise américaine ou internationale. En fait, toute organisme peut avoir un domaine **.com**.

.gov un organisme américain gouvernemental.

.edu une université américaine.

.mil un département militaire américain.

.int un organisme établi par les traités internationaux.

.org un organisme américain ou à but non-lucratif. En fait, tout le monde peut posséder un domaine **.org**.

.net un fournisseur d'accès internet (FAI, ou *ISP* en anglais pour désigner *Internet Service Provider*). Tout revendeur de bande passante peut y avoir droit, toute entreprise de technologie d'information ou toute entreprise peuvent avoir accès à ce type de domaine.

A côté de ce qui vient d'être décrit, le domaine peut se terminer par le code national. Le domaine `.us` est rarement utilisé étant donné que les domaines `.com`, `.edu`, `.org`, `.mil`, `.gov`, `.int` ou `.net` sont les plus courants.

Le domaine associé à un pays peut être précédé d'un descriptif plus fin. Chaque pays peut utiliser une structure propre. Voici quelques exemples pour l'Afrique du Sud :

.co.za une entreprise (company) sud-africaine (`za` = Zuid Afrika en néerlandais).

.org.za un organisme sud-africain à but non-lucratif.

.ac.za une université (academic university) sud-africaine.

.edu.au une institution éducative australienne.

.gov.za un organisme gouvernemental (government organisation) sud-africain.

28.2 Résolution de noms DNS vers des adresses IP.

En pratique, un utilisateur saisit un nom d'hôte dans une application, à titre d'exemple : `www.cranzgot.co.za` dans un navigateur internet. L'application est chargée de trouver l'adresse IP associée à ce nom d'hôte de manière à lui envoyer des paquets. La présente section décrit la méthode de recherche sur l'internet, de sorte que tout un chacun puisse trouver l'adresse IP de n'importe qui d'autre.

Une solution de recherche triviale serait de distribuer une longue liste des noms d'hôte en vis-à-vis des adresses IP et ce, sur chaque machine faisant partie d'internet. Cette solution s'avère rapidement impraticable dès qu'il y a plusieurs milliers de machines. Une autre solution imaginable serait d'avoir à disposition un énorme ordinateur quelque part sur l'internet, dont l'adresse est connue de tous. Cet ordinateur aurait pour responsabilité de répondre aux requêtes de recherche des numéros IP. Ainsi, une application lancée sur une machine locale n'aurait plus qu'à interroger cette machine particulière. Naturellement, avec des milliards de machines constituant l'internet, cette solution engendrerait un trafic internet extrêmement intense. [En vérité, certains réseaux Microsoft pratiquent de cette manière, ce qui n'est pas astucieux].

28.2.1 L'infrastructure DNS d'internet.

La solution de recherche réellement utilisée sur internet est la suivante. Il existe des millions ordinateurs dont le service est de trouver des adresses IP. Ce sont des *serveurs de noms* (ou *serveurs DNS*). Les requêtes qu'ils résolvent sont appelées des *requêtes DNS* (*DNS lookup* ou simplement *lookup*). Chaque serveur de noms ne possède des informations qu'à propos d'une partie d'internet. En permance, ces serveurs s'interrogent mutuellement.

Il n'y a que 13 serveurs de noms principaux (`root`) sur l'internet. [Leur liste peut être obtenue à l'adresse suivante : ftp ://ftp.rs.internic.net/domain/named.root] :

```
a.root-servers.net  198.41.0.4
b.root.servers.net  128.9.0.107
c.root-servers.net  192.33.4.12
d.root-servers.net  128.8.10.90
e.root-servers.net  192.203.230.10
f.root-servers.net  192.5.5.241
g.root-servers.net  192.112.36.4
h.root-servers.net  128.63.2.53
i.root-servers.net  192.36.148.17
j.root-servers.net  198.41.0.10
k.root-servers.net  193.0.14.129
l.root-servers.net  198.32.64.12
m.root-servers.net  202.12.27.33
```

Chaque pays possède aussi un serveur de noms. Chaque organisme à son tour possède un serveur de noms. Chaque serveur de noms n'a d'informations qu'au sujet des machines se trouvant dans son propre domaine, ainsi qu'au sujet des autres serveurs de noms. La structure de cet ensemble est donc une arborescence inversée. Les serveurs de noms principaux ou "root" n'ont d'information sur les adresses IP qu'à propos des noms de serveurs `.com`, `.edu`, `.za`, etc. Le serveur de noms `.za`, lui-même ne possède des informations IP qu'au sujet de `.co.za`, `.ac.za`, `.org.za`, etc. Et à son tour, `.co.za` ne possède de renseignements IP que concernant les serveurs de toutes les compagnies sud-africaines comme, `.cranzgot.co.za`, `.icon.co.za`, `.mweb.co.za`, etc. Finalement, le serveur de noms `.cranzgot.co.za` ne connaît que les IP des machines de Cranzgot Systems, comme `www.cranzgot.co.za`.

Dans ses fichiers de configuration , votre propre machine définit un serveur de noms qui lui est proche géographiquement. Les responsabilités de ce serveur de noms sont (i) de répondre à toute requête au sujet de son nom de domaine, pour lequel il a de l'information et (ii) de répondre à toute autre requête en interrogeant autant de serveurs de l'internet qu'il sera nécessaire de contacter pour satisfaire la demande initiale.

28.2.2 Le processus de résolution de noms.

Dans cette section, nous présentons un cas impliquant `www.cranzgot.co.za`. La séquence de requêtes se déroule de manière à traduire ce nom en une adresse IP. Ce procédé est appelé *résolution de noms d'hôte* et l'algorithme qui réalise cette opération est appelé le *résolveur*.

1. L'application cliente vérifie plusieurs bases de données sur la machine locale. Si elle peut obtenir une réponse directe, elle ne procède pas plus loin.
2. Sinon, l'application effectue une requête sur un serveur de noms, géographiquement proche, et ce à partir d'un fichier de configuration locale. Disons que cette machine se nomme `ns`.

3. L'application fait une requête à `ns` avec "www.cranzgot.co.za ?".
4. `ns` détermine si cette adresse a été récemment recherchée. S'il la possède, il n'y a pas lieu de continuer, vu que le résultat devrait être enregistré dans une mémoire cache locale. L'adresse IP y est accessible.
5. `ns` vérifie si le domaine est local, c'est-à-dire s'il s'agit d'un ordinateur à propos duquel il possède une information directe. Ceci ne sera vrai que si `ns` est le serveur de noms propre à `cranzgot.co.za`.
6. Autrement, `ns` élimine le TLD (Top Level Domain) `.za`. Il effectue une requête à un serveur de noms principal "root" (celui-ci est pré-enregistré), en lui demandant quel serveur de noms est responsable de `.za`. La réponse sera `ucthpx.uct.ac.za` d'adresse IP = `137.158.128.1`.
7. `ns` élimine alors le domaine suivant `co.za` et effectue une requête à `137.158.128.1`, lui demandant quel est le nom de serveur responsable de `.co.za`. La réponse sera `secdns1.posix.co.za` dont l'adresse IP est `160.124.112.10`.
8. `ns` élimine `cranzgot.co.za` et effectue une requête auprès de `160.124.112.10`, lui demandant qui est responsable pour `cranzgot.co.za`. La réponse sera `pizza.cranzgot.co.za` d'adresse IP = `196.28.123.1`.
9. `ns` effectue une requête auprès de `196.28.123.1` en lui enjoignant de lui donner l'adresse de `cranzgot.co.za`. La réponse est `160.123.176.1`.
10. `ns` retourne la réponse à l'application.
11. `ns` enregistre chacun des résultats obtenus dans une mémoire cache locale avec une date d'expiration de manière à s'éviter une recherche complète en seconde fois.

28.3 Configuration de votre machine locale.

Nous discutons ici des fichiers de configuration locaux tels qu'évoqués dans la section précédente. Il n'y a que trois fichiers concernés : `/etc/host.conf`, `/etc/hosts` et `/etc/resolv.conf`. Ils spécifient comment toutes les applications procèdent à une requête d'adresse IP. Notez qu'ils n'ont rien à voir avec les fichiers de configuration du démon serveur de noms lui-même, même si un démon serveur de noms peut être en cours d'exécution sur la machine locale.

Lorsqu'une application doit effectuer une requête à propos d'un nom d'hôte, elle procède ainsi [en réalité, l'application fait un appel de bibliothèque **C** sur la fonction `gethostbyname()`, donc tous ces fichiers de configuration appartiennent en fait aux paquets `glibc` ou `libc` de la bibliothèque **C**. Cependant, ceci est peu important en terme d'administration]. Les explications qui suivent reprennent en détail les étapes 1, 2 et 3 décrites dans la section qui précède. Les fichiers de configuration qui sont mentionnés appartiennent à une installation réelle.

1. L'application vérifie le fichier `/etc/host.conf`. Ce fichier possède usuellement une ligne `order hosts, bind` qui indique que la vérification devrait d'abord concerner (c'est le sens du terme `hosts`) la base locale de données `/etc/hosts` et, rechercher (c'est le sens du terme `bind`) le serveur de noms indiqué dans `/etc/resolv.conf`. Le fichier `/etc/hosts` contient une liste de correspondance des adresses IP et des noms. Un exemple est fourni ci-dessous. Si l'application peut obtenir une réponse à l'issue de cette étape, elle ne procède pas plus loin.

2. L'application recherche la ligne `nameserver <server_nom>` dans le fichier `/etc/resolv.conf`. Il peut y avoir trois lignes de ce type : si une des lignes n'est pas valide, l'application utilise la ligne suivante.
3. L'application envoie une requête avec le nom d'hôte à un serveur de noms. Si le nom d'hôte est non-qualifié, l'application –avant d'essayer la recherche– ajoute un nom de domaine local au nom d'hôte local. Une ligne `search <domain1> <domain2> ... <domainN>` peut être ajoutée à `/etc/resolv.conf` jusqu'à ce que la requête retourne avec succès une adresse IP. Ceci vous évite de devoir taper tous les noms d'hôte des ordinateurs de votre réseau.
4. Le serveur de noms procède selon la hiérarchie des requêtes à partir de l'étape 4, tel que cela a été vu précédemment.

Le fichier `/etc/hosts` devrait ressembler à ceci :

```
127.0.0.1       localhost.localdomain    localhost
192.168.3.9     cericon.cranzgot.co.za   cericon
192.168.3.10    pepper.cranzgot.co.za    pepper
192.168.2.1     onion.cranzgot.co.za     onion
```

Les hôtes `pepper`, `cericon` et `onion` sont ceux avec lesquels cette machine communique le plus. Par conséquent, leur nom est cité. `cericon` est la machine locale et doit être répertoriée. Vous pouvez inclure n'importe quel hôte auquel vous voulez adresser des requêtes rapides ou des hôtes qui doivent être connus. Le fichier `/etc/host.conf` pourrait avoir l'allure suivante :

```
order     hosts, bind, nis
trim      some.domain
spoofalert
nospoof
multi     on
reorder
```

order ce terme désigne l'ordre dans lequel les requêtes sont effectuées. N'essayez pas de jouer sur cette valeur. Elle semble ne pas avoir d'effet. Vous devriez laisser la ligne `order hosts, bind` telle qu'elle (ou utiliser `order hosts, bind, nis`, si vous avez un NIS[1] –voir le `HOWTO` sur le web).

trim ôte le domaine `un.domaine` se trouvant à la fin du nom d'hôte qualifié avant de tenter une requête.Vous n'aurez probablement jamais besoin de cette option.

spoofalert tente des requêtes inverses sur un nom d'hôte après avoir effectuer une requête d'IP : `spoofalert` demande de trouver le nom d'hôte à partir d'une IP. Si la demande ne retourne pas le résultat correct, cela peut signifier qu'une machine essaye de se faire passer pour ce qu'elle n'est pas.

[1] NdT : il s'agit du service d'information du réseau (*Network Information Service*) qui permet à certaines informations d'être connues par toutes les machines disponibles du réseau. Ce service est géré par la bibliothèque **C** standard, `libc` de LINUX. La version NIS+ présente un niveau de sécurité supérieur ainsi qu'une convivialité améliorée pour les installations de grande taille.

C'est une astuce de pirate (ou *cracker*) appellée "usurpation d'identité" (ou en anglais : *spoofing*). `spoofalert` vous avertit d'une telle tentative dans votre journal `/var/log/messages`.

nospoof écarte les résultats qui n'ont pas réussi le test d'usurpation d'identité.

multi on retourne plus d'un résultat s'il y a des alias ou synonymes. En réalité, un hôte peut avoir plusieurs adresses IP et une adresse IP peut correspondre à plusieurs noms d'hôtes. Considérons un ordinateur ayant plusieurs noms (`ftp.cranzgot.co.za` et `www.cranzgot.co.za` sont la même machine), ou alors une machine qui est équipée de plusieurs cartes réseau avec une adresse pour chacune d'elles. L'option `multi on` devrait toujours être activée. `multi off` est une autre option possible. Pour la plupart, les applications utilisent seulement la première valeur retournée.

reorder si plus d'une adresse IP est retournée lors d'une requête, `reordrer` affiche une liste selon l'IP qui présente la route réseau la plus aisée à emprunter.

Bien que toutes ces options existent, un fichier `/etc/hosts.conf` ressemble généralement à ceci :

```
order     hosts, bind
multi     on
```

Le fichier `/etc/resolv.conf` pourrait se présenter ainsi :

```
nameserver 192.168.2.1
nameserver 160.123.76.1
nameserver 196.41.0.131
search cranzgot.co.za ct.cranzgot.co.za uct.co.za
sortlist 192.168.3.0/255.255.255.0 192.168.2.0/255.255.255.0
options ndots :1 timeout :30 attempts :2 rotate np-check-names inet6
```

nameserver spécifie un serveur de noms à rechercher. Il n'y aura pas plus de trois de ces serveurs listés. Le fait d'en avoir plus d'un est une sécurité si jamais un des serveur de noms était hors service. Si c'était le cas, les autres membres de la liste seraient pris en considération séquentiellement.

search si un nom d'hôte avec moins de `ndots` points était donné (c'est-à-dire `1` dans ce cas ; voir la dernière ligne de l'encart), `search` ajouterait chacun des domaines tour à tour au nom d'hôte, en essayant une requête avec chaque combinaison. Cette option vous permet de saisir un nom d'hôte non-qualifié ; l'application détermine à partir de la liste fournie quel est l'organisme auquel cette machine appartient. Vous pouvez mentionner jusqu'à six domaines mais alors, les requêtes prendront beaucoup de temps.

domain la ligne "`domain ct.cranzgot.co.za`" désigne la même chose que "`search ct.cranzgot.co.za cranzgot.co.za co.za`" . Utilisez toujours `search` explicitement au lieu de `domain` de façon à réduire le nombre de requêtes.

options divers paramètres supplémentaires peuvent être précisés dans cette ligne :

ndots voir le cas `search` ci-dessus. Par défaut, la valeur est `1` ;

timeout indique la durée au bout de laquelle une requête est considérée comme ayant échoué. La valeur par défaut est 30 s ;

attempts est le nombre d'essais à réaliser avant de considérer qu'il y a échec. La valeur par défaut est 2. Ceci signifie que si une application contacte un serveur de nom hors service, elle effectuera la requête durant 1 minute avant d'avertir qu'elle ne peut résoudre l'adresse IP.

rotate essaie les serveurs de noms tour-à-tour (dans le jargon : "mode *round-robin*"). Ceci permet de distribuer la charge des requêtes sur différents serveurs de noms.

no-check-names permet de ne pas vérifier les caractères non-valides des noms d'hôtes.

inet6 la page de man associée à `resolv.conf` (`resolver`(5)) précise ceci :

> `inet 6` `sets RES_USE_INET6 in _res.options. This has the effect of trying a AAAA query before an A query inside the gethostbyname function, and of mapping IPv4 responses in IPv6 "tunneled form" if no AAAA records are found but an A record set exists.`

Une requête `AAAA` est une adresse internet IP de type "IPv6" c'est-à-dire de "prochaine génération".

En dépit de cette gamme d'options, un fichier `/etc/resolv.conf` ressemble le plus souvent à ceci :

```
nameserver 192.168.2.254
search cranzgot.co.za
```

28.4 Requêtes inverses.

Une requête inverse (*reverse lookup*), telle que mentionnée sous la rubrique `nospoof` vue à la section précédente, consiste en la détermination du nom d'hôte à partir de l'adresse IP. Le cours d'une telle requête est similaire à celui des requêtes directes qui utilisent une partie de l'adresse IP pour trouver quelles machines sont responsables de telle ou telle gamme d'adresses IP.

Une requête directe (*forward lookup*) est une requête ordinaire car elle est effectuée dans le sens "adresse IP → nom d'hôte".

28.5 *Autoritaire* sur un domaine.

Il a été souligné que les serveurs de noms ne manipulent que l'information relative à leurs domaines. Toute autre information qu'un serveur de noms possède au sujet d'un domaine autre est mise en cache, sous forme de données temporaires accompagnées d'une date d'expiration.

Lorsqu'un serveur possède une information directe sur un domaine, ce serveur est dit "*autoritaire pour le domaine*". Par exemple, le serveur `ns2.cranzgot.co.za` est autoritaire pour le domaine `cranzgot.co.za`. Dès lors, les re-

quêtes provenant d'où que se soit sur l'internet et ayant pour domaine cranzgot-.co.za sont du ressort de ns2.cranzgot.co.za.

28.6 Les commandes host, ping et whois.

La commande host permet de rechercher un nom d'hôte mais aussi une adresse IP en effectuant une requête de serveur de noms. Essayez :

```
host www.cnn.com
```

pour obtenir diverses adresses IP à partir d'un nom d'hôte. Réessayez host plusieurs fois. Observez que l'ordre de sortie des adresses change de manière aléatoire. Ceci montre que la charge est distribuée sur divers serveurs cnn.com.

A présent, prenez une des adresses IP et tapez :

```
host <adresse-IP>
```

Cette commande retourne le nom d'hôte cnn.com.

Notez que la commande host n'est pas disponible sur tous les systèmes UNIX.

Bien que la commande ping ne soit pas directement liée à DNS, elle constitue un moyen rapide d'obtention d'une adresse IP et de vérification qu'un hôte est capable de répondre à des paquets ICMP. Elle est souvent utilisée comme test-clé pour vérifier la configuration d'un réseau et le bon fonctionnement du DNS. Voir la section 26.10.1.

A présent, saisissez :

```
whois cnn.com@rs.internic.net
```

(La commande originale whois de BSD –Berkeley Software Development– fonctionnait comme whois - <host> <user>). Vous obtiendrez une réponse comme celle-ci :

```
[rs.internic.net]

Whois Server Version 1.1

domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http ://www.internic.net
for detailed information

  Domain Name :CNN.COM
  Registrar : NETWORK SOLUTIONS, INC
  Whois Server : whois.networksolutions.com
  Name Server : NS-01A.ANS.NET
  Name Server : NS-01B.ANS.NET
  Name Server : NS-02A.ANS.NET
  Name Server : NS-02B.ANS.NET
```

```
  Updated Date : 22-sep-1999


>>> Last update of whois database : Thu, 20 Jan 00 01 :39 :07 EST <<<

The Registry database contais ONLY, .COM, .NET, .ORG, .EDU domains and
Registrars
```

(Internic parvient à obtenir cette base de données des domaines `.com`, `.net`, `.org` et `.edu`).

28.7 La commande `nslookup`.

`nslookup` est un programme permettant de rechercher interactivement un serveur de noms. Si vous effectuez :

```
nslookup
```

vous obtiendrez une invite derrière laquelle vous pourrez saisir diverses commandes. Si vous entrez un nom d'hôte, `nslookup` vous retournera son ou ses adresse(s) IP ou *vice versa*. En tapant :

```
help
```

derrière l'invite, vous obtiendrez une liste complète des commandes. Par défaut, `nslookup` utilise le premier serveur de noms mentionné dans `/etc/resolv.conf` pour toutes ses requêtes. Cependant, la commande

```
server <serveur_de_noms>
```

force `nslookup` à se connecter au serveur de noms de votre choix.

28.7.1 Les *enregistrements* `NS`, `MX`, `PTR`, `A` et `CNAME`.

Le terme *record* (ou *enregistrement*) est un élément d'information DNS. Entrez la commande :

```
set type=NS
```

Ceci indique à `nslookup` de retourner le second type d'information que DNS peut fournir : le serveur de noms autoritaire pour un domaine ou l'enregistrement (*record*) `NS` du domaine. Vous pouvez entrer ici n'importe quel domaine. Par exemple, si vous saisissez :

```
set type=NS
cnn.com
```

`nslookup` retournera :

```
Non-authoritative answer
cnn.com nameserver = NS-02B.ANS.NET
cnn.com nameserver = NS-02A.ANS.NET
cnn.com nameserver = NS-01B.ANS.NET
cnn.com nameserver = NS-01A.ANS.NET

Authoritative answers can be found from :
NS-02B.ANS.NET internet address = 207.24.245.178
NS-02A.ANS.NET internet address = 207.24.245.179
NS-01B.ANS.NET internet address = 199.221.47.8
NS-01A.ANS.NET internet address = 199.221.47.7
```

Ceci nous apprend que quatre serveurs de noms sont autoritaires pour le domaine `cnn.com` (un plus trois sauvegardes). Cela nous apprend aussi que la source de renseignements n'est pas autoritaire si bien que les données proviennent d'un cache. Enfin, nous voyons quels serveurs de noms sont autoritaires.
Maintenant, passons à un serveur de noms autoritaire pour `cnn.com` :

```
server NS-02B.ANS.NET
```

et lançons la même requête :

```
cnn.com
```

Le nouveau résultat est plus détaillé mais peu différent.

Il y a quelques autres enregistrements (ou *records*) que nous pouvons obtenir d'un serveur de noms.

```
set type=MX
cnn.com
```

Ceci nous permet d'obtenir l'enregistrement *MX* pour le domaine `cnn.com`. Cet enregistrement MX est le serveur responsable de la manipulation du courriel destinée à ce domaine. Les enregistrments *MX* possèdent une priorité (usuellement 10 ou 20). Ceci indique à tout serveur de messagerie de passer en priorité 20 s'il y avait échec avec une priorité de 10. En général , il n'y a qu'un ou deux enregistrements de type *MX*. Le courrier est, en réalité, le seul service internet manipulé par DNS (par exemple, il n'y a pas d'enregistrement NEWSX pour les news ou d'enregistrement WX pour les pages web, quel que soit le type d'information que nous puissions penser qu'ils gèrent).

Essayez aussi :

```
set type=PTR
<adresse-IP>
set type=A
<nom_hote>
set type=CNAME
<nom_hote>
```

Les enregistrements PTR sont des requêtes inverses, ou des *PoinTeuRs* (*PoinTeRs*) vers des noms d'hôtes. Les enregistrements A sont des requêtes directes. Les requêtes directes constituent le type de requêtes par défaut lorsqu'on invoque la première fois qu'on lance nslookup et le type de requêtes que, pour moitié, nous avons vues dans ce chapitre. Le terme A désigne les requêtes d'*Adresses* (*Adresses lookups*). Les enregistrements CNAME sont des requêtes de noms canoniques (*Canonical NAMEs*). DNS permet d'attribuer des synonymes (ou *alias*) à une machine. Le nom réel est dit "canonique". Les requêtes CNAME retournent le nom principal d'une machine.

28.8 La commande dig.

dig est l'acronyme de *domain information groper*. Cette commande permet d'envoyer des requêtes uniques à un serveur DNS aux fins de tests ou d'écriture (elle se comporte comme nslookup mais n'est pas interactive).

Usuellement, elle est utilisée comme :

```
dig @<serveur> <domaine> <type_de_requete>
```

où, <serveur> est la machine sur laquelle est exécuté le démon DNS à rechercher, <domain> est le domaine d'intérêt et <type_de_requete> est un des enregistrements tels que A, ANY, MX, NS, SOA, HINFO, AXFR ou ceux moins triviaux que détaille dig(1). dig peut également être utilisée pour tester une connexion internet : voir la section 21.7.4.

L'enregistrement AXFR est utile. Par exemple,

```
dig @dns.dial-up.net icon.co.za AXFR
```

affiche la liste complète d'un de vos fournisseurs d'accès internet (FAI).

Chapitre 29

Système de fichiers en réseau NFS.

Ce chapitre traite de NFS (*Network File System*), une particularité des systèmes UNIX qui consiste à partager des fichiers à travers un réseau. Nous décrivons comment installer des répertoires pouvant être partagés avec d'autres machines UNIX.[1]

Dès qu'on pense à l'Ethernet haute vitesse, la possibilité logique d'un partage (*sharing*) d'un ou plusieurs systèmes de fichiers au travers un réseau vient à l'esprit. MS-DOS, OS/2, Apple Macintosh et Windows possèdent leur propre schéma de partage (IPX, SMB, etc.). NFS est spécifique aux systèmes UNIX.

Considérez un disque dur et ses dizaines de milliers de fichiers. Ethernet est suffisamment rapide pour que vous puissiez utiliser le disque dur d'une autre machine, en transférant les données nécessaires sous forme de paquets réseau. Il est également possible de rendre "visible" aux autres machines une arborescence de votre propre système. NFS est une suite logicielle réalisant cet objectif relativement ardu : en fait, il s'agit d'un protocole standard qui fonctionne aussi sous LINUX. En revanche, la configuration en est assez simple. Contrairement à d'autres protocoles de partage, NFS permet le partage de fichiers mais pas d'imprimantes ou de messageries.

29.1 Logiciels.

Selon votre distribution, les programmes cités ci-dessous peuvent se trouver dans les répertoires `bin` ou `sbin`. Tous sont des démons. Pour que NFS puisse fonctionner, ces services doivent être démarrés dans l'ordre suivant :

portmap (aussi appelé parfois `rpc.portmap`). Ce démon établit une correspondance des noms de services et des ports. Les processus clients et serveurs doivent demander un numéro de port TCP basé sur un nom de service, et `portmap` gère ces requêtes. En fait, il s'agit d'une version réseau de votre fichier `/etc/services`.

[1] NdT : le lecteur devrait se rapporter à la page principale du projet OpenAFS, qui est un système de partage de fichiers plus récent. Voir `http ://www.openafs.org`

rcp.mountd (aussi appelé `mountd`). Ce service gère la requête initiale entrante provenant d'un client pour monter (`to mount`) un système de fichiers. Il vérifie que cette requête est autorisée.

rpc.nfsd (appelé également `nfsd`). Ce service est le coeur de NFS : c'est le programme serveur de fichiers lui-même.

rpc.lockd (ou `lockd`). Ce démon gère les verrouillages entre différentes machines, sur le même fichier à travers le réseau.

L'acronyme RPC signifie *Remote Procedure Call*. RPC a été développé avec NFS par Sun Microsystems. Il s'agit d'une méthode efficace pour tout programme qui appelle une fonction sur une autre machine. Il peut être utilisé par tout service qui doit effectuer un traitement distribué. De nos jours, RPC n'est plus guère utilisé que par NFS car il a été surclassé par la technologie CORBA [la version orientée-objet de RPC]. Cependant, vous pouvez écrire des applications distribuées grâce à la mise en oeuvre de RPC sous LINUX.

29.2 Exemple de configuration.

Pour partager un répertoire avec une machine distante, il faut que les requêtes DNS directes et inverses fonctionnent aussi bien pour la machine serveur que pour les machines clientes. Le DNS est étudié aux chapitres 28 (volet client) et 41 (volet serveur). Si vous envisagez de seulement tester NFS et de partager les répertoires avec votre machine locale (ce que nous sommes en train de voir), vous *pourriez* constater que DNS est déjà en fonctionnement alors qu'il n'est pas configuré correctement. Le fichier `/etc/hosts` de votre machine locale devrait au moins posséder des entrées correctes (voir la page 320).

La première étape consiste à choisir le répertoire que vous souhaitez partager. Une astuce consiste à partager votre CD-ROM avec l'ensemble du réseau LAN. Ceci vous permet d'opérer en toute sécurité car le lecteur CD-ROM n'est accessible qu'en lecture seule. Créez un fichier `/etc/exports` contenant :

```
/mnt/cdrom 192.168.1.0/24(ro)   localhost(ro)
```

Vous pouvez immédiatement constater que le format du fichier `/etc/exports` consiste simplement en une ligne pour chaque répertoire à partager. A la suite des répertoires à partager, vient une liste des hôtes qui ont l'autorisation de se connecter. Dans le cas présent, les machines autorisées à accéder au lecteur CD-ROM sont celles dont les adresses IP se répartissent sur 24 bits sur `192.168.1`, en ce compris `localhost` qui doit être déclaré (voir la page 26.8 ; pour rappel, si nous supposons être dans un réseau de classe C, il y a 32 - 24 = 8 bits pour les adresses réseau, soit $2^8 = 256$ adresses).

A présent, montons notre CD-ROM de manière usuelle :[2]

```
mkdir -p /mnt/cdrom
mount -t iso9660 -o ro /dev/cdrom /mnt/cdrom
```

Maintenant, démarrons les démons NFS séquentiellement :

[2] NdT : il va de soi qu'il faut adapter le nom du périphérique tel que décrit dans `/etc/fstab` (`/dev/hdc`, par exemple).

```
portmap
rpc.mountd
rpc.nfsd
rpc.lockd
```

Si jamais vous effectuez des modifications dans `/etc/exports`, vous devriez aussi poursuivre en exécutant :

```
exportfs -r
```

ce qui a pour effet de relire le fichier `/etc/exports`. Si vous entrez la commande `exportfs` sans options, vous devriez obtenir :

```
/mnt/cdrom      192.168.1.0/24
/mnt/cdrom      localhost.localdomain
```

c'est-à-dire une liste des répertoires partagés et des hôtes autorisés à y accéder.

Il est utile de tester les montages depuis votre machine locale avant d'effectuer les tests sur une machine distante. Ici, nous réalisons l'opération de montage de NFS :

```
mkdir /mnt/nfs
mount -t nfs localhost :/mnt/cdrom /mnt/nfs
```

Vous constatez que la commande `mount` considère le répertoire de la machine distante comme un "périphérique" bien que son `type` soit `nfs` au lieu d'`ext2`, `ext3`, `vfat` ou `iso9660`. Le nom d'hôte distant est suivi du signe "deux points", lui-même suivi du répertoire à partager. Le chemin est *relatif à la racine*. Cette syntaxe est inusuelle comparativement à d'autres services qui nomment les fichiers relativement à un répertoire "de haut niveau" (ce qui est le cas de FTP et des serveurs web). Le test-clé consiste à lancer `ls` sur le répertoire `/mnt/nfs` de façon à vérifier que son contenu est bien le même que celui de `/mnt/cdrom`. Supposons que notre serveur s'appelle `cdromserver`, nous pouvons lancer la même commande sur toutes les machines clientes :

```
mkdir /mnt/nfs
mount -t nfs cdromserver :/mnt/cdrom /mnt/nfs
```

Si quelque chose va mal, vous rechercherez certainement tous les processus incluant `rpc`, `mount`, `nfs` ou `portmap`. L'arrêt complet de `nfs` équivaut à stopper tous ces processus, au cas où vous désireriez repartir de zéro. Il est également utile de vérifier la dernière partie des journaux :

```
tail -f /var/log/messages
tail -f /var/log/syslog
```

dans une console différente pour attendre les messages d'erreur (ou de succès). Il faut toujours faire cela quelle que soit la configuration que vous mettez en

place. Notez qu'il n'est pas toujours évident que NFS échoue à cause d'un problème de requête DNS directe ou inverse ; aussi, au préalable, vérifiez en double que ces requêtes fonctionnent. `mount` ne devrait pas donner davantage de renseignements que le message classique d'erreur de NFS : "`mount : <xyz> failed, reason given by server : Permission denied`". Un DNS erroné est également indiqué par des pauses d'une minute en cours d'opération.

En majorité, les distributions n'exigeront pas que vous démarreriez ou que vous arrêtiez les démons cités précédemment. A l'instar de la plupart des services, la mise en oeuvre de NFS par RedHat peut être invoquée simplement avec :

```
/etc/init.d/nfs start
/etc/init.d/nfslock start
```

ou avec `/etc/rc.d/init.d/` au lieu de `/etc/init.d/`. Sur Debian, les commandes équivalentes sont :

```
/etc/init.d/nfs-common start
/etc/init.d/nfs-kernel-server start
```

29.3 Droits d'accès.

Plus haut dans ce chapitre, nous avons utilisé `192.168.1.0/24(ro)` pour indiquer que nous voulions donner un accès en lecture seule à une gamme d'adresses IP. Vous pouvez aussi indiquer des noms d'hôtes contenant des caractères de remplacement :

```
/mnt/cdrom    *.mynet.mydomain.co.za(ro)
```

Si vous souhaitez donner les droits en lecture-écriture :

```
/home    *.mynet.mydomain.co.za(rw)
```

Une option supplémentaire, `no_root_squash`, désactive le traitement spécial par NFS des fichiers dont le propriétaire est `root`. Cette option s'avère utile si vous trouver que certains fichiers sont anormalement inaccessibles. `no_root_squash` ne vaut en réalité que pour les systèmes tels que les stations de travail sans disque qui requièrent un accès `root` complet sur l'ensemble du système de fichiers. Voici un exemple d'utilisation :

```
/    *.very.trusted.net(rw,no_root_squash)
```

La page de man de `/etc/exports`, `exports`(5), contient une liste complète des options.

29.4 Securité.

Pour que NFS fonctionne, divers services doivent être lancés. Ceux-ci ne sont pas utilisés hors du cadre de NFS. Beaucoup d'administrateurs agissent naïve-

ment en créant impunément des exportations de répertoires. Leurs machines deviennent la cible de pirates. Un serveur NFS ne devrait être utilisé que bien caché derrière un pare-feu. Par ailleurs, tout serveur internet exposé à l'internet ne devrait *jamais* exécuter `portmap` ni les services RPC. D'ailleurs, il est sage de désinstaller tous ces services associé à NFS, si ce dernier n'est pas en cours d'exécution.

29.5 Noyau NFS.

Il existe deux versions mettant en oeuvre NFS sous LINUX. Bien que ce qui suit constitue un avertissement technique, il est intéressant de réaliser que le serveur NFS a été originellement mis en oeuvre avant que le noyau LINUX ne supporte NFS. Debian supporte deux implémentations sous forme de deux paquets distincts : `nfs-server` et `nfs-kernel-server`. La configuration devrait être la même pour les deux paquets. Votre choix se fera en fonction des versions de ces logiciels et des performances recherchées. A ce stade-ci, vous êtes averti qu'il faut –au moins– vérifier les pages web du noyau. Naturellement, en tant que client, NFS doit nécessairement être supporté par le noyau sous forme d'un système de fichiers régulier pour permettre le montage de répertoires et de fichiers.

Chapitre 30

Les services exécutés sous inetd.

Il existe des centaines de services qu'offre toute distribution LINUX. Il serait très contraignant de devoir gérer ceux-ci en les exécutant un-à-un. Un processus-démon spécial a été développé, qui attend les connexions TCP entrantes et démarre l'exécutable sollicité. Ceci permet d'économiser du temps d'exécution. Cette technique n'est utilisée que par des services appelés occasionnellement, donc, elle ne concerne pas le web, la messagerie ou le DNS.

Le démon qui réalise cet objectif est traditionnellement `inetd` : le sujet de ce chapitre. (La section 37.1 décrit un exemple où vous écrirez votre propre service réseau sous forme d'un script de shell exécutable sous `inetd`).

30.1 Le paquet `inetd`.

Le paquet contenant `inetd` dépend de votre distribution. En effet, RedHat –depuis la version 7.0– a migré vers `xinetd`, qui s'écarte radicalement du logiciel traditionnel `inetd` d'UNIX. `xinetd` est discuté à la section 30.4. Les fichiers fondamentaux d'`inetd` sont le fichier de configuration `/etc/inetd.conf`, l'exécutable `/usr/bin/inetd`, les pages de man d'`inetd` et d'`inetd.conf`, et le script de démarrage `/etc/init.d/inet` (ou, selon les distributions : `/etc/rc.d/init.d/inetd` ou `/etc/init.d/inetd`). Discuté à la section 27.4, `/etc/services` est un autre fichier important.

30.2 Invoquer des services avec `/etc/inetd.conf`.

La plupart des services sont démarrés selon un de ces trois modes : en mode autonome ou *standalone* (ce qui est consommateur de ressources, comme souligné plus haut), sous `inetd`, ou sous `inetd` modéré par une interface TCP ("TCP-wrapper"-moderated). Cependant, certains services ne peuvent être exécutés que selon un *seul* de ces modes. Ci-dessous, nous explicitons les trois modes. Vous devrez avoir installé un paquet `ftp` pour effectuer les exemples (soit `wuftpd` sous RedHat ou `ftpd` sous Debian).

30.2.1 Invoquer un service autonome (*standalone*).

Essayez les commandes suivantes (les commandes alternatives sont données entre parenthèses) :

```
/usr/bin/in.ftpd -D
( /usr/bin/in.wuftpd -s )
```

L'option `-D` indique au service de démarrer en mode Démon (ou mode autonome, dit en anglais *standalone*). Ceci constitue la première méthode.

30.2.2 Invoquer un service `inetd`.

Avec la deuxième méthode, nous demandons à `inetd` de lancer le service pour nous. Modifiez votre fichier `/etc/inetd.conf` et ajoutez ou modifiez la ligne (l'autre possibilité est mise entre parenthèses) :

```
ftp    stream tcp   nowait root  /usr/sbin/in.ftpd in.ftpd
( ftp    stream tcp   nowait root  /usr/sbin/in.wuftpd in.wuftpd )
```

Redémarrez alors le service `inetd` :

```
/etc/init.d/inet restart
( killall -1 inetd )
( /etc/rc.d/init.d/inet restart )
```

et effectuez le test suivant :

```
ps awx | grep ftp
ftp localhost
```

Les champs constituant le fichier `/etc/inetd.conf` ont la signification suivante :

ftp : le nom du service. En consultant le fichier `/etc/services`, nous constatons que le numéro de port est 21.

stream tcp : type de connexion (*socket*) et protocole. Dans ce cas-ci, nous avons à faire avec une connexion à flux TCP.

nowait : indique de ne pas attendre que le processus quitte avant d'écouter d'autres connexions entrantes. Comparez la signification des termes `wait` et `respawn` (chapitre 33).

root : décrit l'ID de l'utilisateur initial sous lequel le service doit être exécuté.

/usr/sbin/in.ftpd (/usr/sbin/in.wuftpd) : il s'agit de l'exécutable.

in.ftpd : c'est la ligne de commandes. Dans ce cas, c'est le nom du programme sans option.

30.2.3 Invoquer un service "TCP wrapper" d'`inetd`.

Avec cette troisième méthode, `inetd` lance le service à notre place sous la commande d'interface (ou *wrapper command*) `tcpd`. Donc, cette méthode est

très proche de celle vue à la section 30.2.2. Cependant, il y a un petit changement dans l'entrée du fichier /etc/inet.conf :

```
ftp    stream tcp   nowait root  /usr/sbin/tcpd /usr/sbin/in.ftpd
( ftp    stream tcp   nowait root  /usr/sbin/tcpd /usr/sbin/in.wuftpd )
```

Ensuite, il faut redémarrer le service inetd comme nous l'avons fait à la sous-section 30.2.2. La modification effectuée sur la ligne de /etc/inet.conf permet à tcpd d'invoquer in.ftpd (ou in.wuftpd) au nom d'inetd. La commande tcpd effectue divers tests sur la connexion entrante afin de décider si cette dernière est de confiance. tcpd vérifie de quel hôte provient la connexion et compare ce dernier aux entrées du fichier /etc/hosts.allow et /etc/hosts.deny. Elle peut refuser des connexions de certains hôtes, ce qui donne un contrôle d'accès fin aux services.

Considérons l'entrée de /etc/inetd.conf et celle du fichier /etc/hosts.allow :

```
in.ftpd : LOCAL, .my.domain
(in.wuftpd : LOCAL, .my.domain)
```

ainsi que celle du fichier /etc/hosts.deny :

```
in.ftpd : ALL
( in.wuftpd : ALL )
```

Dans cet exemple, seront refusées les connexions de toutes les machines dont le nom d'hôte ne se termine pas par .my.domain. Seront acceptées les connexions de la machine locale [celle sur laquelle inetd est excécuté]. A ce stade, il est utile d'essayer d'établir une connexion ftp de différentes machines pour vérifier le contrôle d'accès. Une description complète du format de /etc/hosts.allow et /etc/hosts.deny est donnée dans hosts_access(5). Voici un autre exemple relatif à /etc/hosts.deny :

```
ALL : .snake.oil.com, 146.168.160.0/255.255.240.0
```

Dans ce cas, l'accès sera refusé pour tous les services (ALL) à toutes les machines faisant partie du réseau 146.168.160.0 et à toutes les machines appartenant au domaine .snake.oil.com.

30.2.4 Les conventions de distribution.

Remarquez que les méthodes décrites dans la section précédente ne peuvent être appliquées simultanément. Si un service est déjà activé par une méthode, il ne pourra être démarré dans un autre mode et en cas d'essai, le message d'erreur sera probablement "port in use". Votre distribution est déjà orientée pour exécuter un service soit sous inetd soit en mode autonome. Dans le premier cas, vous trouverez une ligne dans /etc/inetd.conf ; dans l'autre, il y aura un script /etc/init.d/<service> (ou /etc/rc.d/init.d/<service>) susceptible d'être démarré (start) ou arrêté (stop). Typiquement, au lieu d'un script /etc/init.d/ftpd, vous trouverez les scripts /etc/init.d/httpd et

`/etc/init.d/named`.

`named` est étudié au chapitre 41. Notez qu'il y a `toujours` un script `/etc/init.d/inet`.

30.3 Explications de services divers.

Tous les services sont potentiellement des trous de sécurité. Ne tentez pas le diable : désactivez-les tous en commentant toutes les lignes dans le fichier `/etc/inetd.conf`.

Un fichier `/etc/inetd.conf` typique (sans lignes de commentaires) ressemble à ceci :

```
ftp     stream  tcp     nowait  root        /usr/sbin/tcpd      in.ftpd -l -a
telnet  stream  tcp     nowait  root        /usr/sbin/tcpd      in.telnetd
shell   stream  tcp     nowait  root        /usr/sbin/tcpd      in.rshd
login   stream  tcp     nowait  root        /usr/sbin/tcpd      in.rlogind
talk    dgram   udp     wait    nobody.tty  /usr/sbin/tcpd      in.talkd
ntalk   dgram   udp     wait    nobody.tty  /usr/sbin/tcpd      in.ntalkd
pop-3   stream  tcp     nowait  root        /usr/sbin/tcpd      ipop3d
imap    stream  tcp     nowait  root        /usr/sbin/tcpd      imapd
uucp    stream  tcp     nowait  uucp        /usr/sbin/tcpd      /usr/sbin/uucico -l
tftp    dgram   udp     wait    root        /usr/sbin/tcpd      in.tftpd
bootps  dgram   udp     wait    root        /usr/sbin/tcpd      bootpd
finger  stream  tcp     nowait  nobody      /usr/sbin/tcpd      in.fingerd
auth    stream  tcp     wait    root        /usr/sbin/in.identd in.identd -e -o
```

Les services mentionnés ci-dessus ont les fonctions suivantes (le numéro de port est indiqué entre parenthèses) :

ftp (21) Protocole de transfert de fichiers (*File Transfer Protocol*).

telnet (23) connexion par telnet ;

shell (514) `rsh` ou service d'exécution du script de shell distant.

login (513) `rlogin` ou service de connexion à distance.

talk (517) aussi appelé **ntalk** : utilitaire de communication pour l'utilisateur.

pop-3 (110) service de récupération du courriel par *Post Office Protocol* – en majorité les utilisateurs récupèrent leurs courriels via leur fournisseur d'accès internet.

imap (143) Protocole d'accès au courriel par internet (*Internet Mail Access Protocol*) –un service plus sophistiqué mais bien moins sécurisé que POP.

uucp (540) copie UNIX-à-UNIX opérant via TCP.

tftp (69) service FTP commun utilisé, par exemple, par les stations de travail sans disque pour récupérer une image du noyau.

bootpd (67) service de configuration BOOTP IP pour les réseaux LAN qui requièrent une attribution IP automatique.

finger (79) service de requête pour utilisateur.

auth (113) un service qui détermine le propriétaire d'une connexion TCP particulière. Si vous avez une machine avec de nombreux utilisateurs, les administrateurs des autres machines peuvent voir quels utilisateurs sont connectés à leur(s) machine(s) via la vôtre. Pour disposer d'une trace, certains serveurs TCP et IRC demandent aux clients qui se connectent d'exécuter ce service. Désactivez ce service si votre machine ne supporte pas les connexions par shell pour de multiple utilisateurs.

30.4 L'alternative xinetd.

Au lieu d'utiliser la combinaison traditionnelle inetd + tcpd, de nombreuses distributions –dont RedHat depuis la version 7.0– ont adopté xinetd. xinetd regroupe les caractéristiques de tcpd et d'inetd en un seul logiciel. Le paquet xinetd consiste en un fichier de configuration de haut-niveau, /etc/xinetd.conf, un exécutable, /usr/sbin/xinetd et un fichier de configuration pour chaque service sous le répertoire /etc/xinetd.d/. *Cette configuration permet le contrôle d'un paquet comme ftpd (en terme de configuration)* via *un fichier séparé qui lui est propre.*

30.5 Fichiers de configuration.

Le fichier de configuration /etc/xinetd.conf se présente comme ceci :

```
defaults
{
   instances        = 60
   log_type         = SYSLOG authpriv
   log_on_success   = HOST PID
   log_on_failure   = HOST RECORD
}
includedir /etc/xinetd.d
```

Ce fichier indique que xinetd effectue les opérations suivantes : il limite le nombre de connexions simultanées pour chaque service à 60 ; il journalise avec syslog en exploitant son canal authpriv ; il journalise le contenu d'HOST et de PID pour chaque connexion réussie ; il journalise le contenu d'HOST (et aussi l'information RECORD relative à chaque tentative de connexion) pour chaque connexion qui a échoué. Cependant, /etc/xinetd.conf ne donne pas de renseignements cruciaux.

La dernière ligne impose de regarder dans /etc/xinetd.d pour trouver davantage de fichiers spécifiques aux services. Notre service FTP présente un fichier /etc/xinetd.d/wu-ftpd contenant :

```
service ftp
{
   socket_type      = stream
   server           = /usr/sbin/in.ftpd
   server_args      = -l -a
```

```
    wait            = no
    user            = root
    log_on_success  += DURATION USERID
    log_on_failure  += USERID
    nice            = 10
}
```

Ce fichier est similaire à la ligne associée à `ftp` dans le fichier `/etc/inetd.conf` vu à la sous-section 30.2.2, bien qu'il y ait plus d'information. Ce fichier impose les actions suivantes : écouter avec une connexion TCP en flux (`stream` *TCP socket*) ; lancer l'exécutable `/usr/sbin/in.ftpd` lors d'une connexion réussie ; passer l'argument `-l -a` à `in.ftpd` en ligne de commande (voir `ftpd`(8)) ; ne jamais attendre après `in.ftpd` pour sortir avant d'accepter la connexion entrante suivante ; exécuter `in.ftpd` en tant qu'utilisateur `root` ; journaliser `DURATION` et `USERID` lors des connexions réussies ; journaliser `USERID` lors de connexions ayant échoué ; et finalement régler, vis-à-vis de la CPU, la "douceur" (`nice`) du processus en exécutant `in.ftpd` avec une priorité de 10.

30.5.1 Limitations d'accès.

Les options de sécurité de `xinetd` offre beaucoup de souplesse. La plus importante est `only_from` permettant de restreindre les hôtes distants autorisés à recourir à un service. L'usage le plus extrême qu'un administrateur puisse en faire est d'ajouter `only_from 127.0.0.1` au fichier de configuration `/etc/xinetd.conf` :

```
defaults
{
    only_from        = 127.0.0.1 mamachine.local.domain
.
.
.
```

ce qui ne permet à aucune machine distante de faire le moindre usage du service `xinetd`. Une autre possibilité consiste à ajouter `only_from` à un, plusieurs ou tous les fichiers associés aux services et contenus dans `/etc/xinetd.d/` de manière à restreindre l'accès de manière fine pour chaque service.

L'option `only_from` peut fonctionner sur une gamme d'adresses IP sous la forme *nnn.nnn.nnn.nnn/bits*, et aussi sur base des noms de domaines. Par exemple, la forme suivante :

```
only_from  =  127.0.0.1  192.168.128.0/17  .somewhere.friendly.com
```

permet un accès à toutes les machines dont le nom d'hôte se termine par `.somewhere.friendly.com`.

Enfin, il existe une option `no_access` qui travaillant de manière identique à `only_from`, indique les hôtes et/ou les adresses IP des machines qui *n'*ont *pas* l'autorisation de se connecter :

```
no_access   =   .snake.oil.net
```

30.6 Sécurité.

On pourrait penser qu'utiliser `/etc/hosts.deny` (ou `only_from =`) pour interdire l'accès à toutes les machines distantes est suffisant pour sécuriser un système. Ceci *n'*est *pas* vrai : même un utilisateur local qui n'est autorisé qu'à accéder à un service local est potentiellement un trou de sécurité. En effet, un service présente usuellement des privilèges plus élévés que ceux de l'utilisateur. Il est donc indiqué de supprimer tous les services qui ne sont pas absolument nécessaires. Concernant les machines sur l'internet, n'hésitez pas à couper le dernier service qui n'est pas utilisé, voire même à désinstaller `inetd` (ou `xinetd`) entièrement.

Reportez-vous au chapitre 45.

Chapitre 31

exim et sendmail.

Le présent chapitre explique comment faire de votre station LINUX un serveur de messagerie. Une discussion sur le processus de livraison du courriel et sa récupération via POP et IMAP est présentée.

31.1 Introduction.

`exim` et `sendmail` sont des *MTA*s (*mail transfer agents* ou logiciels de transfert du courriel). Un MTA est démon qui, sur le port 25, écoute les connexions entrantes du courriel, gère les spoules[1] [voir la page 239 à propos du spoule en termes généraux] et place les messages dans une file (dans le cas d'`exim`, il s'agit du répertoire `/var/spool/exim/input` ; dans le cas de `sendmail`, il s'agit du répertoire `/var/spool/mqueue`). Ensuite, le MTA renvoie les messages à un autre MTA ou les livre localement dans la boîte électronique d'un utilisateur. Donc, le MTA est le logiciel par excellence qui manipule le courriel et assure son routage ainsi que les livraisons de messages. A la section 11.2, nous avons vu comment se connecter manuellement à un MTA via `telnet`. Dans cet exemple-là, `sendmail` (version 8.9.3) était le MTA fonctionnant sur la machine `mail.cranzgot.co.za`.

`sendmail` est le premier MTA apparu sous UNIX et il est de ce fait populaire. Il est probablement nécessaire de savoir le configurer car de nombreuses entreprises l'emploient.[2] Cependant, `exim` –qui est beaucoup plus aisé à configurer– est tout-à-fait apte à remplacer `sendmail`. Il y a au moins trois MTAs préférables à `sendmail`. Nous nous attacherons à décrire le minimum à propos de `sendmail` et à étudier en détail `exim`.

31.1.1 Fonctionnement du courriel.

Avant d'entrer dans la configuration d'un MTA, il est nécessaire de revoir les bases de la livraison du courriel et la manipulation des informations DNS

[1] NdT : on pourrait éventuellement traduire le terme "spoule" comme une réserve (au sens d'un magasin, où on enregistre des données).

[2] NdT : `sendmail` –programme monolithique– a été développé à une époque où l'internet était peu agressif. Ses modifications progressives en ont fait un programme peu flexible et difficile à configurer. Il existe de bien meilleurs MTAs. Par exemple, `exim` (discuté ici), `qmail` ou encore `Postfix` –développé par W. Venema– et qui est très bien adapté par sa modularité au caractère agressif de l'internet actuel (voir http ://www.postfix.org).

de type MX (voir la sous-section 28.7.1). La transmission du courriel (envoyé par un client courriel interactif) aboutit à la station d'un utilisateur distant selon la séquence suivante :

1. Un utilisateur a configuré son client courriel (Thunderbird, Mozilla, Outlook Express, Netscape, etc.) de manière à utiliser un *serveur (host) SMTP* particulier[3] (pour le courriel sortant de son ordinateur ; ce serveur est parfois appelé *passerelle SMTP*) et un *serveur (host) POP* ou *IMAP* pour le courriel entrant.
2. L'utilisateur compose un message à l'attention de rrabbit@toonland.net, et clique sur le bouton "Envoyer".
3. Le client de courriel initie une connexion sortante SMTP s'adressant au port 25 du serveur SMTP. Un MTA fonctionnant sur l'hôte SMTP répond à la requête arrivant sur le port 25. Le client de courriel utilise le protocole SMTP exactement comme cela a été vu à la section 11.2, même si cela se fait de manière automatisée. Il complète le champ d'adresse du destinataire avec rrabbit@toonland.net et transfère d'une part un en-tête correctement composé (c'est du moins ce qu'on attend de lui) et d'autre part, un corps de message au MTA. Le client de courriel termine alors sa connexion et rapporte d'éventuelles erreurs.
4. Le MTA transfère le message dans une file d'attente qui est périodiquement traitée selon un schéma donné.
5. Lorsque le schéma de traitement l'autorise, le MTA analyse l'adresse du destinataire rrabbit@toonland.net. Il élimine la partie "domaine" de l'adresse électronique, c'est-à-dire tout ce qui se trouve derrière le signe @. Il entreprend une requête DNS de type MX (lookupindexiiMX recordDNS) relative au domaine toonland.net.
   La résolution DNS relative à toonland.net suit la méthode exposée à la section 28.2. En bref et en première approximation, cela signifie que le MTA cherche le serveur de noms qui est autoritaire pour le domaine toonland.net. Il interroge ce serveur de noms pour l'enregistrement MX du domaine toonland.net. Le serveur de noms retourne un nom d'hôte (c'est-à-dire de serveur). Supposons que ce soit mail.toonland.net avec l'adresse IP correspondante. Admettons qu'il s'agisse de 197.21.135.82 (la section 28.7.1 montre comment effectuer une recherche manuelle des enregistrements MX). [Le chapitre 41 vous indique comment configurer un serveur de noms pour retourner un enregistrement MX].
6. Le MTA entreprend une connexion SMTP sur le port 25 de 197.21.135.82. Un autre MTA fonctionnant sur mail.toonland.net répond à la requête. L'adresse de destination, l'en-tête et le corps du message sont transmis en utilisant le protocole SMTP. Le premier MTA clôt la connexion.
7. Le MTA en cours sur mail.toonland.net analyse l'adresse de destination rabbit@toonland.net. Il reconnaît toonland.net comme étant un domaine pour lequel il héberge du courriel (c'est-à-dire le *domaine local*). Il identifie rabbit comme un nom d'utilisateur de son propre fichier /etc/passwd.

[3] NdT : pour fixer les idées, disons un prestataire de service courriel comme skynet, wanadoo, etc.

8. Le MTA en cours d'éxecution sur `mail.toonland.net` ajoute le message à la boîte de courrier électronique (soit `/var/spool/mail/rrabbit`, soit `/home/rrabbit/Maildir/`). *La livraison est à présent terminée. La manière selon laquelle le message est transféré de la boîte électronique de `mail.toonland.net` à la station de travail de Monsieur Rabbit* **n'***est* **pas** *du ressort du MTA et la procédure de transfert ne se produit pas via le protocol SMTP.*
9. Sur sa station de travail, Monsieur Rabbit devrait avoir configuré son client courriel, de manière à utiliser un programme POP ou IMAP de l'hôte `mail.tooland.net` afin de relever son courriel. Les services POP ou IMAP de `mail.tooland.net` écoutent sur les ports 110 et 143, respectivement.
10. Le client courriel de Monsieur Rabbit établit une connexion sur le port 110 (ou 143) et communique en utilisant le protocol POP ou IMAP. Le service POP (ou IMAP) doit assurer la transmission du message vers la boîte électronique se trouvant sur la station de travail de Monsieur Rabbit. Une fois le courrier transmis, il est effacé sur le serveur.
11. Le client courriel de Monsieur Rabbit enregistre le message sur la station de travail selon ses propres directives et marque le message comme étant nouveau.

31.1.2 Configuration d'un serveur POP/IMAP.

POP et IMAP sont invoqués par `inetd` ou `xinetd` –voir le chapitre 30. Hormis en ce qui concerne la limitation de la gamme de clients autorisés à se connecter pour des raisons de sécurité, aucune configuration particulière n'est requise. Les connexions clientes s'authentifient d'elles-mêmes en utilisant l'identifiant de connexion et le mot de passe UNIX. Il existe des paquets spécialisés de POP et d'IMAP pour supporter les différents types de boîtes électroniques (telle que Maildir).

31.1.3 Pourquoi `exim` ?

La page d'entrée du site http ://www.exim.org fournit un argumentaire exhaustif. Disons seulement ici qu'`exim` est le MTA le plus simple à configurer. En outre, si vous songez à la manière dont le courriel fonctionne, sachez que le fichier de configuration d'`exim` fonctionne similairement. Il est très facile de personnaliser `exim` pour réaliser des choses surprenantes. Le paquet est organisé de manière claire, logique et intuitive, à l'inverse du fichier `sendmail.cf` de `sendmail` qui est souvent considéré comme confus et malcommode. `exim` a été conçu en intégrant des contraintes de sécurité, bien que de nombreux administrateurs soulignent que `postfix` et `qmail` ont le dernier mot en matière de sécurité.

31.2 Contenu du paquet `exim`.

Vous pouvez obtenir `exim` sous forme d'un paquet `.rpm` ou `.deb`. Après l'installation, le fichier `/usr/share/doc/exim-?.??/doc/spec.txt` [ou /usr/doc] contient la documentation complète d'`exim`. Il y a aussi une version HTML sur la page web d'`exim`, alors que la page de man ne contient que les informations

à propos de la ligne de commande. exim permet un remplacement complet de sendmail dans la mesure où, pour chaque commande critique de sendmail, il existe une commande d'exim du même nom qui prend les mêmes options. Aussi les scripts faisant appel à ces commandes sont-ils insensibles au remplacement de sendmail par exim. Voici les fichiers-clés :

```
/etc/aliases
/usr/bin/mailq
/usr/bin/newaliases
/usr/bin/rmail
/usr/lib/sendmail
/usr/sbin/sendmail
```

Enfin, il y a le binaire d'exim lui-même, /usr/sbin/exim et les fichiers de configuration /etc/exim/config, /etc/exim/exim.conf, selon votre distribution LINUX. Vous trouverez les scripts de démarrage/arrêt : /etc/init.d/exim [ou /etc/rc.d/init.d/exim].

31.3 Fichier de configuration d'exim.

En guide d'exemple préliminaire, nous allons créer un simple serveur agissant comme réserve (ou magasin) de courriels sur une station de travail personnelle, cericon.cranzgot.co.za.

Les applications clientes (en particulier celles qui ne sont pas UNIX) sont usuellement configurées pour se connecter à un MTA fonctionnant sur une machine distante. Cependant, l'utilisation d'un hôte SMTP distant peut s'avérer ennuyeux si l'hôte ou le réseau se trouve en dysfonctionnement. Le fait d'exécuter exim sur une station de travail locale permet à toutes les applications d'utiliser localhost comme étant leur propre passerelle SMTP, c'est-à-dire qu'exim utilise la mise dans une file et les tentatives périodiques.

Voici une configuration-type. La différence entre celle-ci et la configuration d'un serveur de courriel complet est minime.

```
###################### MAIN CONFIGURATION SETTINGS ######################
log_subject
errors_address = postmaster
freeze_tell_mailmaster = yes
queuelistrequires_admin = false
prod_requires_admin = false
trusted_users = psheer
local_domains = localhost : ${primary_hostname}
never_users = root
#relay_domains = my.equivalent.domains : more.equivalent.domains
host_accept_relay = localhost : *.cranzgot.co.za : 192.168.0.0/16
exim_user = mail
exim_group = mail
end
```

```
###################### TRANSPORTS CONFIGURATION ######################
remote_smtp :
   driver = smtp
   hosts = 192.168.2.1
   host_override
local_delivery :
   driver = appendfile
   file = /var/spool/mail/${local_part}
   delivery_date_add
   envelope_to_add
   return_path_add
   group = mail
   mode_fail_narrower =
   mode = 0660
end

###################### DIRECTORS CONFIGURATION ########################
localuser :
   driver = localuser
   transport = local_delivery
end

###################### ROUTERS CONFIGURATION ##########################
lookuphost :
   driver = lookuphost
   transport = remote_smtp
literal :
   driver = ipliteral
   transport = remote_smtp
end

###################### RETRY CONFIGURATION ############################
*                     *           F,2h,15m; G,16h,1h,1.5; F,4d,8h
end

###################### REWRITE CONFIGURATION ##########################
*@cericon.cranzgot.co.za  psheer@icon.co.za
```

31.3.1 Réglages globaux.

Le fichier de configuration d'`exim` est divisé en six sections logiques séparées par le mot-clé `end`. Le début du fichier ou section `MAIN` contient les paramètres globaux. En voici la signification :

log_subject indique à `exim` de journaliser le sujet dans le fichier de journalisation du courriel. Par exemple, `T="I LOVE YOU"` sera ajouté au fichier de journalisation.

errors_address désigne l'adresse courriel où les erreurs doivent être envoyées. Ce que vous envoyez ici n'a pas d'importance parce que tout le courrier

électronique sera réécrit à psheer@icon.co.za, comme nous le verrons ci-après.

freeze_tell_mailmaster indique l'errors_adress à propos des messages "gelés". Ces messages sont ceux qui ne sont pas livrés (par exemple, en raison d'un problème de droits, ou alors, il peut s'agir d'un message qui a échoué car l'adresse de retour est non-valide). Ils sont marqués comme inutiles, et ne seront plus traités ultérieurement. Notez que les messages "gelés" indiquent parfois qu'il y a une mauvaise configuration de votre système ou de la messagerie.

local_domains chaque message de courriel reçu est traité selon un de ces deux cas : soit en livraison locale, soit en livraison distante. Une livraison locale concerne un ou plusieurs utilisateurs mais se produit sur la machine locale. Une livraison distante est effectuée *via* l'internet. L'option local_domains distingue ces deux cas. Ainsi, compte tenu de la ligne indiquée dans le fichier de configuration ci-dessus, un message destiné à psheer@localhost ou à psheer@cericon.cranzgot.co.za est local, tandis qu'un message adressé à psheer@nimporteou.co.za est distant. Observez bien que la liste est terminée par un double point.

never_users pour des raisons de sécurité, les utilisateurs ne deviennent jamais root.

exim_group spécifie le groupe dans lequel fonctionne exim.

> *Il est important de comprendre que les options host_accept_relay et relay_domains concernent la sécurité.*

host_accept_relay cette option définit le nombre de machines qui sont autorisées à utiliser cericon.cranzgot.co.za comme *relai*. Un relai est un serveur qui envoie du courriel au nom d'une autre machine : c'est-à-dire que nous agissons comme relai chaque fois que nous traitons un message de courriel qui ne provient pas de notre machine ou bien qui n'est pas destiné à un utilisateur de notre machine.
Nous ne voudrons jamais relayer à partir d'un serveur qui ne soit pas de confiance. Pourquoi ? Parce qu'il pourrait, par exemple, permettre à des personnes d'envoyer 100.000 messages à 100.000 adresses différentes, chacun de ces messages contenant nos adresses dans l'en-tête.
host_accept_relay indique une liste d'hôtes de confiance, qui sont autorisés à envoyer des messages *via* notre machine. A nouveau, veuillez constater que la liste est terminée par un double point. En l'occurrence, nous n'avons même pas besoin de mettre les adresses des autres machines de notre LAN, sauf, si nous les jugeons comme étant de confiance.

relay_domains relay_domains fournit une condition supplémentaire pour qu'un hôte quelconque soit autorisé à nous utiliser comme relai. Considérons que nous devions faire un serveur de sauvegarde du courriel pour un domaine particulier. Le courriel vers ce domaine ne provient pas de notre machine. Il ne nous est pas destiné non plus. Il faut néanmoins permettre son acheminement si, et seulement si, l'adresse de destination correspond aux domaines pour lesquels nous effectuons une sauvegarde du courriel. Nous indiquons de tels domaines sous le paramètre relay_domains.

31.3.2 Transports.

La section relative au transport suit immédiatement celle contenant les options de configuration principales. Elle définit diverses *méthodes* d'acheminement du courriel. Nous nous référerons à ces méthodes plus tard, dans le fichier de configuration. `telnet` sur le port 25 a *transporté* un message par SMTP. Il existe une autre méthode de transport qui consiste à ajouter un message à la fin de la liste du courriel. Ces méthodes sont représentées, respectivement, par `remote_smtp :` et `local_delivery :`.

remote_smtp : cette méthode de transport présente les sous-options suivantes :

driver la méthode réelle d'acheminement.L'option `driver =` indique toujours le type de transport, le redirecteur ou le routage.

hosts_override et **hosts** En utilisant ces deux options simultanément, on passe outre toute liste d'hôtes qui a été reprise par des requêtes DNS MX. Par "liste d'hôtes", il faut comprendre les machines établies à partir de l'adresse courriel du destinataire, auxquelles nous pourrions faire des livraisons par SMTP, mais que nous n'utilisons pas.

local_delivery : ce type de transport présente les options suivantes :

driver la méthode réelle de acheminement. L'option `driver =` indique toujours le type de transport, le redirecteur ou le routage.

file le fichier à ajouter au message de courriel. `${local_part}` est remplacé par tout ce qui se trouve avant le caractère `@` de l'adresse du destinataire.

delivery_date_add, enveloppe_to_add, return_path_add sont diverses indications à placer dans les en-têtes.

group, mode_fail_narrower, mode divers réglages des droits.

A ce stade, il devrait être évident, que ces deux méthodes sont effectivement utilisées. En ce qui concerne le MTA, les deux seules choses qui arrivent à un message sont (i) son envoi *via* SMTP à un autre serveur ou (ii) son ajout à la file de courrier.

31.3.3 Redirecteurs.

Si un message arrive et qu'il est listé dans `local_domains`, `exim` tente une livraison locale. Ceci signifie qu'`exim` fonctionne avec une liste de *redirecteurs* jusqu'à ce qu'il en trouve un qui n'échoue pas. Le seul redirecteur listé dans notre configuration est `localuser :` avec `local_delivery` comme transport. Ici, le mécanisme est donc simple : les messages ayant des adresses de destination listées sous `local_domain` sont ajoutés à la boîte électronique de l'utilisateur.

31.3.4 Routeurs.

Si un message arrive et qu'il n'est pas listé dans `local_domains`, `exim` tente une livraison distante. De manière analogue à ce qui est décrit dans la sous-section 31.3.3, `exim` utilise une liste de routeurs jusqu'à ce qu'il en trouve un qui n'échoue pas.

En l'occurrence, deux routeurs sont mentionnés. Le premier vaut pour les adresses usuelles de courriel. Il se sert du pilote `lookuphost` qui effectue une requête DNS MX sur la partie "domaine" de l'adresse électronique (c'est-à-dire, tout ce qui se trouve après `@`). Les enregistrements MX trouvés sont passés au transport `remote_smtp` (dans notre configuration, ils sont ignorés). `lookuphost` échouera si la partie "domaine" de l'adresse électronique est une adresse IP entre crochets.

Le second routeur utilise le pilote `ipliteral`. Il envoye des messages directement à une adresse IP dans le cas d'adresse IP entre crochets comme par exemple : `root@[111.1.1.1]`.

Un routeur dirige le courriel vers un autre hôte.

31.4 Un véritable serveur de courriel.

Le fichier de configuration d'un véritable serveur de courrier électronique diffère peu de celui vu à la section 31.3. Voici le fichier de configuration par défaut associé à `exim-3.16` :

```
###################### MAIN CONFIGURATION SETTING ####################
# primary_hostname =
# qualify_domain =
# qualify_recipient =
# local_domains =
nerver_users = root
# hosts_accept_relay = localhost
# hosts_accept_relay = my.friends.host : 131.111.0.0/16
# relay_domains = my.equivalent.domains : more.equivalent.domains
host_lookup = 0.0.0.0/0
# receiver_unqualified_hosts =
# sender_unqualified_hosts =
rbl_domains = rbl.maps.vix.com
no_rbl_reject_recipients
sender_reject = "*@*.sex*.net :*@sex*.net"
host_reject = "open-relay.spamming-site.com"
rbl_warn_header
# rbl_domains = rbl.maps.vix.com :dul.maps.vix.com :relays.orbs.org
# percent_hack_domains = *
end
###################### TRANSPORTS CONFIGURATION ######################
remote_smtp :
   driver = smtp
# procmail transport goes here <---
local_delivery :
   driver = appendfile
   file = /var/spool/mail/${local_part}
   delivery_date_add
   envelope_to_add
   return_path_add
```

```
   group = mail
   mode = 0660
address_pipe :
   driver = pipe
   return_output
address_file :
   driver = appendfile
   delivery_date_add
   envelope_to_add
   return_path_add
address_reply :
   driver = autoreply
end
##################### DIRECTORS CONFIGURATION ######################
# routers because of a "self=local" (not used in this configuration).
system_aliases :
   driver = aliasfile
   file = /etc/aliases
   search_type = lsearch
   user = mail
   group = mail
   file_transport = address_file
   pipe_transport = address_pipe
userforward :
   driver = forwardfile
   file = .forward
   no_verify
   no_expn
   check_ancestor
# filter
   file_transport = address_file
   pipe_transport = address_pipe
   reply_transport = address_reply
# procmail director goes here <---
localuser :
   driver = localuser
   transport = local_delivery
end
##################### ROUTERS CONFIGURATION ########################
# widen_domains = "sales.mycompany.com :mycompany.com"
lookuphost
   driver = lookuphost
   transport = remote_smtp
# widen_domains =
literal :
   driver = ipliteral
   transport = remote_smtp
end
```

```
###################### RETRY CONFIGURATION ##############################
*                    *           F,2h,15m; G,16h,1h,1.5; F,4d,8h
end
#########################################################################
```

Concernant le support de procmail (voir procmail(1), procmailrc(6) et procmailex(5)), ajoutez simplement :

```
procmail :
   driver = pipe
   command = "/usr/bin/procmail -Y -d ${local_part}"
```

après votre transport remote_smtp, et aussi :

```
procmail :
   driver = localuser
   transport = procmail
   require_files = /usr/bin/procmail
```

à la suite de votre redirecteur user_forward.

31.5 Commandes de shell pour l'administration d'exim.

Comme avec les autres démons, vous pouvez arrêter exim, le démarrer, et le forcer à relire son fichier de configuration avec :

```
/etc/init.d/exim stop
/etc/init.d/exim start
/etc/init.d/exim reload
```

Vous devriez toujours effectuer reload lorsque vous désirez que le fichier de configuration soit relu. Le script de démarrage (start) n'exécute en réalité que la commande exim -bd -q30m, qui indique à exim de démarrer en mode autonome(*standalone*), pour écouter sur le port25 et exécuter un runq toutes les 30 minutes (voir ci-dessous).

Pour faire en sorte qu'exim [mais ceci vaut pour beaucoup d'autres MTA] effectue une boucle sur la file des messages en suspens et, qu'il prenne en considération ceux qui doivent être livrés, effectuez :

```
runq
```

ce qui revient à lancer exim -q.

Pour afficher le courrier en attente d'acheminement, utilisez :

```
mailq
```

qui correspond à exim -bp.

Pour forcer la livraison de n'importe quel courriel de la file, utilisez :

```
exim -qf
```

et pour forcer sur les messages même gelés, exécutez :

```
exim -qff
```

La commande pour effacer un message de la file d'attente est :

```
exim -Mrm <message-id>
```

La page de man exim(8) contient une description exhaustive des commandes d'exim et de leurs options. Cependant, ce qui est décrit ci-dessus représente les commandes et les options que vous utiliserez le plus.

31.6 La file d'attente.

Il est souvent utile de vérifier le répertoire de la file d'attente /var/spool/exim/input/ pour rechercher des messages, ou simplement pour jeter un coup d'oeil à ce qui s'y passe. La simple session suivante :

```
[root@cericon]# mailq
Om    320 14Epss-0008DY-00 <psheer@icon.co.za>
          freddy@elmstreet.org

Om    304 14Ept8-0008Dg-00 <psheer@icon.co.za>
          igor@ghosbusters.com

[root@cericon]# ls -l /var/spool/exim/input/
total 16
-rw-------      1 root       root         25 Jan 6 11 :43 14 Epss-0008DY-00-D
-rw-------      1 root       root        550 Jan 6 11 :43 14 Epss-0008DY-00H
-rw-------      1 root       root         35 Jan 6 11 :43 14 Ept8-0008Dg-00-D
-rw-------      1 root       root        530 Jan 6 11 :43 14 Ept8-0008Dg-00-H
```

montre clairement que les deux messages sont en attente de livraison. Les fichiers se terminant par -H sont des en-têtes d'enveloppe (*envelope headers*) tandis que celles se terminant par -D sont des corps de message. Le document spec.txt vous montre comment interpréter le contenu des fichiers d'en-têtes.

N'ayez pas peur d'éliminer (rm) manuellement les fichiers de ce répertoire, mais faites-le par paires (supprimez *à la fois* l'en-tête *et* le corps). Assurez-vous qu'exim n'est pas en cours à ce moment-là. Pour ce qui concerne l'exemple précédent, les commandes :

```
[root@cericon]# exim -Mrm 14Epss-0008DY-00 14Ept-0008Dg-00
Message 14Epss-0008DY-00 has been removed
Message 14Ept8-0008Dg-00 has been removed
```

```
[root@cericon]# mailq
[root@cericon]#
```

fonctionnent même mieux.

31.7 /etc/aliases pour les adresses équivalentes.

Nous souhaitons parfois que certains messages arrivant à des adresses locales soient réellement acheminés vers d'autres adresses. Par exemple, tous les méls destinés à MAILER_DAEMON devraient être dirigés vers l'utilisateur postmaster ; ou alors, si un utilisateur possède deux comptes, peut-être ne veut-il lire que sur un seul des deux comptes.

Le fichier /etc/aliases permet de réaliser ces opérations. Ce fichier est devenu une institution. Cependant, vous pouvez constater qu'avec exim, la construction d'alias est aisée : vous pouvez spécifier une recherche directe (*lookup*) sur tout fichier associé au redirecteur system_aliases : pourvu que ce fichier possède un double-point comme délimiteur.

Un fichier par défaut de /etc/aliases pourrait contenir des informations comme celles de l'encadré ci-dessous.Vous devriez contrôler que le compte postmaster existe sur votre système et vérifier que vous pouvez lire, envoyer et recevoir des messages en tant qu'utilisateur postmaster.

```
# This is a combination of what I found in the Debian
# and RedHat distributions.

MAILER-DAEMON :             postmaster
abuse :                     postmaster
anonymous :                 postmaster
backup :                    postmaster
backup-reports :            postmaster
bin :                       postmaster
daemon :                    postmaster
decode :                    postmaster
dns :                       postmaster
dns-admin :                 postmaster
dumper :                    postmaster
fetchmail-daemon :          postmaster
games :                     postmaster
gnats                       postmaster
ingres :                    postmaster
info :                      postmaster
irc :                       postmaster
list :                      postmaster
listmaster :                postmaster
lp :                        postmaster
mail :                      postmaster
mailer-daemon :             postmaster
majordom :                  postmaster
man :                       postmaster
```

```
manager :                   postmaster
msql :                      postmaster
news                        postmaster
nobody :                    postmaster
operator :                  postmaster
postgres :                  postmaster
proxy :                     postmaster
root :                      postmaster
sync :                      postmaster
support :                   postmaster
sys :                       postmaster
system :                    postmaster
toor :                      postmaster
uucp :                      postmaster
warnings :                  postmaster
web-master :                postmaster
www-data :                  postmaster

# some users want their mail redirected
arny :                      mail@swartzneger.co.us
larry :                     lking@cnn.com
```

Vous pouvez éliminer bon nombre de ces alias. Pour qu'ils soient actifs, il faut que les services correspondant soient en cours de fonctionnement alors qu'en réalité sur votre système, ils ne sont peut-être même pas installés –par exemple, `games`, `ingres`, etc. Les alias permettent deux choses : (i) prévoir quels utilisateurs de courriel peuvent contacter l'administrateur, le cas échéant ; (ii) capter tout courrier envoyé par les démons du système (par exemple, l'adresse courriel de l'administrateur DNS est imposée par les fichiers de configuration DNS, comme cela est expliqué au chapitre 41).

Notez bien qu'un alias dans le fichier `/etc/aliases` ne doit pas avoir de compte sur le système : `larry` et `arny` ne doivent pas figurer dans `/etc/passwd`.

31.8 Liste de blocage en temps réel – combattre le "*spam*".

31.8.1 Qu'est-ce que le *spam* ?

Le *spam* (ou pourriel) est du courrier électronique non-sollicité [indésirable] envoyé en masse aux utilisateurs, avec un côté promotionnel. Ce type de courriel est envoyé de manière automatique à de nombreuses personnes n'ayant aucune relation avec l'expéditeur et ne sollicitant aucunement ces courriers. Il n'y a pratiquement aucune chance que le destinataire soit intéressé par le sujet du pourriel.

Généralement, le spam est du courriel envoyé à des adresses obtenues sans le consentement du propriétaire de la boîte électronique. Toute personne qui possède une boîte électronique depuis un certain temps recevra un jour ou l'autre des courriels dont le sujet est : `Gagnez rapidement de l'argent !`. La probabilité de recevoir du pourriel augmente avec la durée de vie d'une boîte

électronique. L'envoi de spam est très facile. Les spammeurs recherchent sur l'internet des serveurs de courrier qui permettent de relayer leurs messages. Ils leur envoyent 10.000 messages (disons) dont un relatif à la manière d'obtenir des photos d'adolescentes nues. Si vous faisiez cela, vous seriez un spammeur passible de poursuites judiciaires. Malheureusement, pourvu que le spammeur contrôle seulement un peu ce qu'il fait, un administrateur peu soupçonneux ne sera probablement pas à même de le traquer. Les spammeurs utilisent bien des astuces.

Notez que le spam n'est pas seulement du courriel indésirable. Souvent, le grand public pense que le courriel est analogue aux communications téléphoniques, par exemple. Dans le cas d'appels indésirables par téléphone, le caractère intrusif de l'appel n'a lieu que lorsque vous décrochez. Dans le cas du courriel, vous pouvez éliminer les messages. C'est pour cela que les filtres de messageries sont utiles, ils bloquent le courriel provenant d'un (ou plusieurs) expéditeur(s) sur base de son (ou leurs) adresse(s) (voir `procmailex`(5)). [Lorsque le spam envahit votre boîte, cela devient un problème que vous devez traiter. Faites attention : le fait de répondre à un spammeur en lui envoyant un message "S'il vous plaît ne m'envoyer plus ..." indique que vous êtes vulnérable à son spam et que vous ne recevez pas beaucoup de courrier.].

Quand le courriel devient-il intrusif ? Cela dépend beaucoup du volume de courrier reçu. *Par ailleurs, du fait que ce courriel provient souvent d'origines différentes, vous ne pouvez vous en protéger totalement à l'aide de filtres agissant sur le courriel de votre boîte.*

Typiquement, le pourriel commence par un "`Augmentez vos richesses dès à présent`", le spammeur s'empressant d'ajouter dans le corps du message :

> Ceci n'est pas du spam. Vous recevez ce mél parce que vous êtes sur une liste d'adresses que j'ai achetée. Vous avez accepté de recevoir des informations à propos des opportunités professionnelles. Si ce n'est pas le cas, acceptez nos excuses. Pour être SOUSTRAIT de cette liste, répondez simplement avec le mot REMOVE dans la case "sujet". Ainsi vous ne recevrez plus de courrier de notre part.

Il va sans dire qu'il ne faut pas répondre (certainement pas avec REMOVE dans le sujet) ; autrement, vous indiquerez aux spammeurs que votre adresse est valide.

31.8.2 Prévention élémentaire contre le *spam*.

Revenons au fichier de configuration d'`exim` et commençons par y ajouter :

```
headers_check_syntax
headers_sender_verify
sender_verify
receiver_verify
```

L'option `headers_check_syntax` amène `exim` à vérifier la syntaxe de tous les en-têtes des messages entrants. Les messages qui ne s'y conforment pas ne sont pas pris en compte. Les trois autres options contrôlent la syntaxe des champs :

– `Expéditeur :` (`Sender :`),

– Répondre à : (Reply-To :) et
– En provenance de : (From :)

mais aussi, les adresses manipulées par les commandes SMTP MAIL et RCPT pour déterminer s'il s'agit de véritables adresses de courriel.

Le raisonnement à la base de ce dispositif est que les spammeurs utilisent souvent des en-têtes mal formés pour tromper le MTA de manière à lui faire envoyer des messages que, d'ordinaire, il n'enverrait pas. L'auteur du texte original anglais précise qu'il ne sait pas exactement comment ces mesures sont utilisées par exim mais qu'il est un fait que les courriers sont rejetés au moment où l'échange SMTP débute.

31.8.3 Liste de blocage en temps réel.

Pour en apprendre plus à propos du spam, des serveurs bannis, des rapports à propos du spam et de l'usage du courriel en général, consultez MAPS (*Mail Abuse Prevention System LLC*) à l'adresse http ://www.mail-abuse.org/, et aussi *Open Relay Behavior-modification System* sur http ://www.orbs.org/. [Si ce site ne fonctionne pas, essayez http ://www.orbl.org/ et http ://www.ordb.org/]. Les *RBLs* (*Real-Time Blocking Lists* ou listes de blocage en temps réel) sont une idée assez ancienne incorporée dans exim comme un de ses constituants de base. En voici le principe. Les spammeurs doivent utiliser un serveur autorisant le relais de leur pourriel. L'adresse IP de ce serveur-relai est toujours communiquée en clair au MTA lors de la connexion. Le MTA peut alors vérifier cette IP en se servant d'une base de données contenant des adresses IP bannies définissant les serveurs-relais, et rendues publiques. Concernant exim, la liste se trouve sous rbl_domains. Si l'IP de connexion à exim est détectée dans rbl_domains, exim rejette la connexion. Vous pouvez activer cette fonctionnalité avec [le présent exemple provient de la page web d'exim] :

```
# reject messages whose sending host is in MAPS/RBL
# add warning to messages whose sending host is in ORBS
rbl_domains = blackholes.mail-abuse.org/reject : \
        dialups.mail-abuse.org/reject : \
        relays.mail-abuse.org/reject : \
        relays.orbs.org/warn
# checks all hosts other than those on internal network
rbl_hosts =!192.168.0.0/16 :0.0.0.0/0
# but allow mail to postmaster@my.dom.ain even from rejected hot
recipients_reject_except = postmaster@my.dom.ain
# change some logging actions (collect more data)
rbl_log_headers      # log headers of accepted RBLed messages
rbl_log_rcpt_count   # log recipient info of accepted RBLed messages
```

et ce, dans la section de configuration MAIN. Rappelez-vous également d'enlever la ligne no_rbl_reject_recipients, sans quoi, exim ne journalisera qu'un message d'alerte et ne refusera pas le message électronique illicite.

31.8.4 Administrateur de courriel et responsabilités des utilisateurs.

L'administrateur du courriel et les utilisateurs d'exim doivent être conscients de ceci :

- le spam est une plaie,
- et il est causé par des serveurs mal configurés,
- il est de la responsabilité de l'administrateur du courriel d'assurer les mesures adéquates pour combattre le spam,
- même en tant qu'utilisateur, vous devriez noter d'où proviennent les spams que vous recevez et en avertir l'administrateur,
- beaucoup d'administrateurs du courriel ne sont pas forcément conscients qu'il existe un problème. Rappelez-leur.

31.9 Sendmail.

Le fichier de configuration de sendmail est /etc/sendmail.cf. Ce format de fichier est un héritage des premiers serveurs UNIX et des fichiers de référence sous le répertoire /etc/mail/. Vous pouvez réaliser la plupart des opérations en modifiant divers fichiers sous /etc/mail/ sans être confronté à la complexité de /etc/sendmail.cf.

A l'instar de la plupart des MTAs fournis avec les distributions LINUX, le paquet sendmail fonctionne sans configuration préalable. Comme toujours, cependant, vous devrez ajouter une liste de serveurs-relais. Cette configuration se fait dans le fichier /etc/mail/access pour les versions supérieures ou égales à sendmail-8.10. Pour effectuer des relais à partir de votre propre machine et des machines de votre propre réseau 192.168.0.0/16, disons, aussi bien que du domaine *hosts*.trusted.com, vous devez au moins avoir :

```
localhost.localdomain    RELAY
localhost                RELAY
127.0.0.1                RELAY
192.168                  RELAY
trusted.com              RELAY
```

ce qui correspond exactement à l'option host_accept_relay utilisée dans exim.

Les domaines pour lesquels votre machine est un serveur de courriel de sauvegarde doivent être listés dans le fichier /etc/mail.relay-domains, chacun étant indiqué sur une ligne. Ceci est analogue à l'option relay_domains d'exim. Naturellement, les domaines pour lesquels sendmail reçoit du courriel doivent être indiqués. Ceci est analogue à l'option local_domains d'exim. Ces domaines sont listés dans /etc/mail/local-host-names, chacun sur une ligne.

Le fichier /etc/aliases est utilisé de la même manière par exim ou sendmail.

Ayant configuré sous /etc/mail/, vous devriez à présent effectuer make dans ce répertoire pour construire les tables de recherche (*lookup*) des fichiers. Vous devriez également lancer la commande newaliases chaque fois que vous modifiez /etc/aliases. Dans tous les cas de modification de la configuration, vous redémarrerez votre serveur avec la commande sendmail.

sendmail a reçu un grand nombre d'alertes de sécurité en son temps. Il est

donc impératif de toujours installer la version la plus récente. Notez bien que les versions plus anciennes ont un défaut extrêmement gênant : par défaut, leur configuration permet de faire du relai, d'où la nécessité d'une remise à jour.

Voici une ressource pour utiliser les ficelles de `sendmail`, The *Sendmail FAQ* sur http ://www.sendmail.org/faq/.

Chapitre 32

Démarrage, lilo et initrd.

`lilo` signifie `li`*nux* `lo`*ader* (système de chargement en mémoire du noyau Linux).[1] `LILO :` est l'invite que vous voyez juste après l'amorçage du système et, à partir de laquelle vous pouvez sélectionner le système d'exploitation avec lequel vous souhaitez travailler. A ce stade, il est possible de passer des options au noyau. Ce chapitre explique comment configurer `lilo`, utiliser les options du noyau et charger des systèmes *non-amorçables*.

Le paquet `lilo` lui-même contient les fichiers :

```
/boot/boot.b     /boot/message     /sbin/lilo
/boot/chain.b    /boot/os2_d.b     /usr/bin/keytab-lilo
/usr/share/doc/lilo-<version>
```

qu'il est peu intéressant d'analyser, si ce n'est qu'il faut savoir qu'on y trouve la documentation technique et celle destinée à l'utilisateur au cas où des informations pointues sont recherchées.

32.1 Utilisation.

Lorsque vous démarrez votre système Linux, l'invite `LILO :` –à laquelle vous passez des options d'amorçage (*boot*)– est affichée. Si vous presser la touche  ., vous verrez s'afficher une liste d'options. Le principe est de permettre l'accès à différentes installations Linux sur la même machine ou l'accès à des systèmes d'exploitation différents sur différentes partitions d'un même disque. Plus tard, vous pourrez jeter un coup d'oeil à `/proc/cmdline` pour voir quelles sont les options d'initialisation (ou d'amorçage) utilisées, y compris les options par défaut.

[1] NdT : il existe un système appelé `GRUB` (`GR`*and* `U`*nified* `B`*ootloader*) qui possède un mini-shell et des fonctionnalités plus grandes que celles de `lilo`.

32.2 Théorie.

32.2.1 Séquence de démarrage du noyau.

Un noyau UNIX, pour permettre l'amorçage, doit être chargé en mémoire alors qu'il se trouve sur le disque dur. Il doit aussi être exécuté. L'exécution entraîne la décompression du code lui-même et le lancement du programme [Les termes initialisation –ou amorçage (*boot* en anglais)– viennent du fait qu'un ordinateur ne peut commencer l'exécution d'un programme sans code, et que le code ne peut être placé en mémoire sans l'intervention d'un autre programme –NdT : d'où l'importance d'un chargeur de démarrage]. Après avoir été chargé, décompressé et installé dans la mémoire, le noyau initialise les périphériques matériels. Ensuite, il monte le système de fichiers sur une partition déterminée. C'est seulement à l'issue de cette étape que le noyau exécute `/sbin/init` pour démarrer le système d'exploitation UNIX.

32.2.2 Master boot record (MBR).

La vie de votre PC commence avec un petit programme implanté dans la ROM (BIOS) et qui charge le tout premier secteur du disque dur en mémoire. Ce secteur s'appelle le secteur d'amorçage (*boot sector*) du *master boot record* (MBR). Cette partie de code est de 512 octets et c'est sur elle que repose la mise en route du système d'exploitation. Dans le cas de LINUX, le secteur d'amorçage charge le fichier `/boot/map` qui contient une liste de la localisation précise des secteurs du disque que l'image du noyau LINUX (usuellement le fichier `/boot/vmlinuz`) occupe. Le secteur d'amorçage charge le contenu de chacun de ces secteurs de manière à reconstruire l'image du noyau dans la mémoire. Ensuite, la main est passée au noyau.

Vous pourriez vous demander comment il est possible de charger un fichier depuis un système de fichiers alors que ce dernier n'est pas monté. Retenons que la partition d'amorçage contient un programme à la fois simple et de petite taille qui ne supporte pas l'ensemble des systèmes de fichiers et périphériques où réside l'image du noyau. En fait, `lilo` ne doit pas supporter de système de fichiers pour accéder à un fichier donné, parce qu'il emploie une liste de secteurs que le fichier d'amorçage occupe et qu'il est programmé pour utiliser les *interruptions* du BIOS pour lire ces secteurs [le terme "*interruptions*" cité ici n'a rien à voir avec les interruptions matérielles. Il se rapporte à des fonctionnalités du BIOS mises à disposition du système d'exploitation. Les périphériques matériels peuvent prendre en compte des fonctions personnalisables du BIOS, support rudimentaire nécessaire au démarrage de ces périphériques. Ce support est à distinguer de celui fourni par les pilotes de périphériques, pilotes associés au noyau]. Si le fichier n'est jamais modifié, cette liste de secteurs ne changera pas. Telle est la manière dont `/boot/map` et `/boot/vmlinuz` sont chargés en mémoire.

32.2.3 Partitions de démarrage.

En plus de ce qui a été dit à propos du MBR, chaque partition primaire possède un secteur qui peut initialiser le système d'exploitation dans cette partition. Les partitions MS-DOS (Windows) possèdent cette caractéristique et exécute

ces secteurs d'amorçage de partition pour lancer l'installation de Windows dans une autre partition.

32.2.4 Limitations.

Les BIOS ont hérité de sévères limitations à cause du manque de prévoyance de leurs concepteurs. Premièrement, certains BIOS ne supportent pas plus d'un IDE [du moins selon la documentation de `lilo`].

La seconde limitation est importante. Comme expliqué à la sous-section 32.2.2, `lilo` utilise les fonctions du BIOS pour accéder aux IDEs, mais le BIOS d'un PC est souvent limité à n'accéder qu'aux 1024 premiers cylindres du disque. Donc, ce que `lilo` lit doit se trouver dans les 1024 premiers cylindres, autrement dit, les 500 premiers Mo de l'espace disque. Voici une liste des fichiers qui devraient se trouver sous cette limite :

1. `/boot/vmlinuz`
2. divers fichiers de `lilo` de type `/boot/*.b`
3. tout secteur d'amorçage de partition non-LINUX sur lequel vous voulez lancer un amorçage.

A contrario, une partition racine de LINUX peut résider n'importe où parce que le programme du secteur d'amorçage ne lit jamais cette partition sauf pour les fichiers mentionnés ci-dessus. Un scénario classique pour l'administrateur : le répertoire `/boot` sur une petite partition de moins de 500 Mo avec la partition `/` située au-delà de la limite des 500 Mo. Reportez-vous à la page 198.

Notez que les nouveaux BIOS "LBA" supportent plus que 512 Mo (en fait, jusqu'à 8 Go). L'auteur précise qu'il ne compte pas sur cette caractéristique.

32.3 `lilo.conf` et la commande `lilo`.

"Créer un `lilo`" consiste à lancer la commande `lilo` sous le compte `root` avec un fichier `/etc/lilo.conf` correctement configuré. Le fichier `lilo.conf` a très vraisemblablement été préconfiguré par votre distribution (vérifiez). Un fichier typique permettant d'initialiser une partition Windows et deux partitions LINUX se présente ainsi :

```
boot=/dev/hda
prompt
timeout = 50
compact
vga = extended
lock
password = jAN]")Wo
restricted
append = "ether=9,0x300 ; 0xd0000, 0xd4000, eth0 hisax=1,3,5, 0xd8000, 0xd80,
HiSax"
```

```
image = /boot/vwlinuz-2.2.17
        label = linux
        root = /dev/hda5
        read-only
image = /boot/vmlinuz-2.0.38
        label = linux-old
        root = /dev/hda6
        read-only
other = /dev/hda2
        label = win
        table = /dev/hda
```

L'exécution de `lilo` consiste à installer un chargeur de démarrage dans le MBR, qui sait où aller chercher le fichier `/boot/map`. A son tour, ce dernier sait où rechercher le fichier `/boot/vmlinuz-2.2.12-20`. On obtient alors le résultat suivant :

```
[root@cericon]# lilo
Added linux *
Added linux-old
Added win
```

Il en résulte une sauvegarde du MBR existant, si cela n'a pas été fait précédemment, dans le fichier `/boot/boot.0300` (où `0300` correspond aux numéros majeur et mineur).

A présent, parcourons les options :

boot périphérique d'amorçage. Il s'agit soit de `/dev/hda` soit `/dev/sda`, le plus souvent.

prompt affiche une invite derrière laquelle l'utilisateur peut entrer le système d'exploitation à initialiser ;

timeout le nombre de dizièmes de seconde d'affichage de l'invite (ce délai passé, la première image est initialisée).

compact regroupe des secteurs adjacents, ce qui permet un chargement *beaucoup* plus rapide du noyau.

vga supposons qu'on veuille un mode texte 80x50. Vos scripts de démarrage peuvent remetttre ceci à la valeur 80x25. `vga` recherche récursivement dans `/etc/rc.d` tout fichier contenant "`textmode`".

lock fixe le système d'exploitation à amorcer par défaut [il s'agit d'une fonctionnalité qui est souvent utilisée].

password requiert un mot de passe pour lancer l'initialisation.

restricted requiert un mot de passe seulement si quelqu'un essaie d'entrer des options spéciales à l'invite `LILO :`.

append il s'agit d'une *option d'amorçage du noyau*. Les options d'amorçage du noyau sont importantes pour `lilo` et les modules du noyau. Elles sont discutées à la section 43.5. Elles ne sont pratiquement jamais nécessaires dans les installations simples.

image un noyau LINUX à amorcer.

label le texte à saisir à l'invite pour amorcer le noyau ou la partition qui y figure.

root le système de fichiers que le noyau doit monter.

read-only sémaphore spécifiant que le système de fichiers `root` doit initialement être monté en mode lecture.

other précise les autres systèmes d'exploitation à amorcer : dans le cas présent, une partition Windows.

table information relative à la table des partitions et qui doit être passée au secteur d'amorçage de partition. D'autres options `other =` peuvent suivre et plusieurs images de noyau `image =` sont autorisées. Le fichier `lilo.conf` présenté ci-dessus suppose un schéma de partitionnement tel que :

- **/dev/hda1** partition `ext2` de 10 Mo montable sur `/boot`
- **/dev/hda2** partition Windows 98 sur 500 Mo.
- **/dev/hda3** partition étendue
- **/dev/hda4** partition primaire non-utilisée
- **/dev/hda5** système de fichier racine `ext2`
- **/dev/hda6** second système de fichier racine `ext2` contenant une partition plus ancienne.
- **/dev/hd ?** partition de dégagement (swap) LINUX, `/home` et autres partitions.

32.4 Création de disquettes de démarrage.

Si LILO connaît un problème ou qu'il est absent, nous devrons avoir recours à une méthode d'amorçage alternative. Une disquette capable d'initialiser notre système doit contenir une image du noyau à charger, les scripts susceptibles de charger le noyau en mémoire et les informations pour monter `/dev/hda5` comme système de fichiers racine. Pour réaliser la création d'un tel instrument, insérez une disquette alors que votre système LINUX est en activité, et écrasez le contenu de la disquette à l'aide des commandes suivantes :

```
dd if=/boot/vmlinuz-2.2.17 of=/dev/fd0
rdev /dev/fd0 /dev/hda5
```

Amorcez la disquette. Cette méthode requiert une seconde installation LINUX. Si vous ne disposez que d'un système MS-DOS ou Windows, vous devrez (télé)charger l'utilitaire `RAWRITE.EXE` et une image brute du disque d'amorçage. Ces images sont aisément accessibles et vous permettront de réaliser une disquette d'amorçage à partir de DOS. Nous n'irons pas plus loin.

32.5 Complications avec SCSI et `initrd`.

Certaines des explications qui suivent peuvent être difficiles à comprendre sans une connaissance des modules du noyau telle que décrite au chapitre 43. Vous pouvez donc revenir ici plus tard.

Considérons un système avec un disque IDE et un disque SCSI contenant une installation LINUX. Il y a des interruptions du BIOS pour lire les données relatives aux disques SCSI –comme c'est le cas pour les disques IDE, de sorte que LILO peut accéder avec succès à une image du noyau se trouvant sur la partition SCSI. Or, le noyau (voir le chapitre 43) ne peut supporter tous les périphériques possibles à lui seul. En réalité, il est constitué d'un paquet principal (l'image noyau dont nous parlons dans ce chapitre) et plusieurs centaines de modules qui sont des parties chargeables dans le noyau et qui logent dans `/lib/modules/`. Ces modules supportent des périphériques matériels SCSI, réseau, son, etc. L'image du noyau ne contient pas ces modules. Donc, à l'amorçage le noyau se retrouve dénué de ces derniers. Dans le cas de notre disque SCSI, le noyau doit pouvoir être chargé en mémoire et être exécuté sans pouvoir monter le système de fichiers racine (sauf s'il charge le module SCSI auparavant). Cependant, le noyau lui-même réside dans le système de fichiers de `/lib/modules/`. Ceci fait que la situation est délicate à résoudre. Il n'y a que deux possibilités : (a) utiliser un noyau avec le support SCSI pré-chargé ou (b) utiliser une image système (préliminaire) du fichier racine, appelée `initrd`.

La première méthode est directe quoique consommatrice de temps et permet de créer un noyau avec le support SCSI intégré pour votre carte SCSI (on n'utilise donc pas un noyau ayant des modules séparés). Les pilotes intégrés SCSI et réseau reconnaissent automatiquement, la plupart du temps, les cartes de manière à autoriser un accès direct aux périphériques (`dev`ices). Il n'y a pas lieu de passer des options spéciales (voir le chapitre 43) et vous ne devez plus vous préoccuper de configurer ces périphériques. Cette approche dite à *support intégré* (*compiled-in*) se distingue donc de l'approche à *support modulaire*. La taille de l'image du noyau est bien sûr augmentée de la taille des modules. Le chapitre 43 explique comment compiler de tels noyaux.

La seconde méthode est plus rapide mais plus complexe à mettre en oeuvre. LINUX supporte ce qu'il convient d'appeler une image `initrd` (`init`al `r`AM `d`isk). Il s'agit d'un petit système de fichiers d'environ 1,5 Mo qui est chargé par LILO et monté par le noyau en lieu et place du système de fichiers réel. Le noyau monte ce système de fichiers comme un disque RAM, exécute le fichier `/linuxrc` et, par après, monte le système de fichiers réel.

32.6 Création d'une image `initrd`.

Commençons par créer un petit système de fichiers. Faisons un répertoire `~/initrd` et copions les fichiers suivants dans ce répertoire :

```
drwxr-xr-x    7 root     root         1024 Sep 14 20 :12 initrd/
drwxr-cx-x    2 root     root         1024 Sep 14 20 :12 initrd/bin/
-rwxr-xr-x    1 root     root       436328 Sep 14 20 :12 initrd/bin/insmod
-rwxr-xr-x    1 root     root       424680 Sep 14 20 :12 initrd/bin/sash
drwxr-xr-x    2 root     root         1024 Sep 14 20 :12 initrd/dev/
crw-r--r--    1 root     root      5,    1 Sep 14 20 :12 initrd/dev/console
crw-r--r--    1 root     root      1,    3 Sep 14 20 :12 initrd/dev/null
```

```
brw-r--r--    1 root     root     1,   1 Sep 14 20 :12 initrd/dev/ram
crw-r--r--    1 root     root     4,   0 Sep 14 20 :12 initrd/dev/systty
crw-r--r--    1 root     root     4,   1 Sep 14 20 :12 initrd/dev/tty1
crw-r--r--    1 root     root     4,   1 Sep 14 20 :12 initrd/dev/tty2
crw-r--r--    1 root     root     4,   1 Sep 14 20 :12 initrd/dev/tty3
crw-r--r--    1 root     root     4,   1 Sep 14 20 :12 initrd/dev/tty4
drwxr-xr-x    2 root     root       1024 Sep 14 20 :12 initrd/etc/
drwxr-xr-x    2 root     root       1024 Sep 14 20 :12 initrd/lib/
-rwxr-xr-x    1 root     root         76 Sep 14 20 :12 initrd/linuxrc
drwxr-xr-x    2 root     root       1024 Sep 14 20 :12 initrd/loopfs/
```

Il est permis que le fichier `initrd/bin/insmod` soit la version liée statiquement [c'est-à-dire ne requerrant pas de bibliothèques partagées] copiée depuis `/sbin/insmod.static`, un membre du paquet `modutils-2.3.13`. Le fichier `initrd/bin/sash` est un shell statiquement lié au paquet `sash-3.4`. Vous pouvez recompiler `insmod` depuis les sources si vous n'en possédez pas une version liée statiquement. Alternativement, copiez les DLLs nécessaires depuis `/lib` vers `initrd/lib`. Vous pouvez obtenir la listes de DLLs requises en exécutant `ldd /sbin/insmod`. N'oubliez pas de copier également les liens symboliques et d'exécuter `strip -s <lib>` de sorte à réduire la taille des DLLs.

A présent, dans le répertoire `initrd/lib`, copiez les modules SCSI dont vous avez besoin . Par exemple, si vous avez un adaptateur SCSI Adaptec AIC-7850, vous devez disposer du module `aix7xxx.o` qui se trouve dans le répertoire `/lib/modules/<version>/scsi/`. Alors, placez-le dans le répertoire `initrd/lib` :

```
-rw-r--r--    1 root     root       129448 Sep 27 1999 initrd/lib/aic7xxx.o
```

Le fichier `initrd/linuxrc` devrait contenir un script pour charger les modules nécessaires au noyau afin qu'il accède à la partition SCSI. Dans ce cas-ci, il suffit d'indiquer le module `aic7xxx` [`insmod` peut prendre des options telles que IRQ ou IO-port associées au périphérique. Voir le chapitre 43] :

```
#!/bin/sash

aliasall

echo "Loading aic7xxx module"
insmod /lib/aic7xxx.o
```

Maintenant, vérifions tous les droits et emprisonnons-nous (`chroot`) dans le système de fichiers pour effectuer des tests :

```
chroot ~/initrd /bin/bash
/linuxrc
```

Ensuite, créons une image du système de fichiers similairement à ce qui est décrit dans la section 20.9 :

```
dd if=/dev/zero of=~/file-inird count=2500 bs=1024
losetup /dev/loop0 ~file-inird
mke2fs /dev/loop0
mkdir ~/mnt
mount /dev/loop0 ~/mnt
cp -a initrd/* ~/mnt/
umount ~/mnt
losetup -d /dev/loop0
```

Enfin, compressons le système de fichiers (`gzip`) sous un nom de fichier approprié :

```
gzip -c ~/file-inird > initrd-<version_noyau>
```

32.7 Modifier `lilo.conf` pour `initrd`.

La modification à faire dans `lilo.conf` pour forcer l'utilisation du système de fichiers `initrd` est mineure. Il suffit d'ajouter l'option `initrd`. Par exemple :

```
boot=/dev/sda
prompt
timeout = 50
compact
vga = extended
linear
image = /boot/vmlinuz-2.2.17
        initrd = /boot/initrd-2.2.17
        label = linux
        root = /dev/sda1
        read-only
```

Remarquez l'usage de l'option `linear`. Il s'agit d'une astuce associée au BIOS (voir `lilo`(5)). Elle est souvent nécessaire mais peut rendre vos disques SCSI "non-portables" pour d'autres BIOS, ce qui signifie que vous devrez ré-exécuter `lilo` si vous déplacez votre disque SCSI sur un autre ordinateur.

32.8 Utilisation de `mkinitrd`.

A présent que vous avez appris la méthode manuelle de création d'une image `initrd`, vous pouvez lire la page de `man` de l'utilitaire `mkinitrd`. Ce dernier permet de créer une image en une seule commande. Il s'agit d'une commande typique de RedHat portée sur d'autres distributions, aujourd'hui.

Chapitre 33

init, ?getty et niveaux d'exécution UNIX.

Le présent chapitre explique comment Linux (et d'une manière générale, tout système Unix) s'amorce. Nous continuons donc ce qui a été vu à la section 32.2. Nous abordons ici des utilisations avancées de `mgetty`, comme la réception de télécopies (*faxes*).

33.1 `init`, le premier processus.

Après qu'il ait été décompressé en mémoire, le noyau commence son exécution et l'initialisation du matériel. La dernière opération qu'il réalise est le montage du système de fichiers qui contient nécessairement le programme `/sbin/init` que le noyau exécute. `init` est un des seuls programmes que le noyau exécute explicitement ; `init` a la responsabilité de monter le système Unix ; son PID est 1.

Par ailleurs, une autre notion –celle de niveau d'exécution (ou *run level*)– a été introduite. Le niveau d'exécution est le mode sous lequel la machine fonctionne, sachant que la numérotation des niveaux s'étend de 0 à 9. *Dans* un niveau d'exécution donné, le système Unix fonctionne avec un certain nombre de services. Ainsi, la machine pourrait être un serveur de courriel, ou une station de travail fonctionnant sous X selon le niveau d'exécution. Traditionnellement, les niveaux d'exécution sont :

0	"Halt".
1	Mode mono-utilisateur.
2	Mode multi-utilisateur, sans système de fichiers réseau (NFS).
3	Mode multi-utilisateur complet.
4	Non-utilisé.
5	Station de travail X-Window (identique à 3 + écran graphique).
6	Ré-amorçage (reboot).
7	Non-défini.

8	Non-défini.
9	Non-défini.

L'idée sous-jacente est qu'`init` commence à un niveau d'exécution donné qui peut être modifié de manière manuelle par le super-utilisateur. `init` utilise une liste de scripts pour démarrer et arrêter chacun des nombreux services associés à chaque niveau d'exécution. Ces scripts sont `/etc/rc?.d/K`*NNservice* ou `/etc/rc?.d/S`*NNservice* [Sur certains sytèmes, il s'agit de `/etc/rc.d/rc?.d/-...`] où *NN*, `K` ou `S` est un préfixe forçant l'ordre d'exécution (en effet, les fichiers sont ordinairement exécutés selon l'ordre alphabétique).

Tous ces scripts présentent `start` et `stop` comme options de la ligne de commandes, et ce pour initier ou arrêter un service.

Par exemple, lorsque `init` passe du niveau `3` au niveau `5` (disons), il exécute les scripts particuliers de `/etc/rc3.d/` et `/etc/rc5.d` pour activer et désactiver les services idoines. Par exemple, ceci peut impliquer :

```
/etc/rc3.d/S20exim stop
```

33.2 /etc/inittab.

Le fichier de configuration d' `init` s'appelle `/etc/inittab` et il est lu à chaque amorçage.

33.2.1 Configuration minimale.

La configuration minimale d'`inittab` se présente ainsi :

```
id :3 :initdefault

si : :sysinit :/etc/rc.d/rc.sysinit

10 :0 :wait :/etc/rc.d/rc 0
11 :1 :wait :/etc/rc.d/rc 1
12 :2 :wait :/etc/rc.d/rc 2
13 :3 :wait :/etc/rc.d/rc 3
14 :4 :wait :/etc/rc.d/rc 4
15 :5 :wait :/etc/rc.d/rc 5
16 :6 :wait :/etc/rc.d/rc 6

ud : :once :/sbin/update

1 :2345 :respawn :/sbin/getty 38400 tty1
2 :2345 :respawn :/sbin/getty 38400 tty2
3 :2345 :respawn :/sbin/getty 38400 tty3
4 :2345 :respawn :/sbin/getty 38400 tty4

S0 :2345 :respawn :/sbin/mgetty -n 3 -s 115200 ttyS0 57600

S4 :2345 :respawn :/sbin/mgetty -r -s 19200 ttyS4 DT19200
```

```
x :5 :respawn :/usr/bin/X11/xdm -nodaemon
```

Les lignes sont constituées de champs séparés par des doubles-points. En voici la signification (voir `inittab`(5)) :

id :3 :initdefault : cette directive indique que le niveau d'exécution par défaut est `3`. C'est le niveau dans lequel on arrive après l'étape d'initialisation (il s'agit du mode console). Ce champ est habituellement `3` ou `5` (`5` permet d'accéder au mode graphique).

si : :sysinit :/etc/rc.d/rc.sysinit cette directive indique d'exécuter un script à l'amorçage pour initialiser le système. Si vous analysez le fichier `/etc/rc.d/sysinit`, vous y verrez un long script qui permet d'effectuer les opérations suivantes : montage du système de fichiers `proc` ; initialisation du clavier, des polices de caractères des consoles, du domaine NIS, du nom d'hôte et de la partition swap ; exécution d'`isapnp` et de `depmod -a` ; nettoyage du fichier `utmp` et d'autres fichiers. Sur Debian, il y a un script `/etc/init.d/rcS` qui effectue chacune de ces opérations sous `/etc/rcS.d/` [comme d'habitude, Debian s'organise autour de la solution la plus efficace, la plus nette et la plus élégante].

13 :3 :wait :/etc/rc.d/rc 3 Le premier champ est un descripteur (sa valeur importe peu). Le second est une liste des niveaux d'exécution sous lesquels le script décrit dans le dernier champ doit être invoqué : dans le cas présent, `/etc/rc.d/rc 3` doit être exécuté chaque fois qu'on entre dans le niveau 3. Le terme `wait` signifie qu'il faut faire une pause jusqu'à ce que `/etc/rc.d/rc` ait fini son exécution. Le contenu du fichier `/etc/rc.d/rc` indique que ce dernier exécute simplement les scripts sous `/etc/rc?.d/` de manière appropriée pour un changement de niveau d'exécution.

ud : :once :/sbin/update cette instruction vide le "cache" du disque à chaque changement de niveau d'exécution.

1 :2345 :respawn :/sbin/getty 38400 tty1 cette ligne permet d'exécuter la commande `/sbin/getty 38400 tty1` lorsqu'on entre dans les niveaux allant de `2` à `5`. Le terme `respawn` indique de recommencer l'exécution de cette commande si le processus meurt.

x :5 :respawn :/usr/bin/X11/xdm -nodaemon cette instruction indique qu'il faut exécuter la commande `/usr/bin/X11/xdm -nodaemon` quand on entre dans le niveau `5`. Il s'agit du programme de connexion en mode graphique du système X-Window.

33.2.2 Relecture d'`inittab`.

Si vous modifiez le fichier `inittab`, `init` ne tiendra pas compte des modifications à moins de lancer un `SIGHUP`. Ceci revient à taper :

```
telinit q
```

ce qui enjoint à `init` de relire `/etc/inittab`.

33.2.3 L'erreur "respawning too fast".

Chaque fois qu'une ligne d'`inittab` est incorrecte, vous obtenez l'erreur `respawning too fast :` ce peut être le cas lorsque'un `getty` est exécuté sur un port série non-fonctionnel [ces erreurs sont assez communes]. Commentez la ligne en cause ou supprimez-la et ensuite, lancez :

```
telinit q
```

33.3 Niveaux d'exécution utiles.

On passe rarement de manière manuelle d'un niveau à l'autre. La méthode la plus courante pour arrêter une machine est :

```
shutdown -h now
```

qui, effectivement, fait passer au niveau `0` et :

```
shutdown -r now
```

qui réamorce en passanr par le niveau `6`.

Vous pouvez aussi indiquer le niveau d'exécution derrière l'invite `LILO :`. Tapez :

```
linux 1
```

ou

```
linux single
```

pour entrer en mode mono-utilisateur lorsque vous initialisez votre système. Sur un système en cours de fonctionnement, vous passez en mode mono-utilisateur de cette manière :

```
telinit S
```

Vous pouvez forcer le passage de tout niveau à un autre avec :

```
telinit <N>
```

33.4 Information à propos de `getty`.

La page de `man` de `getty` commence de cette manière :

> **getty** ouvre un port tty, présente l'invite de connexion et invoque la commande /bin/login. Elle est normalement invoquée par *init*(8).

Notez que `getty`, `agetty`, `fgetty` et `mingetty` sont divers mises en oeuvre de `getty`. La trace la plus visible de l'exécution d'`init` est qu'elle engendre une

invite de connexion sur les consoles virtuelles de LINUX. C'est `getty` (ou parfois `mingetty`) –tel que spécifié dans `inittab`– qui affiche cette invite. Une fois l'identifiant introduit, `getty` invoque le programme `/bin/login`, qui présente une invite à saisir le mot de passe.

Le programme `login` (abordé à la sous-section 12.7.1) lance un shell. Quand ce shell meurt (du fait que l'utilisateur quitte sa session en utilisant la commande `exit`), `getty` engendre à nouveau une invite de connexion.

33.5 Sommaire du démarrage.

Si nous couplons nos informations avec celles du chapitre 32, nous disposons d'une image complète du processus de démarrage d'un système UNIX :

1. le premier secteur est chargé en mémoire RAM et exécuté –l'invite `LILO :` apparaît,
2. le noyau est chargé depuis la liste de secteurs,
3. le noyau est exécuté et décompressé,
4. le noyau initialise les périphériques matériels,
5. le noyau monte le système de fichier racine, disons `/dev/hda1`,
6. le noyau exécute `/sbin/init` avec un PID=1,
7. `init` excécute tous les scripts du niveau d'exécution par défaut défini dans `/etc/inittab`,
8. `init` engendre les programmes `getty` sur chaque terminal.
9. `getty` affiche l'invite de connexion,
10. `getty` exécute `/bin/login` pour authentifier l'utilisateur,
11. le programme `login` lance un `shell`.

33.6 Télécopies entrantes et connexions modems.

33.6.1 `mgetty` et terminaux "caractère".

L'idée sous-jacente à `getty` était de gérer les terminaux caractères sur les ordinateurs centraux appelés en anglais *mainframes*. `mgetty` est un programme plus complet dérivé de `getty`, qui permet d'utiliser divers matériels. Une entrée typique d'`inittab` telle que :

```
S4 :2345 :respawn :/sbin/mgetty -r -s 19200 ttyS4 DT19200
```

sera capable d'ouvrir une connexion sur un terminal connecté à un port série sur `/dev/ttyS4`. Voir la page 43.6.12 pour toute information sur la configuration des cartes série multiports.

(Les périphériques LINUX de `/dev/tty1` à `/dev/tty12` tels qu'utilisés par `getty` émulent des terminaux classiques de cette façon.)

33.6.2 Fichiers de journalisation de mgetty.

La journalisation de mgetty se fait dans /var/log/mgetty.log.ttyS?. Ces journaux contiennent toutes les informations utiles au dépannage. Il est intéressant d'utiliser tail -f sur ces fichiers pendant qu'un login est en cours d'installation.

33.6.3 mgetty avec les modems.

Exécuter mgetty (voir mgetty(8)) constitue une manière élémentaire d'ouvrir une communication vers une machine LINUX. Votre script inittab doit contenir la directive :

```
S0 :2345 :respawn :/sbin/mgetty -n 3 -s 115200 ttyS0 57600
```

où -n 3 indique qu'il faut répondre après 3 coups de sonnerie. Il n'y a plus rien d'autre à faire que de connecter votre modem à la prise téléphonique. Vous pouvez utiliser dip -t, comme cela est indiqué à la sous-section 42.1.1, pour que deux machines LINUX puissent communiquer. Voici un exemple de session [ceci suppose qu'une ligne AT&F1 est suffisante. Voir la section 4.5] :

```
[root@cericon]# dip -t
DIP : Dialup IP protocol Driver version 3.3.7o-uri (8 Feb 96)
Written by Fred N. van Kempen, MicroWalt Corporation.

DIP> port ttyS0
DIP> speed 57600
DIP> term
[ Entering TERMINAL mode. Use CTRL-] to get back ]
AT&F1
OK
ATDT5952521
CONNECT 19200/ARQ/V34/LAPM/V42BIS

RedHat Linux Release 6.1 (Cartman)
Kernel 2.2.12-20 on an i686

remote.dialup.private login :
```

Notez que ceci est une session de connexion qui n'a rien à voir avec une communication PPP.

33.6.4 mgetty recevant des télécopies.

Par défaut, mgetty reçoit des télécopies (*fax*), pourvu que votre modem tolère ce mode de transmission [si votre modem accepte les télécopies et que cela ne fonctionne pas, vous devrez consacrer du temps à lire le manuel de votre modem et la documentation info mgetty] et pourvu que vous n'ayez pas désactivé cette fonctionnalité de manière explicite en utilisant l'option -D. Votre fichier inittab devra contenir :

```
S0 :2345 :respawn :/sbin/mgetty -x 4 -n 3 -s 57600 -I '27 21 7654321'
ttyS0 57600
```

Les options signifient, dans l'ordre : fixer le niveau de débogage à 4, répondre après 3 coups de sonnerie, fixer la vitesse du port à 57600, et fixer le numéro d'ID du télécopieur à 27 21 7654321. Vous pourriez subsituer cette commande par :

```
S0 :2345 :respawn :/sbin/mgetty ttyS0 57600
```

et écrire les options de configuration dans le fichier mgetty.config sous /etc/-mgetty+sendfax/ :

```
debug 4
rings 3
speed 57600
fax-id 27 21 7654321
```

Les télécopies aboutiront dans /var/spool/fax/incoming/. Par ailleurs, g3 formate les fichiers, mais observez comment la commande :

```
strings /sbin/mgetty | grep <nouveau_fax>
```

agit avec :

```
/etc/mgetty+sendfax/nouveau_fax
```

qui est un script que mgetty applique discrètement lorsqu'une nouvelle télécopie arrive. Cela peut être utilisé pour convertir des télécopies en un fichier (comme les fichiers .gif [les fichiers .png sont meilleurs]) lisible par des programmes de bureautique. Le script /etc/mgetty+sendfax/nouveau_fax transfère les télécopies dans /home/fax/ sous forme de fichiers .gif auxquels tous les utilisateurs peuvent accéder [il s'agit d'une variante de ce qu'ont écrit les contributeurs de mgetty]. Remarquez comment le script tire parti du programme convert appartenant au paquet logiciel Imagemagic :

```
#!/bin/sh


# you must have pbm tools and they must be in your PATH
PATH=/usr/bin :/bin :/usr/X11R6/bin :/usr/local/bin


HUP="$1"
SENDER="$2"
PAGES="$3"


shift 3
P=1
```

```
while [ $P -le $PAGES ]; do
   FAX=$1
   BASENAME=`basename $FAX`
   RES=`echo $BASENAME | sed 's/.\(.\).*/\1/'`
   if [ "$RES"="n" ]; then
      STRETCH="-s"
   else
      STRETCH=""
   fi
   nice g32pbm $STRETCH $FAX > /tmp/$BASENAME.pbm \
      && rm -f $FAX \
      && nice convert -colorspace gray -colors 16 -geom \
         '50%x50%' /tmp/$BASENAME.pbm /home/fax/$BASENAME.gif \
      && rm -f /tmp/$BASENAME.pbm \
      && chmod 0666 /home/fax/$BASENAME.gif
   shift
   P=`expr $P + 1`
done

exit 0
```

Chapitre 34

Envoi de télécopies.

Dans ce chapitre-ci, nous étudions le programme `sendfax`,[1] de manière à utiliser une pseudo-imprimante qui utilisera automatiquement un modem pour envoyer ses travaux d'impression à des télécopieurs distants.

34.1 De l'imprimante au télécopieur.

Cette partie prolonge la section 22.10.

A ce stade, vous devriez lire la section `sendfax` de la page d'`info` de `mgetty`. La commande `sendfax` est essentiellement un programme qui envoie des télécopies *via* un modem. De la même manière que `mgetty`, elle lit un fichier de configuration dans `/etc/mgetty+sendfax/`. Ce fichier de configuration s'appelle `sendfax.config` et peut contenir une information aussi minimale que :

```
verbose y
debug 5
fax-devices ttyS0
fax-id 27 21 7654321
max-tries 3
max-tries-continue y
```

Ci-dessous, vous trouverez un script `fax_filter.sh` qui permet d'envoyer un

[1] NdT : `hylafax` est un autre logiciel libre complet de télécommunication pour machines UNIX. Il permet d'envoyer et de recevoir des télécopies depuis des machines en réseau TCP/IP, et de partager un modem de manière transparente entre plusieurs machines sur réseau TCP/IP.
Ce programme s'appuie sur une architecture client/serveur. Parmi ses caractéristiques :
- les télécopies peuvent être de toutes tailles (A4, B4), en 98 ou 196 lpi,
- les télécopies peuvent être transmises ou reçues en format "1D-" ou "2D-encoded",
- hylafax supporte tout modem de classe 1, classe 2 ou classe 2.0, - les documents qui peuvent être transmis sont au format Postscript ou TIFF classe F,
- hylafax peut produire des pages de garde,
- hylafax fournit une passerelle avec la messagerie,
- les documents reçus sont en format TIFF, classe F. Ils peuvent être convertis en Postscript par l'utilitaire `fax2ps` et imprimés ou envoyés par courriel sous forme de document en fonction du numéro de télécopieur de l'expéditeur.
- le modem-télécopieur est partagé avec les applications de communications sortantes qui respectent le protocole "UUCP locking" : `cu`, `minicom`, `tip`, `kermit`, `uucp`, `slip`, et `ppp`.

Le lecteur se référera au Howto sur http ://www.hylafax.org/howto/

travail d'impression via le télécopieur cible après avoir demandé son numéro téléphonique grâce à la commande `gdialog`. [`gdialog` est une partie du paquet `gnome-utils`]. Une entrée idoine dans `/etc/printcap` est :

```
fax :\
        :sd=/var/spool/lpd/fax :\
        :mx#0 :\
        :sh :\
        :lp=/dev/null :\
        :if=/var/spool/lpd/fax/fax_filter.sh :
```

Le fichier `fax_filter.sh` lui-même pourrait contenir un script comme celui-ci [rappelez-vous de permettre la rotation du fichier `/var/log/fax` comme cela a été vu à la page 240] :

```
#!/bin/sh

exec 1>>/var/log/fax
exec 2>>/var/log/fax

echo
echo
echo $@

echo "Starting fax `date` : I am `id`"

export DISPLAY=localhost :0.0
export HOME=/home/lp

function error()
{
   gdialog --title "Send Fax" --msgbox "$1" 10 75 || \
      echo 'Huh? no gdialog on this machine'
   cd /
   rm -Rf /tmp/$$fax || \
      gdialog \
         --title "Send Fax" \
         --msgbox "rm -Rf /tmp/$$fax failed" \
         10 75
   exit 1
}

mkdir /tmp/$$fax || error "mkdir /tmp/$$fax failed"
cd /tmp/$$fax || error "cd /tmp/$$fax failed"

cat > fax.ps

if /usr/bin/gdialog \
         -- title "Send Fax" \
```

```
        --inputbox "Enter the phone number to fax :" \
        10 75 "" 2>TEL ; then
   :
else
   echo "gdialog failed" "< TEL""
   rm -Rf /tmp/$$fax
   exit 0
fi

TEL="< TEL"
test -z "$TEL" && error 'no telephone number given'
cat fax.ps | gs -r204x98 -sOutputFile=- -sDEVICE=faxg3 -dBATCH -q - \
   1>fax.ps.g3 || error 'gs failed'

ls -al /var/lock/
/usr/sbin/sendfax -x 5 -n -l ttyS0 STEL fax.ps.g3 || \
   error "sendfax failed"

rm -Rf /tmp/$$fax

exit 0
```

34.2 Binaire d'interface setgid.

Le script présenté ci-dessus n'est toutefois pas suffisant. Dans ce qui précède, `sendfax` demande un accès au périphérique `/dev/ttyS0` et au répertoire `/var/lock/` (pour créer un fichier de verrouillage du modem, voir la section 35.4). Ceci ne peut se faire en tant qu'utilisateur `lp` (sous lequel le filtre que nous utilisons dans le script est exécuté). Sur RedHat, la commande `ls -ald /var/lock /dev/ttyS0` montre que seul `uucp` est autorisé à accéder au modem. Pour contourner cette restriction, nous devons créer un binaire setgid (voir le chapitre 15) qui s'exécute en tant qu'utilisateur `uucp`. Réalisez cela en compilant le programme suivant :

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv)
{
   char **a ;
   int i ;

/* set the real group ID to that of the effective group ID */
   if (setdig (getegid ())) {
       perror ("sendfax_wrapper : setgid failed") ;
```

```
        exit (1);
    }

/* copy all arguments */
    a = (char **) malloc ((argc + 1) * sizeof (char *));
    for (i = 1; i < argc; i++)
        a[i] = (char *) strdup (argv [i]);
    a[argc] = NULL;

/* execute sendfax */
    a[0] = "/usr/sbin/sendfax";
    execvp (a[0], a);

/* exit on failure */
    perror ("sendfax_wrapper : failed to execute /usr/sbin/sendfax");
    exit (1);
}
```

en utilisant les commandes :

```
gcc sendfax_wrapper.c -o /usr/sbin/sendfax_wrapper -Wall
chown lp :uucp /usr/sbin/sendfax_wrapper
chmod g+s,o-rwx /usr/sbin/sendfax_wrapper
```

Ensuite, remplacez `sendfax` par `sendfax_wrapper` dans le script de filtre. Vous constaterez que `sendfax_wrapper` ne fait qu'exécuter `sendfax` après avoir converti l'ID du groupe en l'ID effectif du groupe (GID) tel qu'obtenu à partir de la fonction `getegid` à la ligne 12. L'ID effectif de groupe est `uucp` en raison du setgid (`g+s`) se trouvant dans la commande `chmod`. D'où le fait que `sendfax` est exécuté sous le groupe `uucp` avec un accès légitime au périphérique modem.

Sur votre propre système, il peut s'avérer plus judicieux d'essayer de mettre en oeuvre ce qui vient d'être décrit sans une interface (*wrapper*). Debian, par exemple, a un groupe `dialout` pour autoriser les accès aux modems. Soyez également conscient que certaines distributions n'utilisent pas forcément `uucp` de la même manière que RedHat. Vous devrez peut-être créer un utilisateur adéquat pour atteindre votre objectif.

Chapitre 35

uucp et uux.

uucp est une commande permettant de copier un fichier d'un système UNIX sur un autre système UNIX. Le programme uux exécute une commande sur une autre machine UNIX, même si cette commande est en train de recevoir des données de **stdin** sur le système local. uux est extrêmement utile pour automatiser de nombreuses fonctions distribuées, comme le courriel ou les nouvelles (*news*).

Les commandes uucp et uux proviennent du paquet uucp (*Unix-to-Unix Copy*). L'intérêt d'uucp peut sembler mineur par rapport aux commandes modernes telles que ssh, scp et même les programmes de transfert FTP. Cependant, uucp –quoiqu'étant un peu ancien– possède des caractéristiques essentielles, comme celle d'exécuter un travail en différé, en plaçant ce dernier dans une file d'attente et en recontactant la machine distante durant la nuit –par exemple– pour terminer l'opération.

uucp est antérieur à l'internet ; il a été initialement utilisé pour mettre en oeuvre un système de courriel exclusivement basé sur les lignes téléphoniques et les modems. Par conséquent, ce programme possède des protocoles élaborés pour assurer que votre fichier et/ou votre commande exécute(nt) bien ce pourquoi il (elle) est destiné(e), avec une tolérance maximale quant au nombre d'erreurs et un nombre minimum de transmissions. C'est pourquoi, uucp devrait toujours être utilisé pour des tâches automatisées lorsque des connexions non-fiables (par modems) sont mises en jeu. La version d'uucp associée aux distributionx LINUX est appelée "Taylor UUCP".

Il est très important de se rappeler que lors d'une connexion interrompue par un problème de ligne, uucp n'élimine pas les données déjà transmises. Donc, quels que soient la lenteur ou les erreurs de communication, le travail progresse sans reprendre à zéro. Comparez cela aux protocoles SMTP ou POP3/IMAP : toute coupure de transmission exige de recommencer l'opération depuis le début.

35.1 Opérer en ligne de commandes.

Pour copier un ficher d'une machine à une autre, entrez simplement :

```
uucp <nom_de_fichier> <machine>!<chemin>
```

Vous pouvez également lancer la commande sur un système distant :

```
echo -n 'Bonjour, ceci est un petit message\n\n-paul' | \
                                    uux - 'cericon!rmail' 'john'
```

ce qui exécute `rmail` sur le système distant appelé `cericon` et alimente la commande `rmail` avec un petit texte. Notez bien que vous devez placer le signe `!` entre guillemets simples pour en éviter l'interprétation par le shell. Ces commandes échoueront presque toujours avec un message `permission denied by remote`. L'erreur sera rapportée dans un message de courriel à l'utilisateur qui exécute la commande.

35.2 Configuration.

La documentation d'`uucp` est au format HTML sur RedHat (`/usr/doc/-uucp/version/uucp.html` ou `/usr/share/...`), et au format `info` sur Debian et RedHat. Voyons à présent une configuration typique de base.

Le paquet `uucp` a été l'objet de nombreuses révisions depuis l'apparition des premiers réseaux communiquant par modems. Les dernières éditions GNU incluses dans LINUX ont un format de fichier de configuration qui diffère très certainement des premières versions d'`uucp`. Les réseaux de communications actuels combinent l'usage d'`uucp` et des communications PPP (point-à-point) classiques. De ce fait, ils n'exploitent pas toutes les facilités de communication propres à `uucp`. Par exemple, si vous déployez un réseau d'hôtes distants qui utilisent des modems, ces hôtes devraient toujours utiliser `uucp` pour charger (*upload*) et réceptionner le courriel, plutôt que POP3/IMAP ou SMTP, ceci, en raison des problèmes de transmission évoqués plus haut. En d'autres mots, `uucp` fonctionne comme un service TCP ordinaire tout en étant plus tolérant aux erreurs de communication.

Pour configurer `uucp` comme un service TCP, modifiez `/etc/inetd.conf` comme suit :

```
uucp  stream tcp  nowait  uucp /usr/sbin/tcpd  /usr/lib/uucp/uucico -l
```

tout en étant très attentif à restreindre les hôtes autorisés à se connecter, en exploitant les techniques discutées au chapitre 30. De la même manière, sous `xinetd`, créez un fichier `/etc/xinetd.d/uucp` contenant :

```
service uucp
{
    only_from    = 127.0.0.1 192.168.0.0/16
    socket_type  = stream
    wait         = no
    user         = uucp
    server       = /usr/lib/uucp/uucico
    server_args  = -l
    disable      = no
}
```

Les fichiers de configuration d'`uucp` se trouvent sous `/etc/uucp/`. A présent,

nous allons configurer une machine cliente, machine1.cranzgot.co.za, pour envoyer du courriel *via* server1.cranzgot.co.za, avec server1.cranzgot.co.za exécutant le service uucico mentionné dans les configurations inetd.conf et xinetd.conf.

uucp possède un mécanisme d'authentification assez ancien qui exploite sa propre liste d'utilisateurs et de mots de passe complètement distincts des comptes UNIX ordinaires. Avant tout, nous devons ajouter un "utilisateur" commun et un mot de passe pour les deux machines afin de permettre l'authentification. Pour machine1.cranzgot.co.za, nous ajoutons la ligne suivante dans le fichier /etc/uucp/call :

```
server1    machine1login    pAsSwOrD123
```

qui indique à uucp d'utiliser la connexion machine1login lorsqu'il y a tentative de communication avec server1. Sur la machine appelée server1.cranzgot.co.za, nous pouvons ajouter la ligne suivante au fichier /etc/uucp/passwd :

```
machine1login    pAsSwOrD123
```

Notez que le nom uucp "server1" a été choisi pour server1.cranzgot.co.za pour des raisons de convenance : les noms uucp n'ont, cependant, rien à voir avec les noms de domaine.

Ensuite, nous devons indiquer à uucp comment on va utiliser machine1. Les opérations de "connexion à" ou de "connexion depuis" machine1 doivent être mentionnées dans le fichier /etc/uucp/sys. Notre entrée est :

```
system machine1
call-login *
call-password *
commands rmail
protocol t
```

Toutefois, il peut y avoir autant d'entrée qu'il est jugé utile. Les seules informations que server1 doit connaître à propos de machine1 sont l'utilisateur et son mot de passe, et aussi le protocole. Le caractère * indique qu'il faut extraire de /etc/uucp/passwd le nom d'utilisateur et son mot de passe. Le terme protocol t indique qu'il faut utiliser un protocole de correction (comme c'est aussi le cas pour TCP). L'option commands prend une liste (avec séparation par un espace) de commandes permises. Pour des raisons de sécurité, les commandes non-répertoriées dans cette liste ne sont pas exécutées, d'où l'origine du signal permission denied by remote mentionné à la section 35.1.

Le fichier /etc/uucp/sys sur machine1 contient :

```
system server1
call-login *
call-password *
time any
port TCP
address 192.168.3.2
```

```
protocol t
```

En l'occurrence, `time any` indique à quels moments de la journée `uucp` doit effectuer ses appels vers `server1`. Par defaut, la valeur de `time` est `Never` [Voir la documentation d'`uucp` sous **Time Strings** pour plus d'information]. L'option `TCP port` signifie que nous employons un modem nommé `TCP` pour effectuer la communication. Tous les modems sont définis dans le fichier `/etc/uucp/port`. Nous pouvons ajouter notre modem dans le fichier `/etc/uucp/entry` de la manière suivante :

```
port TCP
type tcp
```

ce qui, clairement n'apparaît pas comme un modem.

Enfin, dans la file d'attente, nous pouvons placer un travail de transfert de courriel avec :

```
echo -e 'Bonjour Jacques\n\Comment allez-vous?\n\n-Gilles" | \
          uux - --nouucico 'server1!rmail' 'jacques@beanstalk.com'
```

et copier un fichier à l'aide de la commande :

```
uucp --nouucico README 'cericon!/var/spool/uucppublic'
```

Observez que `/var/spool/uucppublic/` est le seul répertoire auquel vous êtes autorisé à accéder par défaut. Vous conserverez très probablement cette configuration pour des raisons de sécurité.

uucico

Bien que nous ayons mis un travail en attente, rien ne sera transféré tant que le programme `uucico` ne sera exécuté (`uucico` est un acronyme d'*Unix-to-Unix copy in copy out*). Le principe est qu'à la fois `server1` et `machine1` peuvent avoir mis en attente une certaine quantité de travaux, si bien que quand `uucico` est en cours sur les deux ordinateurs et en négociation avec chacune d'elles, les travaux des deux machines sont traités un à un, et cela quelle que soit la machine qui ait initié la connexion.

D'ordinaire, `uucico` est lancé par le démon `crond` (voir le chapitre 38) toutes les heures.[1] Vous pouvez lancer `tail -f /var/uucp/Log` lorsque vous venez d'exécuter `uucico` manuellement de cette manière :

```
uucico --debug 3 --force --system server1
```

Plus le degré de débogage est élévé, plus le contenu de `Log` est chargé. La commande ci-dessus communique de manière `--force`e avec le `--system`e `server1` à tout moment après qu'ait eu lieu la dernière communication. Usuellement, il

[1] NdT : `crond` est un démon qui lance récurremment des commandes ou des scripts à des heures, des jours, etc. donnés. Il est à distinguer d'`at` qui effectue une opération une seule fois (voir le chapitre 38).

y a des contraintes sur les appels après un échec de communication : l'option `--force` contourne ce problème.

Si votre serveur de messagerie sur `server1` est configuré correctement, il devrait maintenant avoir mis en attente le message sur la partie distante.

35.3 Communication par modem.

Si vous décidez d'employer `uucp` "à l'ancienne", vous pouvez utiliser `mgetty` pour répondre aux appels `uucp` sur `server1` en ajoutant la ligne suivante au fichier `/etc/inittab` :

```
S0 :2345 :respawn :/sbin/mgetty -s 57600 ttyS0
```

et aussi la ligne :

```
machine1login  uucp  machine1login  /usr/sbin/uucico -l -u machine1login
```

au fichier `/etc/mgetty+sendfax/login.config` (`/etc/mgetty/login.config` sur Debian). Vous ajouterez également un compte UNIX `machine1login` avec un mot de passe `pAsSwOrD123`. Cette méthode fonctionne car `mgetty` et `uucico` partagent la même invite de connexion et la même invite de mot de passe ; cependant, `mgetty` se réfère à `/etc/passwd` au lieu de `/etc/uucp/passwd` lors de l'authentification. Par ailleurs, concernant la connexion modem, `protocol t` est enclin à l'erreur : remplacez ce motif par `protocol g`.

Sachez que la configuration décrite ci-dessus supporte les télécopies, les connexions (*login*), la voix et PPP (voir la section 42.5).

Pour communiquer depuis `machine1`, vous devez indiquer un modem au préalable (en plus de `TCP`) dans le fichier `/etc/uucp/port` :

```
port ACU
type modem
device /dev/ttyS0
dialer mon_modem
speed 57600
```

`ACU` est une terminologie ancienne qui désigne l'*unité d'appel automatique* (c'est à dire le modem : *Automatic Calling Unit*). Nous devons apporter des précisions supplémentaires pour les ports série, telles que le périphérique (`/dev/ttyS0` pour un modem sur COM1) et la vitesse de transfert sur la ligne série. Il est également nécessaire de préciser un moyen d'initialiser le modem : d'où l'option `dialer mon_modem`. Un fichier `/etc/uucp/dial` devrait contenir une entrée correspondant à "`mon_modem`" comme ceci [cet exemple suppose qu'une initialisation AT&F1 est suffisante] :

```
dialer mon_modem
chat "" AT&F1\r\d\c OK\r ATDT\D CONNECT
chat-fail RING
chat-fail NO\sCARRIER
```

```
chat-fail ERROR
chat-fail NO\sDIALTONE
chat-fail BUSY
chat-fail NO\sANSWER
chat-fail VOICE
complete \d\d+++\d\dATH\r\c
abort \d\d+++\d\dATH\r\c
```

Nous verrons davantage de détails sur les modems lors de l'analyse de `pppd` au chapitre 42. Une fois le modem correctement spécifié, nous pouvons modifier l'entrée dans le fichier `sys` en :

```
system server1
call-login *
call-password *
time any
port ACU
phone 555-6789
protocol g
```

Les mêmes commandes `uux` devraient maintenant fonctionner lors des communications.

35.4 Fichiers de verrouillage de `tty`/UUCP.

Nous avons fait allusion aux fichiers de verrouillage à la section 34.2. Voici quelques informations supplémentaires.

Depuis un moment déjà, vous avez noté que certains services utilisent les périphériques séries et beaucoup d'entre eux peuvent même employer le même périphérique à des moments différents. Il pourrait en résulter des conflits si deux services essayaient de recourir simultanément au même périphérique. Par exemple, que se passerait-il si quelqu'un essayait d'envoyer une télécopie alors qu'une autre personne est en communication ?

La solution se trouve dans le verrouillage par fichier UUCP (*UUCP lock file*). Il s'agit d'un fichier créé par un processus en cours dans `/var/lock` sous la forme *LCK..device*, qui indique le port série utilisé par ce processus. Supposons que `sendfax` soit en cours d'exécution et exploite un modem connecté sur `/dev/ttyS0`, un fichier `/var/lock/LCK..ttyS0` apparaît dès la création du processus. Ceci est dû au fait que `sendfax`, parmi d'autres programmes `mgetty`, respecte la convention de verrouillage de fichiers UUCP. Ce fichier contient l'ID de processus du programme utilisant le périphérique série. Un fichier de verrouillage associé à un processus mort est appelé *fichier de verrouillage éventé (stale lock file)* et peut être éliminé manuellement.

35.5 Déboguer `uucp`.

Il est rare que les mises en oeuvre d'`uucp` fonctionnent sans problème du premier coup. Heureusement, vous disposez d'une série d'options plus ou moins

volubiles de débogage. `uucico` prend l'option `--debug` pour permettre d'indiquer le niveau de débogage. Vous devriez examiner les fichiers :

- `/var/log/uucp/Log`,
- `/var/log/uucp/Debug` et
- `/var/log/uucp/Stats`,

de manière à avoir une idée de ce qui se passe du point de vue technique. Il y a aussi un répertoire important : `/var/spool/uucp/`. Vous pouvez spécifier le niveau de débogage avec `--debug` *niveau* où *niveau* est un entier allant de `0` à `11`. Vous pouvez aussi tirer parti de `--debug chat` pour ne visualiser que les commentaires relatifs à la communication par modem. Voici les autres options au complet [sur base de la documentation d'`uucp`] :

–**debug abnormal** messages de débogage pour des situations anormales telles que les erreurs récupérables,

–**debug chat** idem pour les scripts de "chat",

–**debug handshake** idem pour les échanges initiaux entre deux machines,

–**debug uucp** idem pour les protocoles de session UUCP,

–**debug proto** idem pour les protocoles de liaison individuelle,

–**debug port** idem pour les actions sur le port de communication,

–**debug config** messages de débogage lors de la lecture des fichiers de configuration,

–**debug spooldir** idem pour les actions dans le répertoire de spoule,

–**debug execute** idem lorsqu'un autre programme est exécuté.

–**debug incoming** affiche toutes les données entrantes dans le fichiers de débogage,

–**debug outgoing** idem pour les messages sortants

–**debug all** regroupe tous les messages de débogage.

35.6 Utilisation d'`uux` avec `exim`.

Sur `machine1`, nous allons utiliser `exim` pour mettre en réserve tous les messages électroniques via `uucp`. L'utilisation d'`uucp` requiert un transport par tube (les transports d'`exim` sont expliqués à la sous-section 31.3.2). Le MTA `exim` envoie simplement un message électronique *via* l'entrée standard stdin de la commande `uux` et ne se préoccupe plus de rien. A partir de là, `uux` devient responsable de l'exécution de `rmail` sur `server1`. Le fichier `exim.conf` se présente comme suit :

```
##################### MAIN CONFIGURATION SETTINGS ####################
log_subject
errors_address = admin
local_domains = localhost : ${primary_hostname} : machine1 : \
                                    machine1.cranzgot.co.za
host_accept_relay = 127.0.0.1 : localhost : ${primary_hostname} : \
                            machine1 : machine1.cranzgot.co.za
never_users = root
exim_user = mail
```

```
exim_group = mail
end
###################### TRANSPORTS CONFIGURATION ######################
uucp :
   driver = pipe
   user = nobody
   command = "/usr/bin/uux - --nouucico ${host}!rmail \
                                    ${local_part}@${domain}"
   return_fail_output = true
local_delivery :
   driver = appendfile
   file = /var/spool/mail/${local_part}
   delivery_date_add
   envelope_to_add
   return_path_add
   group = mail
   mode_fail_narrower =
   mode = 0660
end
###################### DIRECTORS CONFIGURATION ####################
localuser :
   driver = localuser
   transport = local_delivery
end
###################### ROUTERS CONFIGURATION ######################
touucp :
   driver = domainlist
   route_list = "* server1"
   transport = uucp
end
###################### RETRY CONFIGURATION ########################
*                    *           F,2m,1m
end
```

Sur la machine appelée `server1`, `exim` doit être exécuté comme serveur de courriel véritable de manière à acheminer correctement le message vers sa destination. Naturellement, sur `server1`, `rmail` est l'expéditeur ; par conséquent, du point de vue d'`exim`, le message semble lui venir de la machine locale. Ceci signifie qu'il n'y a pas de configuration supplémentaire requise pour supporter le courriel *provenant de* la commande `uux`.

Notez que vous pouvez ajouter d'autres domaines à votre `route_list` de façon que vos communications aient lieu directement sur la machine réceptrice. Par exemple :

```
route_list = "machine2.cranzgot.co.za machine2 ; \
                machine2                machine2 ; \
                machine3.cranzgot.co.za machine3 ; \
                machine3                machine3 ; \
                *                        server1"
```

Vous pouvez ajouter d'autres entrées à votre fichier /etc/uucp/sys, comme par exemple :

```
system machine2
call-login *
call-passwd *
time any
port ACU
phone 555-6789
protocol g

system machine3
call-login *
call-passwd *
time any
port ACU
phone 554-3210
protocol g
```

Le fichier de configuration exim.conf sur server1 doit aussi avoir un routeur pour recueillir les messages de machine1. Le routeur ressemblera à :

```
###################### ROUTERS CONFIGURATION ####################
toouucp :
   driver = domainlist
   route_list = "machine2.cranzgot.co.za machine2 ; \
                 machine2                machine2 ; \
                 machine3.cranzgot.co.za machine3 ; \
                 machine3                machine3"
   transport = uucp
lookuphost :
   driver = lookuphost
   transport = remote_smtp
end
```

Ce routeur envoie tous les messages électroniques correspondant à nos hôtes qui appellent via le transport uucp tandis que tous les autres méls (c'est-à-dire ceux destinés à l'internet) sont dirigés vers le routeur lookuphost.

35.7 Programmation des appels sortants.

Jusqu'à présent, nous avons employé uucico de manière manuelle. Par lui-même, uucico ne travaille pas en tant que démon et doit être invoqué par crond. Tous les sytèmes qui utilisent uucp ont une entrée dans le fichier /etc/crontab ou un script sous /etc/cron.hourly.

Voici un exemple typique d'entrée de /etc/crontab dans le cas de machine1 :

```
45 *       * * * uucp /usr/lib/uucp/uucico --master
40 8,13,18 * * * root /usr/bin/uux -r server1
```

L'option `--master` indique à `uucico` d'effectuer une boucle parmi les travaux en attente et d'appeler toutes les machines pour lesquelles des travaux sont en attente. Cela se fait toutes les heures. La seconde ligne effectue une commande vide trois fois par jour en ce qui concerne la machine `server1`. Ceci force `uucico` à contacter `server1` au moins trois fois par jour comme s'il y avait un travail à réaliser. L'idée est de trouver les travaux qui proviendraient d'autres sources. Ce processus est connu sous le nom de création de *poll file*.

De ce que nous venons de voir, il ressort clairement que vous pouvez utiliser `uucp` sur TCP grâce à une liaison initiée par le démon `pppd`. Si la communication est établie à la demande, un appel `uucp` sera déclenché et il en résultera une connexion TCP directe vers l'hôte distant. C'est communément le cas quand des systèmes satellites contactent un fournisseur de services internet qui n'a pas d'équipement `uucp`. Pour satisfaire la demande, un serveur `uucp` est déployé. Le serveur a une connexion internet permanente et écoute sur TCP tout transfert `uucp`.

Chapitre 36

Le système de fichiers LINUX.

Ce chapitre est une traduction du texte original intitulé *Filesystem Hierarchy Standard*. Un reformatage mineur et une modification de la numérotation ont été réalisés (adaptation au présent document). L'original peut être obtenu à l'adresse http ://www.pathname.com/fhs/.

Si vous vous êtes déjà posé les questions "Où doit se trouver le fichier *xxx* dans mon système de fichiers ?" ou "Que contient le répertoire *yyy* ?", lisez ce document. Vous pouvez le considérer comme le fin mot en matière d'organisation du système de fichiers LINUX. Bien qu'il constitue une référence pour les personnes construisant des distributions LINUX, tous les administrateurs pourront tirer avantage des règles et des explications fournies par le document.

Système de fichiers normalisé –Version 2.2

Groupe "Normalisation du système de fichiers"

publié par Rusty Russell et Daniel Quinlan

SOMMAIRE

Cette norme comprend un ensemble de conditions et de directives concernant l'emplacement des fichiers et des répertoires dans le cas des systèmes d'exploitation de type UNIX. Les directives sont prévues pour supporter l'interopérabilité des applications, les outils d'administration, les outils de développement et les scripts ainsi que la plus grande uniformité possible de la documentation de ces systèmes.

36.1 Introduction.

36.1.1 Objectifs.

Cette norme permet :
- aux logiciels de localiser des fichiers et répertoires installés, et
- aux utilisateurs de localiser des fichiers et répertoires installés.

Nous pratiquons de cette manière en :
- indiquant les principes de base pour chaque zone du système de fichiers,
- indiquant les fichiers et répertoires minimaux nécessaires,
- énumérant les exceptions à ces principes,
- énumérant les cas spéciaux pour lesquels il y a des conflits "historiques".

Le document "Système de fichiers normalisé" est utilisé par :[1]
- les fournisseurs de logiciels indépendants créant des applications conformes au SFN, et qui travaillent avec des distributions non-conformes à ce dernier,
- aux développeurs de systèmes d'exploitation conformes au SFN,
- aux utilisateurs afin qu'ils maintiennent un système conforme au SFN.

36.1.2 Conventions.

La police `Courier` est utilisée pour les noms de fichiers et de répertoires.

Les composants variables des noms de fichiers sont représentés en étant entourés des caractères "`<`" et "`>`", `<ainsi>`. Les adresses de courrier électronique sont aussi entourées de "<" et ">" mais sont écrites dans la police tranditionnelle.

Les composants optionnels des noms de fichiers sont entourés entre les caractères "`[`" et "`]`" et peuvent être combinés avec la convention "`<`" et "`>`". Par exemple, si un fichier existant pouvait être trouvé avec ou sans extension, il pourrait être représenté par `<nomfichier>[.<extension>]`.

36.2 Le système de fichiers.

Cette norme suppose que le système d'exploitation sous-jacent au système de fichiers conforme SFN supporte les même caractéristiques fondamentales que celles présentes dans les systèmes de fichiers UNIX.

Il est possible de définir deux catégories indépendantes de fichiers : les fichiers partageables – non partageables et, les fichiers variables – statiques. Il devrait y avoir un lien simple et directement compréhensible allant des répertoires jusqu'aux types de données qu'ils contiennent : les répertoires peuvent être des points de montage pour d'autres systèmes de fichiers ayant des caractéristiques qui différent de celles du système de fichiers sur lesquels ils sont montés.

Une donnée partageable peut être mise en commun entre plusieurs hôtes différents ; le terme "non-partageable" désignant une donnée spécifique à un hôte particulier. Par exemple, les répertoires "home" des utilisateurs contiennent des données partageables, mais les fichiers de verrouillage de périphériques contiennent des données non-partageables.

[1] NdT : dans la présente traduction, l'expression "Système de fichiers normalisé" est représentée sous forme de l'acronyme SFN.

Les données statiques comprennent les binaires, les bibliothèques, la documentation et toute information qui ne varie pas sans l'intervention de l'administrateur système. Les données variables sont toute autre information qui, *a contrario*, peuvent être modifiées sans l'intervention de ce dernier.

Principe. La distinction entre données partageables et non partageables est nécessaire pour plusieurs raisons :

- Dans un environnement en réseau (par exemple, plus d'un hôte par site), une bonne partie des données peut être partagée entre les différentes machines par économie de place et pour faciliter les tâches de maintenance.
- Dans un environnement en réseau, certains fichiers contiennent des informations spécifiques à une seule machine. Par conséquent, ces systèmes de fichiers ne peuvent être partagés sans prendre de mesures spéciales.
- Les systèmes de fichiers de type UNIX entremêlaient les données partageables et non partageables dans la même hiérarchie, rendant difficile le partage de grandes parties du système de fichiers.

Le qualificatif "partageable" peut être utilisé pour supporter, par exemple :

- Une partition `/usr` (ou des composants de `/usr`) montés en lecture seule à travers le réseau en utilisant NFS.
- Une partition `/usr` (ou des composants de `/usr`) montés à partir d'un support en lecture seule. Un CD-ROM peut être considéré comme un système de fichiers en lecture seule partagé avec d'autres systèmes compatibles SFN, en utilisant le système de courrier comme un réseau.

La distinction entre "statique" et "variable" affecte le système de fichiers de deux manières principales :

- Puisque le répertoire / contient à la fois des données statiques et variables, il doit être monté en lecture-écriture.
- Puisque le répertoire traditionnel `/usr` contient à la fois des données variables et statiques, et vu que nous souhaitons le monter en lecture seule (voir ci-dessus), il est nécessaire de fournir une méthode pour disposer d'un répertoire `/usr` monté en lecture seule. Cet objectif est atteint par la création d'un répertoire `/var` monté en lecture-écriture (ou qui fait partie d'une autre partition en lecture-écriture, telle que /), qui remplace bien des fonctions traditionnelles de la partition `/usr`.

Le tableau ci-dessous résume la situation. Ce dernier constitue seulement un exemple associé à un système conforme SFN ; d'autres arrangements sont possibles.

	partageable	non partageable
statique	/usr /opt	/etc /boot
variable	/var/mail /var/spool/mail	/var/run /var/lock

36.3 Le système de fichiers racine

36.3.1 Objectifs.

Le contenu du système de fichiers racine doit être adéquatement agencé pour amorcer, reconstituer, rétablir et/ou réparer le système :

- Pour démarrer un système, il doit y avoir suffisamment d'information pour monter d'autres systèmes de fichiers. Ceci comprend les utilitaires, la configuration, les informations du chargeur de démarrage et d'autres données essentielles au démarrage. `/usr` et `/var` sont structurés de manière telle à être localisés sur d'autres partitions ou systèmes de fichiers.
- Pour permettre le rétablissement et/ou la réparation d'un système, les utilitaires nécessaires à l'administrateur expérimenté pour diagnostiquer et reconstruire un système endommagé doivent être présents sur le système de fichiers racine.
- Pour reconstituer un système, les utilitaires nécessaires à cette reconstitution à partir des sauvegardes système (sur disques, bandes, etc.) doivent être présents sur le système de fichiers racine.

Principe. Les principaux arguments utilisés pour équilibrer ces considérations (qui plaident pour mettre beaucoup d'informations sur le système de fichiers racine) sont les suivantes :

- Le système de fichiers est souvent monté à partir d'une unité de stockage de très petite taille.
- Le système de fichiers racine contient de nombreux fichiers de configuration spécifiques au système ; par exemple, un noyau spécifique au système, un nom d'hôte différent, etc. Ceci signifie que le système de fichiers racine n'est pas toujours partageable entre systèmes mis en réseau. Lui donner une petite taille sur des systèmes en réseau minimise l'espace perdu en fichiers non partageables sur les serveurs. Cela permet aussi d'avoir des stations de travail avec des disques durs locaux de plus petite taille.
- Bien que vous puissiez disposer d'un système de fichiers racine sur une partition de grande taille, et le remplir à votre guise, certains administrateurs préféreront gérer plusieurs partitions plus petites. Si vous disposez de nombreux fichiers installés, il se pourrait que surgissent des incompatibilités avec d'autres systèmes utilisant des systèmes de fichiers racine sur des partitions plus petites. Si vous êtes développeur, il se peut que votre hypothèse de travail se transforme en un problème pour un grand nombre d'utilisateurs.
- Les erreurs de disque qui corrompent les données sur le système de fichiers racine constituent un problème plus important que les erreurs sur toute autre partition. En cas d'un plantage système, le risque de corruption des données d'un système de fichiers racine de petite taille est diminué.

Les logiciels ne doivent jamais créer ou requérir des fichiers ou des sous-répertoires spéciaux dans le répertoire racine. D'autres endroits du SFN existent pour apporter une souplesse plus que suffisante pour l'installation de paquets. Il y a plusieurs raisons pour ne pas introduire un sous-répertoire supplémentaire dans le répertoire racine :

- l'introduction d'un sous-répertoire prend une place plus ou moins importante sur la partition racine alors que l'administrateur cherche à en mini-

miser la taille pour des raisons de performances ou de sécurité.
- l'introduction d'un sous-répertoire escamote la discipline que l'administrateur système a mis en place pour installer le système de fichiers parmi les différents volumes montables.

36.3.2 Conditions.

Il est requis que les répertoires suivants, ou les liens symboliques pointant vers ces répertoires se trouvent dans / :

/	———	le répertoire racine
	_ bin	commandes binaires essentielles
	_ boot	fichiers statiques du chargeur de démarrage
	_ dev	fichiers de périphériques
	_ etc	fichiers de configuration de la machine
	_ lib	bibliothèques partagées et modules du noyau
	_ mnt	point de montage des fichiers temporaires
	_ opt	paquets logiciels additionnels
	_ sbin	binaires des commandes systèmes
	_ tmp	fichiers temporaires
	_ usr	structure secondaire
	_ var	données variables

Chaque répertoire mentionné ci-dessus est décrit en détail dans les sections qui suivent. /usr et /var sont l'objet d'une section complète dans le présent document et ce, en raison de leur complexité.

36.3.3 Options spécifiques.

Les répertoires suivants, ou les liens symboliques pointant vers ces répertoires, doivent se trouver dans /, si le sous-système correspondant est installé :

/	———	le répertoire racine
	_ home	répertoires personnels des utilisateurs (optionnel)
	_ lib<qual>	bibliothèques partagées essentielles de format alternatif (optionnel)
	_ root	répertoire personnel du superutilisateur (optionnel)

36.3.4 /bin : répertoire des commandes binaires essentielles à l'utilisateur (usage pour tous les utilisateurs).

36.3.4.1 Objectifs.

/bin contient des commandes qui peuvent être utilisées aussi bien par l'administrateur que par les utilisateurs, mais qui sont nécessaires lorsque d'autres

systèmes de fichiers ne sont pas montées (par exemple, en mode mono-utilisateur –*single user mode*). Il peut aussi comprendre des commandes qui sont utilisées indirectement par des scripts.[2]

36.3.4.2 Conditions.

Il ne peut y avoir de sous-répertoires dans `/bin`. Les commandes suivantes ou les liens symboliques pointant vers ces commandes doivent se trouver dans le répertoire `/bin`.

`cat`	utilitaire de concaténation de fichiers sur la sortie standard
`chgrp`	utilitaire de changement de propriétés sur les groupes
`chmod`	utilitaire de changement des droits d'accès aux fichiers
`chown`	utilitaire de changement du propriétaire et du groupe d'un fichier
`cp`	utilitaire de copie des fichiers et des répertoires
`date`	utilitaire d'affichage de la date et de l'heure
`dd`	utilitaire de conversion et de copie d'un fichier
`df`	utilitaire pour décrire l'utilisation de l'espace disque pour un système de fichiers donné
`dmesg`	utilitaire d'affichage et de contrôle des messages du tampon du noyau
`echo`	commande d'affichage d'une ligne de texte
`false`	commande n'ayant pas d'action, mais renvoyant un code de retour valant 1, signifiant "échec"
`hostname`	commande d'affichage ou de modification d'un nom d'hôte
`kill`	utilitaire permettant d'envoyer des signaux aux processus
`ln`	commande pour faire des liens entre fichiers
`login`	commande pour débuter une session sur un système
`ls`	utilitaire pour afficher le contenu des répertoires

[2] Les binaires de commandes qui ne sont pas vraiment essentielles au point d'être installés dans `/bin` doivent figurer dans `/usr/bin`. Les commandes qui sont utiles aux utilisateurs non-*root* (le Système X-Window, `chsh`, etc.) ne sont généralement pas considérées comme essentielles pour figurer sous la partition racine.

`mkdir`	utilitaire de création de répertoires
`mknod`	commande pour créer des fichiers spéciaux de type caractères ou blocs
`more`	commande d'affichage d'un texte sous forme d'une page
`mount`	commande pour monter un système de fichiers
`mv`	commande pour déplacer ou renommer des fichiers
`ps`	utilitaire pour rapporter l'état d'un processus
`pwd`	utilitaire pour afficher le nom du répertoire courant (chemin absolu)
`rm`	utilitaire pour supprimer des fichiers ou des répertoires
`rmdir`	utilitaire pour supprimer des répertoires vides
`sed`	l'éditeur de flux "sed"
`sh`	le shell Bourne
`stty`	commande de chargement et d'affichage des caractéristiques des lignes d'un terminal
`su`	utilitaire de modification de l'ID d'un utilisateur
`sync`	utilitaire pour nettoyer les tampons d'un système de fichiers.
`true`	commande ne renvoyant qu'un code de retour valant 0 et signifiant "réussite"
`umount`	utilitaire de démontage des système de fichiers
`uname`	commande d'affichage des informations système

Si `/bin/sh` n'est pas un véritable shell Bourne, il doit y avoir des liens physiques ou symboliques pointant vers la commande du vrai shell.

Les commandes `[` et `test` doivent être placées ensemble soit dans `/bin`, soit dans `/usr/bin`.

Principe. Par exemple, `bash` se comporte différemment selon qu'on l'appelle en tant que `sh` ou `bash`. L'utilisation d'un lien symbolique permet aussi aux utilisateurs de vérifier aisément que `/bin/sh` n'est pas un vrai shell Bourne.

L'obligation d'avoir les binaires de `[` et de `test` (même s'ils sont mis en oeuvre de manière interne au shell) est conforme au standard POSIX.2.

36.3.4.3 Options spécifiques.

Les programmes suivant, ou les liens symboliques vers ces programmes, doivent se trouver dans `/bin` si le sous-système qui leur correspond est installé :

`csh`	shell C (optionnel)
`ed`	éditeur "ed" (optionnel)
`tar`	utilitaire d'archivage tar (optionnel)
`cpio`	utilitaire d'archivage cpio (optionnel)
`gzip`	utilitaire de compression GNU (optionnel)
`gunzip`	utilitaire de décompression GNU (optionnel)
`zcat`	utilitaire de décompression GNU (optionnel)
`netstat`	utilitaire de statistiques du réseau (optionnel)
`ping`	utilitaire ICMP de test du réseau (optionnel)

Principe. Si les programmes `gunzip` et `zcat` sont présents, il doit y avoir un lien symbolique ou physique vers `gzip`. `/bin/csh` peut être un lien symbolique vers `/bin/tcsh` ou `/usr/bin/tcsh`.

Les commandes `tar`, `gzip` et `cpio` ont été ajoutées pour permettre de restaurer un système (pourvu que `/` soit resté intact).

Inversément, s'il certain qu'aucune remise en état du système depuis la partition racine n'est nécessaire, ces binaires peuvent être ignorés (par exemple, une partition racine en ROM, montage de `/usr` *via* NFS). Si une remise en état via le réseau est prévue, ftp ou tftp doit être disponible sur la partition racine (avec tous les composants nécessaires pour pouvoir établir une connexion ftp).

36.3.5 /boot : répertoire des fichiers statiques du chargeur de démarrage.

36.3.5.1 Objectifs.

Ce répertoire contient tout ce qui est nécessaire au processus d'amorçage sauf les fichiers de configuration et l'installateur "map". Donc, `/boot` stocke les données qui sont utilisées avant que le noyau ne commence à exécuter les programmes en mode utilisateur. Ceci comprend les secteurs d'amorçage maître sauegardés, les fichiers de la table des secteurs, et toute autre donnée qui n'est pas directement modifiée manuellement.[3]

[3] Les programmes nécessaires pour organiser le chargeur de démarrage pour qu'il soit capable d'amorcer un fichier doivent être présents dans `/sbin`.

36.3.5.2 Options spécifiques.

Le noyau du système d'exploitation doit être placé soit dans / ou dans `/boot`.[4]

36.3.6 /dev : répertoire des fichiers de périphériques.

36.3.6.1 Objectifs.

Le répertoire `/dev` contient les fichiers de périphériques et les fichiers spéciaux.

36.3.6.2 Options spécifiques.

S'il faut que des périphériques dans `/dev` soient crées, `/dev` contiendra la commande MAKEDEV, qui crée autant de périphériques que nécessaire. Il peut aussi y avoir un fichier `MAKEDEV.local` pour les périphériques locaux.

Si cela s'avère nécessaire, MAKEDEV doit pourvoir créer tout fichier de périphérique susceptible d'être trouvé sur le système, et pas seulement ceux qu'une implémentation particulière installe.

36.3.7 /etc : répertoire de configuration du système utilisé.

36.3.7.1 Objectifs.

`/etc` contient les fichiers de configuration et les répertoires qui sont spécifiques au système en fonctionnement.[5]

36.3.7.2 Conditions.

Aucun binaire ne peut être placé sous `/etc`. Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent se trouver dans `/etc` :

`/etc`	`————-`	le répertoire de configuration
	`\_ opt`	configuration de `/opt`

36.3.7.3 Options spécifiques.

Les répertoires suivants, ou les liens symboliques vers les répertoires doivent se trouver dans /etc, si le sous-système correspondant est installé :

---

[4] Sur certaines machines de type i386, il peut s'avérer nécessaire que `/boot` soit placé sur une partition séparée se trouvant sous le 1024^{eme} cylindre du périphérique d'amorçage en raison de contrainte matérielle.
Certains systèmes MIPS exigent une partition `/boot` montée avec un système de fichiers MS-DOS ou quelqu'autre système de fichiers accessible par un progiciel (*firmware*). Ceci peut conduire à des restrictions par rapport aux noms de fichiers utilisables à l'intérieur de `/boot` (çà ne vaut que pour les systèmes concernés).

[5] L'ensemble des scripts invoqués pendant le démarrage peut être de type System V, BSD ou autre. Des spécifications ultérieures dans ce domaine peuvent être ajoutées aux versions future de la présente norme.

/etc	———	le répertoire de configuration
	_ X11	configuration du système X-Window (optionnel)
	_ sgml	configuration de SGML et de XML (optionnel)

Les fichiers suivants, ou les liens symboliques vers les fichiers, doivent se trouver dans /etc si le sous-système correspondant est installé :[6]

csh.login	fichier d'initialisation au niveau système des connexions du shell (optionnel)
exports	liste de contrôle d'accès aux système de fichiers NFS (optionnel)
fstab	information statique à propos du système de fichiers (optionnel)
ftpusers	liste de contrôle d'accès utilisateur du démon FTP (optionnel)
gateways	fichier qui liste les passerelles pour le routage (optionnel)
gettydefs	caractéristiques des terminaux utilisées par getty (optionnel)
group	fichier des groupes utilisateurs (optionnel)
host.conf	fichier de configuration de résolution (optionnel)
hosts	information statique à propos des noms d'hôtes (optionnel)
hosts.allow	fichier d'accès des hôtes pour les interfaces (*wrappers*) TCP (optionnel)
hosts.deny	fichier d'accès des hôtes pour les interfaces (*wrappers*) TCP (optionnel)
hosts.equiv	liste d'hôtes de confiance pour rlogin, rsh, rcp (optionnel)
hosts.lpd	liste d'hôtes de confiance pour lpd (optionnel)
inetd.conf	fichier de configuration d'inetd (optionnel)
inittab	fichier de configuration pour init (optionnel)
issue	message de pré-connexion et fichier d'identification (optionnel)
id.so.conf	liste des répertoires supplémentaires pour rechercher les bibliothèques partagées (optionnel)

[6] Les systèmes qui utilisent des mots de passe cryptés (*shadow*) devront avoir des fichiers de configuration supplémentaires dans **/etc** (**/etc/shadow** et autres) et des programmes dans **/usr/bin** (**useradd**, **usermod**, et autres).

`motd`	message du jour d'après-connexion (optionnel)
`mtab`	information dynamique sur les systèmes de fichiers (optionnel)
`mtools.conf`	fichier de configuration de mtools (optionnel)
`networks`	information statique sur les noms de réseau (optionnel)
`passwd`	fichier de mots de passe (optionnel)
`printcap`	base de données pour les imprimantes lpd (optionnel)
`profile`	fichier d'initialisation au niveau système pour les connexions du shell sh (optionnel)
`protocols`	liste du protocole IP (optionnel)
`resolv.conf`	fichier de configuration de résolution de noms (optionnel)
`rpc`	liste du protocole RPC (optionnel)
`securetty`	contrôle d'accès TTY pour la connexion du superutilisateur (optionnel)
`services`	noms des ports pour les services internet (optionnel)
`shells`	noms de chemin desshelles de connexion valides (optionnel)
`syslog.conf`	fichier de configuration pour syslogd (optionnel)

mtab ne doit pas adopter la nature statique de `/etc`, ceci pour des raisons historiques.[7]

36.3.7.4 /etc/opt : fichiers de configuration pour /opt.

Buts. Les fichiers de configuration spécifiques aux paquets logiciels d'applications supplémentaires doivent être installés dans le répertoire `/etc/opt/<paquet>`, avec `<paquet>` désignant la sous-arborescence du répertoire `/opt` où les données statiques du paquet sont stockées.

Conditions. Aucune structure n'est imposée concernant l'arrangement interne de `/etc/opt/<paquet>`.

Si un fichier de configuration doit résider dans un autre emplacement de manière telle que le paquet ou le système fonctionne correctement, ce fichier peut être placé à un autre endroit que `/etc/opt/<paquet>`.

Principe. Reportez-vous à la section relative à `/opt`.

[7] Sur certains systèmes Linux, il peut y avoir un lien symbolique vers `/proc/mounts`, auquel cas l'exception que nous venons de décrire n'est pas de mise.

36.3.7.5 /etc/X11 : fichiers de configuration pour X-Window (optionnel).

Buts. `/etc/X11` est l'emplacement pour toute configuration de X11. Ce répertoire est nécessaire pour permettre le contrôle local si `/usr` est monté en lecture seule.

Conditions. Les fichiers suivants, ou les liens symboliques vers ces fichiers, doivent être logés dans `/etc/X11` si le sous-système correspondant est installé :[8]

`Xconfig`	fichier de configuration des premières versions d'XFree86 (optionnel)
`XF86Config`	fichiers de configuration pour les versions 3 et 4 d'XFree86 (optionnel)
`Xmodmap`	fichier de modifications du clavier d'X11 (optionnel)

Les sous-répertoires de `/etc/X11` peuvent inclure ceux d'`xdm` et de tous les autres programmes qui les requièrent (certains gestionnaires de fenêtres, par exemple).[9]

Nous recommandons que ces gestionnaires de fenêtres avec un seul fichier de configuration intitulé `.*wmrc` nomment ce dernier : `system.*wmrc` (à moins qu'il n'y ait une dénomination très largement répandue). Nous recommandons de ne pas utiliser de sous-répertoire. Tous les sous-répertoires des gestionnaires de fenêtres doivent être nommés de manière identique au binaire du gestionnaire utilisé réellement.

36.3.7.6 /etc/sgml : fichiers de configuration pour SGML et XML (optionnel).

Objectifs. Les fichiers de configuration génériques définissant les paramètres de haut-niveau des systèmes SGML ou XML sont installés dans `/etc/sgml`. Les fichiers avec des noms `*.conf` indiquent des fichiers de configuration. Les fichiers avec des noms de type `*.cat` sont les catalogues centralisés DTD,[10] contenant des références à tous les autres catalogues nécessaires pour utiliser DTD. Le super-catalogue appelé `catalog` référence tous les catalogues centralisés.

36.3.8 /home : répertoire des utilisateurs (optionnel).

36.3.8.1 Objectifs.

`/home` est un répertoire tout-à-fait classique, mais il est clairement un système de fichiers localisé très spécifiquement.[11] Selon la machine hôte, l'emplace-

[8] NdT : notez qu'au cours de la traduction, le système XFree86 a été abandonné au profit d'Xorg pour des raisons de licence.

[9] `/etc/X11/xdm` contient les fichiers de configuration d'`xdm`. Il s'agit de la plupart des fichiers antérieurement placés dans `/usr/lib/X11/xdm`. Certaines variables locales d'`xdm` sont stockées dans `/var/lib/xdm`.

[10] NdT : voir page 415 pour la définition.

[11] Diverses personnes placeront les comptes d'utilisateurs à des endroits différents. Cette section se contente de suggérer un emplacement type pour les répertoires "home" des utilisateurs ; néanmoins, nous recommandons que les distributions conformes au SFN utilisent `/home`

ment peut différé. Donc, aucun programme ne devrait se trouver à cet endroit.[12]

36.3.9 /lib : bibliothèques partagées et modules essentiels du noyau.

36.3.9.1 Objectifs.

Le répertoire `/lib` contient les images de bibliothèques partagées nécessaires pour l'amorçage du système et permettre l'exécution de commandes du système de fichiers racine, c'est-à-dire les binaires de `/bin` et `/sbin`.[13]

36.3.9.2 Conditions.

Au minimum, les modèles suivant de noms de fichiers sont nécessaires (il peut s'agir des fichiers ou des liens symboliques pointant sur les noms) :

`libc.so.*`	la bibliothèque C liée dynamiquement (optionnel)
`ld*`	le chargeur / l'éditeur de liens de durée d'exécution (optionnel)

Si un préprocesseur **C** est installé, `lib/cpp` doit être un référence à ce dernier et ce, pour des raisons historiques.[14]

36.3.9.3 Options spécifiques.

Les répertoires suivants, ou les liens symboliques pointant vers ces derniers, doivent se trouver dans `/lib`, si le sous-système correspondant est installé :

`/lib`	———	bibliothèques partagées essentielles et modules du noyau
	`\_ modules`	modules chargeables du noyau (optionnel)

---

pour héberger les répertoires d'utilisateurs.

Sur les petits systèmes, chaque répertoire d'utilisateur est typiquement un des sous-répertoires `/home`, tels que : `home/smith`, `/home/torvalds`, `/home/operator`, etc. Sur des systèmes plus importants (en particulier, lorsque les répertoires `/home` sont partagés sur des systèmes différents), il est pertinent de subdiviser les répertoires personnels des utilisateurs. La subdivision peut être réalisée en utilisant des répertoires tels que `/home/staff`, `/home/guest`, `/home/students`, etc.

[12] Si vous voulez trouver le répertoire personnel d'un utilisateur, vous devriez utiliser la fonction `getpwent` (3) plutôt que de vous baser sur `/etc/passwd` car l'information relative à l'utilisateur peut être stockée sur un système distant grâce à l'usage de NIS.

[13] Les bibliothèques partagées, seulement nécessaires aux binaires de `/usr` (telles que les binaires de X Window), ne doivent pas se trouver dans `/lib`. Seules les bibliothèques partagées nécessaires à l'exécution des binaires de `/bin` et `/sbin` doivent de trouver dans `/lib`. En particulier, la bibliothèque `libm.so.*` peut être placée dans `/usr/lib` si elle n'est pas requise par un quelconque programme de `/bin` ou `/sbin`.

[14] L'emplacement usuel de cette bibliothèque est `/usr/lib/gcc-lib/<cible>/<version>/cpp`. `lib/cpp` peut soit pointer vers cette bibliothèque, soit vers tout autre référence à ce binaire qui se trouve dans le système de fichiers (par exemple, `/usr/bin/cpp` est souvent utilisé).

36.3.10 /lib<qual> : bibliothèques partagées essentielles au format alternatif (répertoire optionnel)

36.3.10.1 Objectifs.

Il peut y avoir une ou plusieurs variantes du répertoire `/lib` sur les systèmes qui supportent plus d'un format binaire exigeant des bibliothèques séparées.[15]

36.3.10.2 Conditions.

Si un ou plusieurs de ces répertoires existe(nt), les conditions s'appliquant à leur contenu sont les mêmes que celles pour le répertoire `/lib`, sauf que `/lib<qual>/cpp` n'est pas requise.[16]

36.3.11 /mnt : points de montage des systèmes de fichiers montés temporairement.

36.3.11.1 Objectifs.

Ce répertoire est fourni de manière telle que l'administrateur puisse monter temporairement un système de fichiers si cela s'avère nécessaire. Le contenu de ce répertoire est local et ne devrait pas affecter la manière avec laquelle un programme est exécuté.

Ce répertoire ne doit pas être utilisé par le programme d'installation : un répertoire temporaire approprié non-utilisé par le système devrait être employé.

36.3.12 /opt : paquets logiciels pour applications additionnelles.

36.3.12.1 Objectifs.

`/opt` est réservé à l'installation de paquets additionnels.

Un paquet à installer dans `/opt` doit placer ses fichiers statiques dans une arborescence séparé intitulée `/opt/<paquet>`, où `<paquet>` est le nom du paquet logiciel.

36.3.12.2 Conditions.

Les répertoires `/opt/bin`, `/opt/doc`, `/opt/include`, `/opt/info`, `/opt/lib` et `/opt/man` sont réservés à l'usage de l'administrateur d'un système local.

/opt	———	paquets logiciels supplémentaires
	_ <paquet>	répertoires et fichiers du paquet additionnel <paquet>

Les paquets peuvent fournir des fichiers "frontaux" (*front-end*) prévus pour être placés par l'administrateur –par copie ou via un lien– dans les répertoires cités

[15] Ceci est commun dans le cas des supports à 32 et à 64 bits pour des systèmes exploitant des formats binaires multiples, mais qui requièrent des bibliothèques de mêmes noms. Dans un tel cas, `/lib32` et `/lib64` peuvent être les répertoires des bibliothèques, et `/lib` un lien symbolique à l'une de ces dernières.

[16] `/lib<qual>/cpp` est encore autorisée : ceci pour le cas où `/lib` et `/lib<qual>` sont les mêmes (l'une étant un lien symbolique vers l'autre).

ci-dessus. Cependant, ces paquets doivent fonctionner normalement en absence de ces répertoires.

Les programmes qui sont invoqués par les utilisateurs doivent être placés dans le répertoire `/opt/<paquet>/bin`. Si le paquet comprend des pages de manuel UNIX, celles-ci doivent se trouver dans `/opt/<paquet>/man` et respecter la même sous-structure que `/usr/share/man`.

Les fichiers des paquets qui sont variables (modifications associées à une utilisation normale) doivent être installés dans `/var/opt`. Voir la section relative à `/var/opt` pour davantage d'information.

Les fichiers de configuration spécifiques à l'hôte doivent être installés dans `/etc/opt`. Voir la section relative à `/etc` pour des informations supplémentaires.

Aucun autre fichier associé à un paquet donné ne peut se trouver en dehors des répertoires `/opt`, `/var/opt` et `/etc/opt` sauf les paquets qui doivent résider à des endroits spécifiques dans l'arborescence du système de fichiers de manière à fonctionner correctement. Par exemple, les fichiers de verrouillage des périphériques doivent résider dans `/var/lock` et les périphériques dans `/dev`.

Les distributions peuvent installer les logiciels dans `/opt`, mais ne doivent pas modifier ou supprimer les fichiers installés par l'administrateur du système local sans le consentement de cet administrateur.

Principe. L'utilisation de `/opt` pour les logiciels supplémentaires est une pratique bien établie dans la communauté UNIX. L'interface d'applications binaires de "System V [AT&T 1990]", basée sur "System V Interface Definition (Third Edition)", fournit une structure `/opt` très similaire à celle définie dans ce document.

Le standard intitulé "Intel Binary Compatibilty Standard v. 2 (iBCS2)" fournit également une structure analogue à celle de `/opt`.

Généralement, toute donnée requise pour supporter un paquet sur un système doit se trouver dans `/opt/<paquet>`, en ce compris les fichiers prévus pour être copiés dans `/etc/opt/<paquet>` et `/var/opt/<paquet>` aussi bien que les répertoires réservés dans `/opt`.

Les restrictions mineures sur les distributions utilisant `/opt` sont nécessaires car des conflits sont possibles entre les logiciels installés en mode local et ceux de la distribution, en particulier dans le cas des noms de chemins "établis" de certaines bibliothèques binaires.

36.3.13 /root : répertoire "home" (optionnel) du super-utilisateur.

36.3.13.1 Objectifs.

Le répertoire du compte "root" peut être fixé par les développeurs ou localement, mais `/root` est l'emplacement recommandé.[17]

[17] Si le répertoire du compte "root" n'est pas logé sur la partition racine, il sera nécessaire de s'assurer qu'il est, par défaut, dans `/` si on ne peut le localiser.

Nous recommandons de ne pas utiliser le compte "root" pour des tâches qui peuvent être réalisées sous un compte utilisateur non-privilégié, et que le compte root ne soit utiliser que pour de l'administration système. Pour cette raison, nous recommandons que les sous-répertoires de courrier électronique et d'autres applications n'apparaissent pas dans le compte "root" et aussi, que les messages électroniques d'administration (root, postmaster, webmaster) soient redirigés vers un utilisateur approprié.

36.3.14 /sbin : répertoire des binaires du système.

36.3.14.1 Objectifs.

Les utilitaires exploités dans le cadre de l'administration système (et d'autres commandes exécutées sous root seulement) sont stockés dans `/sbin`, `/usr/sbin` et `/usr/local/sbin`. `/sbin` contient les binaires essentiels à l'amorçage, à la récupération, à la reconstitution et/ou à la réparation du système en addition aux binaires de `/bin`. [18]

Les programmes exécutés après que `/usr` soit monté (quand il n'y a pas de problèmes) sont généralement placés dans `/usr/sbin`. Les programmes d'administration système installés en local devraient se trouver dans `/usr/local/sbin`.[19]

36.3.14.2 Conditions.

La commande suivante, ou le lien symbolique vers cette commande, doit se trouver dans `/sbin`.

`shutdown`	commande pour éteindre le système

36.3.14.3 Options spécifiques.

Les fichiers suivants, ou les liens symboliques vers ces fichiers, doivent être placés dans `/sbin` si le sous-système correspondant est installé :

`fastboot`	réamorce le système sans vérifications des disques (optionnel)
`fasthalt`	arrête le système sans vérification des disques (optionnel)
`fdisk`	gestionnaire de tables de partition (optionnel)
`fsck`	utilitaire de réparation et de vérification du système de fichiers (optionnel)
`fsck.*`	idem mais pour un système particulier de fichiers (optionnel)

[18] A l'origine, les binaires `/sbin` se trouvaient dans `/etc`.

[19] La décision de mettre tel ou tel programme dans les répertoires de `/sbin` est simple à prendre : si un utilisateur normal (mais pas l'administrateur) exécute un programme directement, alors ce dernier doit être placé dans `/bin`. Les utilisateurs ordinaires ne devraient jamais avoir les répertoires de `/sbin` dans leur PATH.

Par exemple, des fichiers tels que `chfn` que les utilisateurs utilisent occasionnellement seront placés dans `/usr/bin`. En revanche, `ping` (bien qu'il soit absolument nécessaire à l'administrateur pour la récupération et le diagnostic du réseau) est souvent employé par les utilisateurs et donc, doit rester dans `/bin`.

Nous recommandons que les utilisateurs aient les droits en exécution et en lecture pour tous les programmes de `/sbin`, sauf peut-être les programmes setuid et setgid. La séparation entre `/bin` et `/sbin` n'a pas été introduite pour des raisons de sécurité ou pour empêcher les utilisateurs de "voir" le système d'exploitation, mais pour fournir un bon partitionnement entre les binaires que chacun emploie et ceux que l'administrateur utilise au premier chef pour ses tâches d'administration. En termes de sécurité, il n'y a pas d'avantage à interdir aux utilisateurs l'accès à `/sbin`.

`getty`	programme getty (optionnel)
`halt`	commande d'arrêt du système (optionnel)
`ifconfig`	programme de configuration des interfaces réseau (optionnel)
`init`	processus initial (optionnel)
`mkfs`	commande pour construire un système de fichiers (optionnel)
`mkfs.*`	idem pour un système de fichiers particulier (optionnel)
`mkswap`	commande pour installer une zone de dégagement *swap* (optionnel)
`reboot`	commande de réamorçage du système (optionnel)
`route`	utilitaire de la table de routage (optionnel)
`swapon`	permet la pagination et l'activation de la partition de dégagement (optionnel)
`swapoff`	inversément (optionnel)
`update`	démon de nettoyage périodique des tampons du système de fichiers (optionnel)

36.3.15 /tmp : répertoire de fichiers temporaires.

36.3.15.1 Objectifs.

Le répertoire `/tmp` doit être disponible aux programmes qui requièrent un espace pour y déposer des fichiers temporaires.

On ne peut supposer que les fichiers ou répertoires de `/tmp` sont préservés entre deux invocations d'un programme donné.

La norme intitulée IEEE P1003.2 (POSIX, partie 2) impose des conditions similaires à celles de la section précédente.

Bien que les données enregistrées dans `/tmp` puissent être supprimées intentionnellement, il est recommandé que les fichiers et répertoires de `/tmp` puissent être supprimés chaque fois que le système est amorcé.

Le SFN ajoute cette recommandation sur la base d'une pratique courante qui s'est perpétuée, mais il n'en fait pas une condition parce que l'administration système n'est pas une partie intégrante du champ de cette norme.

36.4 La hiérarchie de /usr.

36.4.1 Objectifs.

`/usr` est la deuxième section majeure du système de fichiers. `/usr` est partageable et constitué de donnés en lecture seule. Ceci signifie que `/usr` devrait être partageable entre différentes machines conformes au SFN et que rien n'y

doit être écrit. Toute information variable dans le temps ou qui est spécifique de la machine doit être écrite ailleurs.

Les paquets logiciels ne doivent pas utiliser un sous-répertoire direct de l'arborescence de `/usr`.

36.4.2 Conditions.

Les répertoires suivants, ou les liens vers ces répertoires, sont requis dans `/usr` :

/usr	———	Deuxième système
	_ bin	principales commandes des utilisateurs
	_ include	fichiers d'en-têtes inclus dans les programmes C
	_ lib	bibliothèques
	_ local	répertoire local (vide après l'installation principale)
	_ sbin	binaires non-vitaux du système
	_ share	données indépendantes de l'architecture

36.4.3 Options spécifiques.

/usr	———	Deuxième système
	_ X11R6	Système X Window, version 11, sous-version 6 (optionnel)
	_ games	jeux et binaires de programmes éducatifs (optionnel)
	_ lib<qual>	bibliothèques de format alternatif (optionnel)
	_ src	code source (optionnel)

Une exception est faite pour le système X Window en raison d'une pratique largement acceptée.

Comme indiqué dans l'encadré ci-dessous, les liens symboliques pointant vers les répertoires doivent être présents. Ceci est dû à la nécessité de préserver la compatibilité avec les anciens systèmes jusqu'à ce qu'on puisse considérer que tous les systèmes utilisent `/var`.

```
/usr/spool -> /var/spool
/usr/tmp -> /var/tmp
/usr/spool/locks -> /var/lock
```

Lorsqu'un des liens sus-mentionnés n'est plus nécessaire, il peut être supprimé.

36.4.4 /usr/X11R6 : le système X-Window, version 11, variante 6 (optionnel).

36.4.4.1 Objectifs.

Cette structure de répertoires et de fichiers est réservée au système X Window, version 11, sous-version 6, et aux fichiers qui s'y rapportent.

Pour simplifier le problème, et rendre XFree86 compatible avec le système X Window sur d'autres systèmes, les liens symboliques suivants doivent être présents si /usr/X11R6 existe :

```
/usr/bin/X11 -> /usr/X11R6/bin
/usr/lib/X11 -> /usr/X11R6/lib/X11
/usr/include/X11 -> /usr/X11R6/include/X11
```

En général, les logiciels ne doivent pas être installés ou gérés via les liens symboliques. Ces derniers sont destinés aux utilisateurs seulement. La difficulté est liée aux versions antérieures du système X-Window. Dans les périodes de transition, il est impossible de déterminer quelle version est utilisée.

36.4.4.2 Options spéciales.

Les données spécifiques à l'hôte dans `/usr/X11R6/lib/X11` devraient être considérées comme appartenant à un fichier de démonstration. Les applications réquerrant de l'information à propos de l'hôte courant doivent se référer à un fichier de configuration de `/etc/X11`, qui peut être lié au fichier `/usr/lib/X11R6/lib`.[20]

36.4.5 /usr/bin : la plupart des commandes d'utilisateurs.

36.4.5.1 Objectifs.

Ceci est le répertoire primaire des commandes exécutables du système.

36.4.5.2 Options spécifiques.

Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent se trouver dans `/usr/bin`, si le sous-système correspondant est installé :

`/usr/bin`	———	Binaires non nécessaires au mode mono-utilisateur
	`\_ mh`	commandes pour le système de manipulation du courrier électronique MH (optionnel)

`/usr/bin/X11` doit être un lien symbolique vers `/usr/X11R6/bin` si ce dernier existe.

[20] Des exemples des fichiers de configuration sont `Xconfig`, `XF86Config` ou `system.twrc`. Ndt : il est à noter que le système XFree86 est considéré comme obsolète depuis ~2004. Le système Xorg le remplace. Le fichier-type de configuration d'Xorg est `/etc/X11/xorg.conf`.

Les fichiers suivants, ou les liens symboliques pointant vers ces fichiers, doivent se trouver dans `/usr/bin`, si le sous-système correspondant est installé :

`perl`	Acronyme de *Practical Extraction and Report Language* (optionnel)
`python`	langage interprété Python (optionnel)
`tclsh`	shell simple comprenant l'interpréteur Tcl (optionnel)
`wish`	shell de fenêtrage simple Tcl/Tk (optionnel)
`expect`	programme de dialogue interactif (optionnel)

Du fait que les interpréteurs de scripts shell (invoqués avec l'expression `#!<chemin>` sur la première ligne du script) ne peuvent utiliser d'eux-mêmes un chemin, il est avantageux de normaliser leur emplacement. Les interpréteurs des shells Bourne et C- sont déjà dans `/bin`, mais Perl, Python et Tcl sont souvent dans des emplacements différents. Il peut y avoir des liens symboliques pointant vers leur emplacement réel.

36.4.6 /usr/include : répertoires pour les fichiers "include" normalisés.

36.4.6.1 Objectifs.

Ceci est l'emplacement où devraient être placés les fichiers "include" associés au langage de programmation **C** et les fichiers d'usage général pour le système.

36.4.6.2 Options spéciales.

Les répertoires suivants, ou les liens symboliques pointant vers ces répertoires, doivent être dans `/usr/include`, si le sous-système correspondant est installé :

`/usr/include`	———	Fichiers "include"
	`\_ bsd`	fichiers "include" compatibles avec BSD (optionnel)

Le lien symbolique `/usr/include/X11` doit pointer vers `/usr/X11R6/include/X11`, si ce dernier existe.

36.4.7 /usr/lib : bibliothèques pour la programmation et les paquets.

36.4.7.1 Objectifs.

`/usr/lib` comprend les fichiers-objets, les biblothèques et les binaires internes pour lesquels il n'est pas prévu que les utilisateurs ou des scripts de shell

les exécutent directement.[21]

Les applications peuvent utiliser un sous-répertoire unique sous `/usr/lib`. Si une application utilise un sous-répertoire, toutes les données dépendant de l'architecture, exclusivement utilisées par l'application doivent être placées dans ce sous-répertoire.[22]

36.4.7.2 Options spécifiques.

Pour des raisons historiques, `/usr/lib/sendmail` doit être un lien symbolique vers `/usr/sbin/sendmail` si ce dernier existe.[23]

Si `/lib/X11` existe, `/usr/lib/X11` doit être un lien symbolique vers `/lib/X11` ou à tout vers quoi le lien symbolique `/lib/X11` pointe.[24]

36.4.8 /usr/lib<qual> : bibliothèques (optionnelles) pour les formats alternatifs.

36.4.8.1 Objectifs.

`/usr/lib<qual>` joue le même rôle que `/usr/lib` relativement à un format binaire alternatif, si ce n'est que les liens symboliques `/usr/lib<qual>/sendmail` et `/usr/lib<qual>/X11` ne sont pas requis.[25]

36.4.9 /usr/local : hiérarchie locale.

36.4.9.1 Objectifs.

L'arborescence `/usr/local` est utilisée par l'administrateur lorsqu'il faut installer des logiciels localement (c'est-à-dire sur une machine). Il est nécessaire d'être attentif à ne pas écraser de logiciels lors d'une mise à jour. Ce répertoire peut être utilisé pour les programmes et les données partageables au sein d'un groupe d'utilisateurs de la machine hôte, ces programmes et données ne devant pas figurer dans `/usr`.

Les logiciels installés localement doivent être placés dans `/usr/local` plutôt que dans `/usr` sauf s'ils sont destinés à remplacer ou à mettre à jour un logiciel originellement présent dans `/usr`.[26]

---

[21] Divers fichiers statiques, ainsi que des sous-répertoires, indépendants de l'architecture et spécifiques à des applications doivent se trouver dans `/usr/share`.

[22] C'est le cas, par exemple, pour le sous-répertoire `perl5`, pour les modules et bibliothèques de Perl5.

[23] Certaines commandes telles que `makewhatis` et `sendmail` ont été traditionnellement placées dans `/usr/lib`. La commande `makewhatis` est un binaire interne et doit être placée dans le répertoire des binaires; les utilisateurs accèdent seulement à `catman`. Les nouveaux binaires de `sendmail` sont à présent placés par défaut dans `/usr/sbin`. De plus, les systèmes utilisant un agent de tranfert de courrier compatible avec `sendmail` doivent fournir `/usr/sbin/sendmail` en tant que lien symbolique vers l'exécutable approprié.

[24] Les données spécifiques à l'hôte pour un système X Window ne doivent pas être stockées dans `/usr/lib/X11`. Les fichiers de configuration spéficiques à l'hôte tels qu'`Xconfig`, `XF86Config` (NdT : ou `xorg.conf`) doivent être enregistrés dans `/etc/X11`. Ceci s'adresse aussi aux données de configuration telles que celles du système `system.twmrc`, même s'il y a un lien symbolique vers un fichier de configuration plus général (probablement dans `/usr/X11R6/lib/X11`).

[25] Dans le cas où `/usr/lib` et `/usr/lib<qual>` sont les mêmes (l'un étant un lien symbolique vers l'autre), ces fichiers et les sous-répertoires associés devront exister.

[26] Les logiciels placés dans `/` ou `/usr` peuvent être écrasés par des mises à jour du système

36.4.9.2 Conditions.

Les répertoires suivants, ou les liens symboliques pointant vers ces répertoires, doivent se trouver dans `/usr/local` :

`/usr/local`	———	arborescence locale
	`\_ bin`	binaires locaux
	`\_ games`	binaires des jeux installés localement
	`\_ include`	fichiers d'en-têtes **C** locaux
	`\_ lib`	bibliothèques locales
	`\_ man`	pages de manuel locales
	`\_ sbin`	binaires du système installés localement
	`\_ share`	arborescence locale, indépendante de l'architecture
	`\_ src`	code source local

Aucun autre répertoire, en dehors de ceux mentionnés dans le tableau qui précède, ne doit se trouver dans `/usr/local` après une installation d'un système conforme au SFN.

36.4.9.3 Options spéciales.

Si les répertoires `/lib<qual>` ou `/usr/lib<qual>` existent, les répertoires équivalents doivent aussi se trouver dans `/usr/local`.

36.4.10 /usr/sbin : binaires non-essentiels du système standard.

36.4.10.1 Objectifs.

Ce répertoire contient tout binaire non-essentiel au système, utilisé exclusivement par l'administrateur système. Les programmes d'administration qui sont requis pour la réparation, la récupération, le montage de `/usr`, ou d'autres fonctions essentielles doivent être placés dans `/sbin`.[27]

36.4.11 /usr/share : données indépendantes de l'architecture.

36.4.11.1 Objectifs.

L'arborescence de `/usr/share` accueille tous les fichiers de données indépendantes de l'architecture en lecture seule.[28]

Cette structure est prévue pour être partagée parmi toutes les architectures d'un système d'exploitation donné ; donc, par exemple, un site avec des plate-formes i386, Alpha et PPC peut très bien contenir un unique répertoire `/usr/share` monté de manière centralisée. Remarquez, cependant, que

(bien que nous recommandions que les distributions n'écrasent pas les données dans /etc lors des mises à jour). C'est pour cette raison que les logiciels installés localement ne doivent pas être installés hors de `/usr/local`, sauf motifs particuliers.

[27] Les programmes d'administration du système, installés localement, devraient être placés dans `/usr/local/sbin`.

[28] Beaucoup de ces données se trouvaient à l'origine dans `/usr` (`man`, `doc`) ou `/usr/lib` (`dict`, `terminfo`, `zoneinfo`).

`/usr/share` n'est généralement pas prévu pour être partagé entre des systèmes d'exploitation différents ni par des versions différentes du même système d'exploitation.

Tout programme ou paquet qui contient ou requiert des données qui ne doivent pas être modifiées devrait enregistrer ces données dans `/usr/share` (ou `/usr/local/share`, si ce programme ou ce paquet est installé localement). A cet égard, il est recommandé qu'un sous-répertoire soit utilisé dans `/usr/share`.

Les données des jeux enregistrées dans `/usr/share/games` doivent être purement statiques. Tous les fichiers modifiables, tels que les fichiers de résultats, les journaux des parties, etc., devraient être placés dans `/var/games`.

36.4.11.2 Conditions.

Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent se trouver dans `/usr/share` :

`/usr/share`	———	données indépendante de l'architecture
	`\_ man`	manuels en ligne
	`\_ misc`	diverses données indépendantes de l'architecture

36.4.11.3 Options spéciales.

Les répertoires suivants, ou les liens symboliques pointant vers ceux-ci, doivent se trouver dans `/usr/share`, si les sous-systèmes correspondant sont présents :

`/usr/share`	———	données indépendantes de l'architecture
	`\_ dict`	dictionnaires (optionnel)
	`\_ doc`	documentation diverse (optionnel)
	`\_ games`	fichiers de données statiques des jeux (optionnel)
	`\_ info`	répertoire primaire du système Info de GNU (optionnel)
	`\_ locale`	information locale (optionnel)
	`\_ nls`	catalogues de messages pour le support du langage Native (optionnel)
	`\_ sgml`	données SGML et XML (optionnel)
	`\_ terminfo`	répertoires de la base de données terminfo (optionnel)
	`\_ tmac`	macros de troff non distribuées avec groff (optionnel)
	`\_ zoneinfo`	information sur le fuseau horaire – configuration (optionnel)

Il est recommandé que les répertoires indépendants de l'architecture et spéficiques aux applications soient placés ici. De tels répertoires comprennent `groff`, `perl`, `ghostscript`, `texmf` et `kdb` (Linux) ou `syscons` (BSD). Cependant, ils peuvent être placés dans `/usr/lib` pour des raisons de rétro-compatibilité, et ce, à la discrétion de la distribution. De manière analogue, une arborescence

/usr/lib/games peut être utilisée en plus de l'arborescence de /usr/share/games si la distribution envisage d'y placer des données relatives aux jeux.

36.4.11.4 /usr/share/dict : dictionnaire (optionnel).

Objectifs. Ce répertoire accueille les dictionnaires du système. Traditionnellement, il contient seulement les fichiers words en anglais, tels qu'utilisés par look(1) ; divers programmes orthographiques exploitent l'orthographe anglaise ou américaine.

La raison pour laquelle seuls les dictionnaires sont confinés dans ce répertoire vient de ce qu'ils sont les seuls fichiers communs à tous les vérificateurs orthographiques.

Options spécifiques. Les fichiers suivants, ou les liens symboliques vers ces fichiers, doivent être présents dans /usr/share/dict, si le sous-système correspondant est installé :

words	Dictionnaire anglais (optionnel)

Les sites qui requièrent à la fois les orthographes anglaise et américaine simultanément peuvent lier words à /usr/share/dict/american-english ou /usr/share/dict/british-english.

Les dictionnaires associés aux autres langues peuvent être ajoutés en substituant le mot english comme ceci : /usr/share/dict/french ou /usr/share/dict/danish, etc. Si possible, ces dictionnaires devraient utiliser un jeu de caractères ISO-8859 approprié aux langues en question. Si possible (mais cela ne l'est pas toujours), le jeu de caractère Latin1 (ISO-8859-1) devrait être utilisé.

Les autres dictionnaires, s'il y a en a, doivent se retrouver dans ce répertoire-ci.

36.4.11.5 /usr/share/man : Pages du manuel.

Objectifs. Cette section présente en détail l'organisation des pages de manuel dans le système, y compris /usr/share/man. Référez-vous aussi à la section relative à /var/cache/man/.

Le répertoire primaire <mandir> du système est /usr/share/man. /usr/share/man contient l'information du manuel pour les commandes et les données sous les systèmes de fichiers / et /usr.[29]

Les pages de manuel sont stockées dans <mandir>/<locale>/man<section>-/<arch>. Une explicitation des termes <mandir>, <locale>, man<section> et <arch> est donnée ci-dessous.

Chaque section peut être décrite ainsi :

- man1 : programmes des utilisateurs

  Les pages du manuel qui décrivent les commandes accessibles aux utilisateurs sont contenues dans ce chapitre. La majorité de la documentation des programmes qu'un utilisateur exploitera se trouve là.

[29] Naturellement, il n'y a pas de pages de manuel dans / parce qu'elles ne sont pas nécessaires au moment de l'amorçage ni en cas d'urgence (on vous assure).

- `man2` : appels système
  Cette section décrit tout ce qui se rapporte aux appels systèmes (requêtes adressées au noyau pour effectuer des opérations).
- `man3` : fonctions de biblothèques et sous-routines.
  La section 3 décrit les routines de bibliothèques des programmes qui ne sont pas des appels directs aux services du noyau. Cette partie et la section 2 sont essentiellement intéressantes pour les programmeurs.
- `man4` : fichiers spéciaux
  La section 4 décrit les fichiers spéciaux, les fonctions de pilotes s'y rapportant, le support réseau disponible sur le système. Typiquement, ceci inclut les fichiers de périphériques trouvés dans `/dev` et l'interface noyau au support des protocoles du réseau.
- `man5` : formats de fichiers
  Les formats pour de nombreux fichiers de données sont documentés dans la section 5. Ceci comprend beaucoup de fichiers "include", des fichiers de sortie de programmes et des fichiers système.
- `man6` : jeux
  Cette section documente les jeux, les démonstrations et de manière générale, les programmes triviaux. Différentes personnes ont des appréciations diverses sur le caractère essentiel de ces programmes.
- `man7` : divers
  Les pages de manuel qui sont difficiles à classer dans les rubriques précédentes aboutissent ici. Les paquets associés aux macros de traitement de textes et à troff se retrouvent dans cette section.
- `man8` : administration système
  Les programmes utilisés par les administrateurs pour la gestion et l'entretien du système sont documentés dans cette section. Certains de ces programmes sont aussi occasionnellement utiles pour les utilisateurs normaux.

Options spécifiques. Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent se trouver dans `/usr/share/<mandir>/<locale>`, sauf s'ils sont vides :[30]

`/<mandir>/<locale>`	———	arborescence des pages de manuel
	`\_ man1`	programmes d'utilisateur (optionnel)
	`\_ man2`	appels système (optionnel)
	`\_ man3`	appels bibliothèque (optionnel)
	`\_ man4`	fichiers spéciaux (optionnel)
	`\_ man5`	formats de fichiers (optionnel)
	`\_ man6`	jeux (optionnel)
	`\_ man7`	divers (optionnel)
	`\_ man8`	administration système (optionnel)

La composante `<section>` décrit la section du manuel.

Des dispositions doivent être prises concernant la structure de `/usr/share/man` de manière à supporter les pages de manuel qui sont écrites dans différentes (ou de multiples) langues. Ces dispositions doivent prendre en considération le

[30] Par exemple, si `/usr/local/man` n'a pas de pages de manuel dans la section 4 (Périphériques), alors `/usr/local/man/man4` peut être omis.

stockage et les références à ces pages de manuel. Les facteurs pertinents comprennent la langue (y compris les différences de nature géographique) et le jeu de code de caractères.

La dénomination des sous-répertoires de langues de `/usr/share/man` est basée sur l'appendice E de la norme POSIX 1003.1 qui décrit les chaînes d'identification locales ; il s'agit de la méthode la plus largement acceptée pour décrire un environnement culturel linguistique. La chaîne `<string>` est :

```
<langue>[_<pays>] [.<jeu_de_caractère>] [,<version>]
```

Le champ `<langue>` doit être extrait de la norme ISO 639 (un code de représentation des noms de langues). Il doit être constitué de deux caractères et de lettres minuscules, exclusivement.

Le champ `<pays>` doit être constitué de deux lettres du code ISO 3166 (un code désignant les pays), si possible. (La plupart des personnes connaissent les codes à deux lettres désignant les pays dans les adresses de courriel).[31] Ce champ doit comprendre deux caractères et des lettres majuscules, exclusivement.

Le champ `<jeu_de_caractère>` doit être conforme à la norme décrivant le jeu de caractères. Si le champ `<jeu_de_caractère>` n'est constitué que de chiffres, le nombre qu'ils forment appartient au code international décrivant le jeu de caractères. Il est recommandé d'utiliser une représentation numérique autant que faire se peut (normes ISO, en particulier), sans inclure de symboles de ponctuation ni de lettres minuscules.

Un paramètre indiquant la `<version>` peut être placé après le champ `<jeu_de_caractère>`, le délimiteur étant une virgule. Ceci peut être employé pour faire une distinction linguistique ; par exemple, dans le cas de dictionnaires. Cette norme recommande de ne pas utiliser le champ `<version>`, sauf si cela est réellement nécessaire.

Les systèmes qui utilisent une seule langue et un seul code pour toutes les pages du manuel peuvent ne pas avoir de sous-chaîne `<locale>` et stocker toutes les pages de manuel dans `<mandir>`. Ainsi, les systèmes qui ont seulement des pages de manuel en anglais, codées en ASCII peuvent stocker ces pages (répertoires `man<section>`) directement dans `/usr/share/man`. En fait, il s'agit là du cas et de l'arragnement classiques.

Dans les pays pour lesquels existe un code de caractères standard accepté, les systèmes peuvent ne pas utiliser le champ `<jeu_de_caractère>`, mais il est fortement recommandé de l'inclure, notamment dans le cas des pays avec plusieurs standards.

Divers exemples sont donné le tableau des langues.

[31] Le Royaume-Uni constitue une exception importante à cette règle, puisqu'on trouve GB dans ISO 3166 et UK dans la plupart des adresses de messages électroniques.

Langue	Pays	Jeu de caractères	Répertoire
anglais	-	ASCII	/usr/share/man/en
anglais	Royaume Uni	ASCII	/usr/share/man/en_GB
anglais	Etats-Unis	ASCII	/usr/share/man/en_US
français	Canada	ISO 8859-1	/usr/share/man/fr_CA
français	France	ISO 8859-1	/usr/share/man/fr_FR
allemand	Allemagne	ISO 646	/usr/share/man/de_DE.646
allemand	Allemagne	ISO 6937	/usr/share/man/de_DE.6937
allemand	Allemagne	ISO 8859-1	/usr/share/man/de_DE.88591
allemand	Suisse	ISO 646	/usr/share/man/de_CH.646
japonais	Japon	JIS	/usr/share/man/ja_JP.jis
japonais	Japon	SJIS	/usr/share/man/ja_JP.sjis
japoanis	Japon	UJIS (ou EUC-J)	/usr/share/man/ja_JP.ujis

De manière analogue, des dispositions doivent être prises pour les pages de manuel dépendantes de l'architecture, telle que la documentation à propos des pilotes de périphériques ou les commandes d'administration système de bas niveau. Elles doivent être placées sous le répertoire <arch> dans le répertoire approprié de man<section> ; par exemple, une page de man pour la commande ctrlaltdel(8) sur i386 pourrait être placée dans /usr/share/man/<locale>/man8/i386/ctrlaltdel(8).

Les pages de manuel pour les commandes et données sous /usr/local sont stockées dans /usr/local/man. Les pages de manuel pour X11R6 sont conservées dans /usr/X11R6/man. Il s'en suit que tous les répertoires des pages du manuel doivent avoir la même structure que /usr/share/man.

Les sections des pages de catalogue (cat<section>) contenant des entrées de pages de manuel formatées sont aussi localisées dans les sous-répertoires de <mandir>/<locale>, mais ne sont pas nécessaires et peuvent ne pas être distribuées à la place des pages de man "nroff".

Les sections numérotées de 1 à 8 sont usuellement définies. En règle générale, le nom de fichier pour les pages de manuel localisées dans une section particulière se termine par .<section>.

Par ailleurs, des jeux de grande taille de pages du manuel présentent un suffixe additionnel au nom de fichier des pages. Par exemple, les pages de manuel relatives au système de gestion du courriel MH ont un suffixes mh (à toutes les pages). Toutes les pages du manuel relatives au système X Window doivent avoir un suffixe x.

La pratique consistant à placer des pages de manuel en diverses langues dans les sous-répertoires appropriés de /usr/share/man peut également être appliquée à d'autres arborescences telles que /usr/local.man et /usr/X11R6/man (cette partie de la norme s'applique également à la structure de la section /var/cache/man décrite ci-après).

36.4.11.6 /usr/share/misc : données diverses indépendantes de l'architecture.

Ce répertoire contient divers fichiers indépendants de l'architecture qui ne requièrent pas de sous-répertoire séparé sous /usr/share.

Options spécifiques. Les fichiers suivants, ou les liens symboliques vers ces fichiers, doivent se trouver dans `/usr/share/misc`, si le sous-système correspondant est installé :

`ascii`	table du jeu de caractères ASCII (optionnel)
`magic`	liste par défaut des nombres magiques pour les lignes de commandes (optionnel)
`termcap`	base de données des possibilités techniques des terminaux (optionnel)
`temcap.db`	base de données des possibilités techniques des terminaux (optionnel)

Les autres fichiers (spécifiques aux applications) peuvent apparaître ici,[32] mais une distribution, à sa discrétion, peut bien les placer dans `/usr/lib`.

36.4.11.7 /usr/share/sgml : données SGML et XML (optionnelles).

Objectifs. `/usr/share/sgml` contient les fichiers indépendant de l'architecture qui utilisent les applications SGML ou XML, tels que les catalogues ordinaires (non centralisés, voir `/etc/sgml`), les DTD,[33] ou les feuilles de style.

Options spécifiques. Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent être placés dans /usr/share/sgml, si le sous-système correespondant est installé :

`/usr/share/sgml`	———	données SGML et XML
	`\_ docbook`	docbook DTD (optionnel)
	`\_ tei`	tei DTD (optionnel)
	`\_ html`	html DTD (optionnel)
	`\_ mathml`	mathml DTD (optionnel)

Les autres fichiers qui ne sont pas spécifiques à une DTD donnée peuvent résider dans leur propre sous-répertoire.

[32] Par exemple :
`{airport, birthtoken, egnchar, getopt, gprof.callg, gprof.flat, inter.phone, ipfw.samp.filters, ipfw.samp.scripts, keycap.pcvt, mail.help, mail.tildehelp, man.template, map3270, mdoc.template, more.help, na.phone, nslookup.help, operator, scsi_modes, sedmail.hf, style, units.lib, vgrindefs, vgrindefs.db, zipcodes}`

[33] NdT : (1) DTD = Document Type Definition : description d'un type de document en SGML.
(2) SGML = Standard Generalized Markup Language. Il s'agit d'un métalangage utilisé pour définir de manière générale des langages associés à des documents hypertextes de tout type, normalisés sous ISO 8879. HTML en est un dérivé (très) simplifié, XML aussi.
(3) XML signifie eXtensible Markup Language. C'est une norme d'échange de documents informatisés issue de SGML, grâce au travail de l'ERB (Editorial Review Board), sous l'égide du W3C. En fait, il s'agit une version simplifiée de SGML. XML intègre l'idée de métadonnées, et permet de définir les balises souhaitées en fonction des besoins de l'utilisateur. Le but est de produire des documents vraiment structurés, grâce à un langage qu'on peut définir en fonction des circonstances, mais selon une syntaxe très stricte.

36.4.12 /usr/src : codes sources (optionnel).

36.4.12.1 Objectifs.

Tout code non-local devrait être placé dans ce sous-répertoire.

36.5 La hiérarchie de /var.

36.5.1 Objectifs.

`/var` contient des fichiers de données variables. Ceci inclut les répertoires et fichiers de "spoule", les données administratives et de connexion ainsi que des fichiers transitoires ou temporaires.

Certaines parties de `/var` ne sont pas partageables entre différents systèmes ; c'est, par exemple, le cas de `/var/log`, `/var/lock` et `/var/run`. D'autres parties sont partageables, notamment, `/var/mail`, `/var/cache/man`, `/var/cache/fonts` et `/var/spool/news`.

Ici, `/var` est analysé pour permettre de monter `/usr` en lecture seule. Tout ce qui est "entré" dans `/usr` et y a été écrit durant les opérations effectuées par le système (et qui ne concerne pas l'installation ou la maintenance) doit se trouver dans `/var`.

Si `/var` ne peut faire l'objet d'une partition séparée, il est préférable le plus souvent de faire transférer `/var` depuis la partition / vers `/usr`. (Ceci est parfois effectué en vue de réduire la taille de la partition racine ou lorsque l'espace de la partition racine est compté.) Cependant, `/var` ne doit pas être lié à `/usr` parce que cela rendrait la séparation entre `/usr` et `/var` très difficile à faire et que des conflits d'attribution de nom peuvent surgir. Il vaut mieux lier `/var` à `/usr/var`.

En règle générale, les applications ne doivent pas ajouter de répertoires au niveau supérieur de `/var`. De tels répertoires devraient seulement être ajoutés s'ils sont impliqués au niveau du système et après consultation de la liste de diffusion du SFN (*FHS-mailing list*).

36.5.2 Conditions.

Les répertoires suivants, ou les liens symboliques vers ces répertoires, sont nécessaires dans `/var` :

`/var`	———	données variables
	`\_ cache`	données de *cache* des applications
	`\_ lib`	information variable d'état
	`\_ local`	données variables pour `/usr/local`
	`\_ lock`	fichiers de verrouillage
	`\_ log`	fichiers et répertoires de journalisation
	`\_ opt`	données variables pour `/opt`
	`\_ run`	données pertinentes pour les processus en cours
	`\_ spool`	données de spoule des applications
	`\_ tmp`	fichiers temporaires préservés entre deux ré-amorçages du système

Plusieurs répertoires sont mis en réserve dans la mesure où ils ne doivent pas être utilisés arbitrairement, vu qu'il pourrait y avoir un conflit dû à des pratiques locales ou historiques. Il s'agit de :

```
/var/backups
/var/cron
/var/msgs
/var/preserve
```

36.5.3 Options spécifiques.

Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent se trouver dans `/var`, si le sous-système correspondant est installé :

`/var`	———	données variables
	`\_ account`	journaux de comptabilisation des processus (optionnel)
	`\_ crash`	nettoyage des plantages du système (optionnel)
	`\_ games`	données variables relatives aux jeux (optionnel)
	`\_ mail`	fichiers des boîtes de courrier électronique des utilisateurs (optionnel)
	`\_ yp`	fichiers de la base de données du Service d'Information Réseau (NIS ; *Network Information Services*) (optionnel)

36.5.4 /var/account : processus de comptes rendus des journaux (optionnel)

36.5.4.1 Objectifs.

Ce répertoire maintient un journal comptabilisant les processus en cours d'activité et les données d'utilisation des processus composés (tels qu'utilisés sur des systèmes de type Unix par `lastcomm` et `sa`).

36.5.5 /var/cache : données de *cache* des applications.

36.5.5.1 Objectifs.

`/var/cache` est prévu pour les données de *cache* provenant des applications. De telles données sont généralement enregistrées à la suite d'entrées/sorties effectuées sur de longues durées ou par des calculs. L'application qui est responsable de la création de ces données doit être capable de régénérer ou restaurer ces dernières. Contrairement à `/var/spool`, les fichiers en cache peuvent être supprimés sans perte de données. Les données doivent rester valides entre les invocations de l'application et le réamorçage du système.

Les fichiers localisés sous `/var/cache` peuvent être nettoyés de manière spécifique par une application, par l'administrateur système ou par les deux. L'application concernée doit toujours être capable de retrouver ces fichiers après une suppression manuelle (généralement , ceci se produit en raison d'un manque

d'espace disque). Aucune autre condition n'est formulée quant au format de données des répertoires de *cache*.

L'existence d'un répertoire séparé pour les données en *cache* permet aux administrateurs système de mettre en place une politique de sauvegarde sur différents disques depuis d'autres répertoires de `/var`.

36.5.5.2 Options spéciales.

`/var/cache`	———	répertoires de *cache*
	`\_ fonts`	polices produites localement (optionnel)
	`\_ man`	pages de manuel formatées localement (optionnel)
	`\_ www`	données de cache ou mandataire WWW (optionnel)
	`\_ <paquet>`	données de cache spécifiques à <paquet> (optionnel)

36.5.5.3 /var/cache/fonts : polices produites localement (optionnel).

Objectifs. Le répertoire `/var/cache/fonts` devrait être utilisé pour héberger toutes les polices créées dynamiquement. En particulier, toutes les polices qui sont produites automatiquement par `mktexpk` doivent être placées dans des sous-répertoires de `/var/cache/fonts`, nommés de manière appropriée.[34]

Options spécifiques. Les autres polices créées dynamiquement peuvent être placées dans cette arborescence, sous des répertoires de `/var/cache/fonts`, nommés de manière appropriée.

36.5.5.4 /var/cache/man : pages de manuel formatées localement (optionnel).

Objectifs. Ce répertoire offre un emplacement spécifique pertinent pour les machines qui fournissent une partition `/usr` en lecture seule, mais pour lesquelles on souhaite permettre la mise en stock de pages de man formatées localement. Les machines qui montent `/usr` en écriture (par exemple, lors d'installations en mode mono-utilisateur) peuvent ne pas utiliser `/var/cache/man` et écrire directement des pages de man formatées dans les répertoires `cat<section>` de `/usr/share/man`. Nous recommandons qu'au lieu de cela, ces machines utilisent au maximum les options suivantes :

- préformater toutes les pages de manuel à côté des versions non-formatées,
- permettre de ne pas stocker des pages de man formatées et requérir que soit effectué le formatage chaque fois que des pages sont installées,
- permettre de garder localement des pages de manuel formatées dans `/var/cache/man`.

[34] En fait, la présente norme ne concerne pas la "Structure de Répertoires de TEX" (un document qui décrit la disposition des fichiers et répertoires associés à TEX), mais il peut être utile de lire la "Structure de Répertoires de TEX" . Voir `ftp ://ctan.tug.org/tex/`.

La structure de /var/cache/man nécessite de rendre compte à la fois des multiples arborescences de pages de man et la possibilité d'un support de multiples langues.

Supposons une page de manuel non-formatée qui apparaît normalement dans <chemin>/man/<locale>/man<section> : le répertoire pour accueillir les pages non-formatées est /var/cache/man/<catpath>/<locale>/cat<section>, où le terme <catpath> est dérivé de <chemin> en ôtant toute composante précédant usr et/ou suivant share.[35] (Notez que la composante <locale> peut être manquante.)

Les pages de man écrites dans /var/cache/man peuvent éventuellement être supprimées ou transférées dans les répertoires préformatés appropriés de l'arborescence originelle de man ; de même, les pages de man formatées de l'arboresence de man peuvent être supprimées si on n'y accède pas après une durée déterminée.

Si des pages de man formatées proviennent d'un système se trouvant sur un support en lecture seule (tel qu'un CD-ROM, par exemple), elles doivent être installées dans l'arborescence originelle de man (par exemple, /usr/share/man/cat-<section>). /var/man/cache est réservé en tant que zone de stockage des pages de man formatées.

La version 1.2 de la norme spécifiait que /var/catman était l'arborescence adéquate. Le chemin a été modifié depuis pour devenir /var/cache de manière à mieux refléter la nature dynamique des pages de man formatées. Le nom du répertoire a été changé en man pour permettre l'étoffement de l'arborescence afin d'inclure des formats pré-traités, autres que "cat", tels que Postscript, HTML, ou DVI.

36.5.6 /var/crash : dépôts des plantages du système (optionnel).

36.5.6.1 Objectifs.

Ce répertoire est un dépôt des informations associées aux plantages de la machine. A la date de sortie de cette version de la norme, cette fonctionnalité n'était pas supportée par Linux.

36.5.7 /var/games : données de jeux (optionnel).

36.5.7.1 Objectifs.

Toute donnée variable associée aux jeux dans /usr devrait être placée ici. Le répertoire /var/games devrait contenir les données variables trouvées naguère dans /usr ; les données statiques, tels que les textes d'aide, les descriptions des niveaux, etc., doivent rester ailleurs, par exemple dans /usr/share/games.

La nouvelle norme a attribué à /var/games une arborescence qui lui est propre (alors que dans la version 1.2 de la norme, il y avait un répertoire /var/lib). La séparation ainsi introduite permet un meilleur contrôle local des stratégies de sauvegardes, de gestion des droits et de l'usage des disques. Elle permet aussi le partage entre hôtes et la réduction de l'encombrement de

[35] Par exemple, /usr/share/man/man1/ls.1 est formaté en /var/cache/man/cat1/ls.1, et /usr/X11R6/man/<locale>/man3/XtClass.3x en /var/cache/man/<locale>/cat3XtClass.3x.

/var/lib. En outre, /var/games se trouve dans le chemin utilisé traditionnellement par BSD.

36.5.8 /var/lib : information d'état.

36.5.8.1 Objectifs.

Cette arborescence contient les informations d'état d'une application ou du système. Les informations d'état sont des données que les programmes modifient durant leur exécution, et qui sont associées à une machine donnée. Il n'est jamais nécessaire que les utilisateurs modifient les fichiers dans /var/lib pour configurer un paquet.

Les informations d'état sont généralement exploitées pour préserver les conditions de travail d'une application (ou d'un groupe d'applications liées entre elles) entre différentes invocations ou entre différentes utilisations de la même application. Les informations d'état devraient en principe rester valides après le réamorçage, ne devraient pas être des informations de journalisation (*logging output*) ni être des données stockées.

Une application (ou un groupe d'applications liées entre elles) doit utiliser un sous-répertoire /var/lib pour ses données.[36] Il existe un sous-répertoire requis, /var/lib/misc, prévu pour les fichiers d'état n'utilisant pas de sous-répertoires propres. Les autres sous-répertoires ne devraient être présents que si l'application en question est incluse dans la distribution.

/var/lib/<nom> est l'emplacement qui doit être utilisée pour tous les supports des paquets des distributions. Bien sûr, les différentes distributions peuvent utiliser différents noms.

36.5.8.2 Conditions.

Les répertoires suivants, ou les liens symboliques vers ces répertoires, sont requis dans /var/lib :

/var/lib	———	informations d'état variables
	_ misc	diverses données d'état

36.5.8.3 Options spécifiques.

Les répertoires suivants, ou les liens symboliques vers ces répertoires, doivent se trouver dans /var/lib, si le sous-système correspondant est installé :

/var/lib	———	informations d'état variables
	_ <editeur>	fichers de sauvegardes et état de l'éditeur <editor> (optionnel)
	_ <pkgtool>	fichiers de support des paquets (optionnel)

[36] Une différence importante entre la présente version de la norme et les précédentes tient en ce que les applications doivent désormais utiliser un sous-répertoire /var/lib.

_ <paquet>	données d'état des paquets et des sous-systèmes (optionnel)
_ <kwclock>	répertoire d'état d'hwclock (optionnel)
_ xdm	données variables du gestionnaire d'affichage X (optionnel)

36.5.8.4 /var/lib/<editor> : état et fichiers de sauvegarde de l'éditeur (optionnel).

Objectifs. Ces répertoires contiennent une sauvegarde des fichiers produits lors d'un plantage inoppiné d'un éditeur (par exemple : elvis, jovi, nvi).

Les autres éditeurs peuvent ne pas nécessiter un répertoire pour la récupération de fichiers dans ces circonstances, mais un espace bien défini pour y enregistrer d'autres informations alors qu'ils sont en cours d'exécution. Ce type d'information devrait se retrouver dans un sous-répertoire tel que `/var/lib` (par exemple, GNU Emacs devrait placer les fichiers de verrouillage dans `/var/lib/emacs/lock`).

Les futurs éditeurs requerront éventuellement des informations d'états supplémentaires en plus des fichiers de récupération lors de plantages et des fichiers de verrouillage. Cette information devra alors être placée dans `/var/lib/<editeur>`.

Les versions précédentes de Linux, comme toutes les versions commerciales, utilisent `/var/preserve` pour vi et ses clones. Cependant, chaque éditeur utilise son propre format de fichiers de récupération, de sorte qu'un répertoire séparé est nécessaire pour chaque éditeur.

Les fichiers de verrouillage spécifiques aux éditeurs sont usuellement différents des fichiers de verrouillage des ressources ou des périphériques, stockés dans `/var/lock` et, par conséquent, ils sont logés dans `/var/lib`.

36.5.8.5 /var/lib/hwclock : répertoire d'état pour hwclock (optionnel).

Objectifs. Ce répertoire contient le fichier `/var/lib/hwclock/adjtime`.

Dans la version 1.2 du SFN, ce fichier était connu sous le nom de `/etc/adjtime`, mais comme `hwclock` le remet à jour, cet emplacement était incorrect.

36.5.8.6 /var/lib/misc : données diverses.

Objectifs. Ce répertoire contient les données variables qui ne sont pas placées dans `/var/lib/`. Une tentative devrait être engagée pour utiliser des noms uniques dans ce répertoire de manière à éviter des conflits relatifs à l'espace de noms.[37]

[37] Cette arborescence devrait contenir des fichiers stockés dans `/var/db` dans les versions de BSD, ce qui inclut `locate.database` et `mountdtab`, ainsi que la (ou les) base(s) de données des symboles du noyau.

36.5.9 /var/lock : fichiers de verrouillage.

36.5.9.1 Objectifs.

Les fichiers de verrouillage devraient être enregistrés dans la structure `/var/lock`.

Les fichiers de verrouillage des périphériques et des autres ressources partagées par de multiples applications, tels que les fichiers de verrouillage des périphériques série qui, originellement se trouvaient dans `/usr/spool/locks` ou `/usr/spool/uucp` doivent maintenant apparaître dans `/var/lock`. La convention d'attribution de nom à utiliser est `LCK..` suivie du nom de la base du fichier périphérique. Par exemple, pour verrouiller `/dev/ttyS0`, le fichier `LCK..ttyS0` doit être créé.[38]

Le format utilisé pour les contenus des fichiers de verrouillage doit être le format de verrouillage de fichiers HDB UUCP. Le format HDB permet d'enregistrer l'identificateur de processus (PID) sous forme d'un nombre décimal ASCII à 10 octets plus une mise à la ligne. Par exemple, si le processus 1230 fait appel à un fichier de verrouillage, celui-ci devrait contenir onze caractères : espace, espace, espace, espace, espace, espace, un, deux, trois, zéro, et une mise à la ligne (nouvelle ligne).

36.5.10 /var/log : fichiers et répertoires des journaux.

36.5.10.1 Objectifs.

Ce répertoire contient divers fichiers de journalisation. La plupart des journaux doivent être écrits dans ce répertoire ou dans un sous-répertoire dédié.

36.5.10.2 Options spéciales.

Les fichiers suivants, ou les liens symboliques vers ces fichiers, doivent être présents dans `/var/log`, si le sous-répertoire correspondant est installé :

`lastlog`	enregistrement de la dernière connexion de chaque utilisateur
`messages`	messages système provenant de `syslogd`
`wtmp`	enregistrement de toutes les connexions et déconnexions

36.5.11 /var/mail : fichiers de courriel d'utilisateurs.

36.5.11.1 Objectifs.

Le répertoire de réserve (*spool*) du courriel doit être accessible via `/var/mail` et la réserve des fichiers doit être structurée à l'aide de `<nom_utilisateur>`.[39]

Les fichiers de la boîte électronique d'un utilisateur, situés à cet endroit doivent être enregistrés sous le format standard UNIX des boîtes électroniques.

[38] Tout qui veut utiliser `/dev/ttyS0` a la possibilité de lire le fichier de verrouillage et d'agir en conséquence (tous les verrouillages dans `/var/lock` devraient être en lecture pour tous).

[39] Notez que `/var/mail` peut être un lien symbolique vers tout autre répertoire.

L'emplacement logique pour ce répertoire a été modifié de `/var/spool/mail` vers `/var/mail`, en vue d'harmoniser la plupart des mises en oeuvre UNIX. Ce changement est important pour l'interopérabilité vu qu'un unique répertoire `/var/mail` est souvent partagé entre de multiples hôtes et de multiples mises en oeuvre UNIX (en dépit de problèmes de verrouillage du système de fichiers NFS).

Il est important d'observer qu'il n'y a pas de contraintes pour déplacer physiquement la réserve de courrier dans cet endroit. Cependant, les programmes et les fichiers d'en-tête doivent être modifiés pour utiliser `/var/mail`.

36.5.12 /var/opt : données pour /opt.

36.5.12.1 Objectifs.

Les données variables des paquets logiciels dans `/opt` doivent être installées dans `/var/opt/<paquet>`. Le terme `<paquet>` est le nom de la sous-arborescence dans `/opt` où les données statiques provenant d'un paquet additionnel sont stockées. Aucune structure n'est imposée quant à l'arrangement interne de `/var/opt/<paquet>`.

Veuillez vous rapporter à `/opt`.

36.5.13 /var/run : données du temps de fonctionnement.

36.5.13.1 Objectifs.

Ce répertoire contient des données d'information système décrivant le système depuis son amorçage. Les fichiers se trouvant dans ce répertoire doivent être nettoyés (supprimés ou tronqués) au début du processus d'amorçage. Les programmes peuvent avoir un sous-répertoire de `/var/run`. Cette approche est encouragée dans le cas des programmes qui utilisent plus d'un fichier de durée d'exécution.[40] Les fichiers d'identificateur de processus (PID), qui sont à l'origine placés dans `/etc`, doivent être logés dans `/var/run`. La convention pour nommer les fichiers de PID est `<nom-programme>.pid`. Par exemple, le fichier PID de `crond` est nommé `/var/run/crond.pid`.

36.5.13.2 Conditions.

Le format interne des fichiers de PID reste inchangé. Les fichiers doivent consister en une représentation ASCII décimale, suivie d'un caractère de nouvelle ligne. Par exemple, si `crond` est le processus de PID 25, `/var/run/crond.pid` devrait contenir trois caractères : deux, cinq, et une nouvelle ligne.

Les programmes qui lisent les fichiers de PID devraient montrer un peu de souplesse : c'est-à-dire qu'ils devraient être capables d'ignorer les espaces excédentaires, les zéros placés au début, l'absence de mise à la ligne ou les lignes supplémentaires du fichier de PID. Les programmes qui créent les fichiers de PID devraient utiliser la spécification unique décrite dans le paragraphe ci-avant.

Le fichier `utmp`, qui enregistre de l'information à propos des utilisateurs employant le système, est localisé dans ce répertoire-ci.

[40] `/var/run` ne devrait pas être activé en écriture pour les utilisateurs non-privilégiés (root ou les utilisateurs exécutant des démons). Il y a un problème de sécurité majeur dès que tout utilisateur peut écrire dans ce répertoire.

Les programmes qui contiennent des *sockets* de domaines UNIX transitoires doivent les placer dans ce répertoire-ci.

36.5.14 /var/spool : données en attente.

36.5.14.1 Objectifs.

`/var/spool` contient les données en attente d'un traitement. Les données de `/var/spool` représentent des travaux devant être effectués ultérieurement par un programme, un utilisateur ou l'administrateur ; souvent, les données de ce type sont supprimées après leur traitement.[41]

36.5.14.2 Options spécifiques.

Les répertoires suivants, ou les liens symboliques qui pointent vers ces derniers, doivent être placés dans `/var/spool`, si le sous-système correspondant est installé :

`/var/spool`	———	répertoires de spoule
	`\_ lpd`	répertoire de spoule d'impression (optionnel)
	`\_ mqueue`	file d'attente du courriel sortant (optionnel)
	`\_ news`	répertoire de spoule des nouvelles (*news*) (optionnel)
	`\_ <rwho`	fichiers associés à rwhod (optionnel)
	`\_ uucp`	répertoire de spoule pour UUCP (optionnel)

36.5.14.3 /var/spool/lpd : files d'impression du démon d'imprimantes en ligne (optionnel).

Objectifs. Le fichier de verrouillage `lpd.clock` pour `lpd`, doit être placé dans `/var/spool/lpd`. Nous suggérons que le fichier de verrouillage de chaque imprimante soit placé dans le répertoire de spoule, imprimante par imprimante, et soit nommé `lock`.

Options spécifiques. Voici la structure du répertoire `/var/spool/lpd` :

`/var/spool/lpd`	———	informations d'état variables
	`\_ <imprimante>`	spoules de l'imprimante spécifique `<imprimante>` (optionnel)

36.5.14.4 /var/spool/rwho : fichiers rhwod (optionnel).

Objectifs. Ce répertoire contient les information relatives à `rhwod` pour les autres systèmes sur le réseau local.

Certaines versions de BSD utilisent `/var/rwho` pour ces données ; étant donné son emplacement historique dans `/var/spool` sur d'autres systèmes et sa

[41] Les fichiers de verrouillage UUCP doivent être placés dans `/var/lock`. Voir la section précédente sous `/var/lock`.

correspondance approximative à la définition des données mises en réserve (ou "spoulées"), cet emplacement est supposé plus approprié.

36.5.15 /var/tmp : fichiers temporaires préservés entre les redémarrages du système.

36.5.15.1 Objectifs.

Le répertoire `/var/tmp` est mis à disposition des programmes qui requièrent des fichiers et des répertoires temporaires devant être préservés entre deux réamorçage du système. En conséquence, il est plus pertinent d'enregistrer ces données dans `/var/tmp` que dans `/tmp`.

36.5.16 /var/yp : fichiers de base de données relatives au Service d'Information Réseau (NIS)

36.5.16.1 Objectifs.

Les données variables pour le Service d'Information Réseau (*Network Information Service* ou NIS), connue formellement comme les Pages Jaunes de Sun (*Sun Yellow Pages* ou YP), doivent être placées dans ce répertoire.

`/var/yp` est le répertoire de référence pour les données NIS (YP) et est presqu'exclusivement dévolu à la documentation et aux systèmes NIS.[42]

36.6 Annexes spécifiques au système d'exploitation.

36.6.1 Linux.

Cette section constitue une annexe propre au système Linux.

36.6.1.1 / : répertoire racine.

Sur les sytèmes Linux, si le noyau se trouve dans `/`, nous recommandons d'utiliser les noms `vmlinux` ou `vmlinuz`, qui sont régulièrement utilisés dans les paquets sources récents du noyau Linux.

36.6.1.2 /bin : binaires des commandes essentielles à l'utilisateur (pour tous les utilisateurs).

Les systèmes Linux qui requièrent ces commandes placent les fichiers additionnels dans `/bin`.

```
{ setserial }
```

[42] NIS ne devrait pas être confondu avec Sun NIS+, qui utilise un répertoire différent : `/var/nis`.

36.6.1.3 /dev : périphériques et fichiers spéciaux.

Tous les périphériques et fichiers spéciaux de `/dev` devraient respecter le document *Linux Allocated Devices*, disponible avec les sources du noyau Linux. Celui-ci est maintenu par H. Peter Anvin <hpa@zytor.com>.

Les liens symboliques dans `/dev` ne devraient pas être distribués avec le système Linux sauf s'ils respectent le document *Linux Allocated Devices*.

L'obligation de ne pas faire de liens symboliques "à la légère" est formulée parce que des montages locaux différent souvent de ce qui est fait sur les machines de développement des distributions. Ainsi, si un script d'installation d'une distribution configure les liens symboliques au moment de l'installation, ces derniers ne seront pas souvent remis à jour si des modifications sur le matériel sont réalisées sur une machine locale. Cependant, lorsqu'ils sont créés de manière responsable localement, ces liens peuvent être d'un bon usage.

36.6.1.4 /etc : configuration du système.

Les systèmes Linux qui requièrent des fichiers additionnels les placent dans `/etc`.

`{ lilo.conf }`

36.6.1.5 /proc : fichier virtuel d'information sur les processus et le réseau.

Le système de fichiers `proc` est *de facto* une méthode standard de Linux pour manipuler des processus et des information du système, plutôt que `/dev/kmem` ou les autres méthodes de ce type. Nous encourageons fortement l'usage de `proc` pour enregistrer et récupérer des informations relatives aux processus, à la mémoire et au noyau.

36.6.1.6 /sbin : binaires essentiels au système.

Les systèmes Linux placent ces fichiers supplémentaires dans `/sbin`.

- Les commandes du second système de fichiers étendu (ext2) (optionnel) :
  `{ badblocks, dumpe2fs, e2fsck, mke2fs, mklost+found, tune2fs }`
- l'installateur de la table du chargeur de démarrage (optionnel) :
  `{ lilo }`

36.6.1.7 Fichiers optionnels pour /sbin :

- Binaires statiques :
  `{ ldonfig, sln, ssync }`

`ln` statique (`sln`) et `sync` statique (`ssync`) sont très utiles lorsque les choses vont mal. Le premier usage de `sln` (pour réparer des liens symboliques mal construits dans `/lib` après une mise à jour mal organisée) n'est plus un problème majeur depuis que le programme `ldconfig` (usuellement placé dans `/usr/bin`) existe et qu'il peut agir comme un guide lors de la mise à jour des bibliothèques dynamiques. `sync` statique est d'une grande utilité dans certains cas d'urgence. Notez que ces commandes ne doivent pas forcément être liées statiquement aux commandes classiques `ln` et `sync` (quoique cela puisse arriver).

Le binaire `ldconfig` est optionnel dans `/sbin` vu qu'une machine peut exécuter `ldconfig` lors de son amorçage plutôt qu'exclusivement lors des mises à jour des bibliothèques partagées. (Il n'est cependant pas sûr qu'il soit ou non avantageux d'exécuter `ldconfig` à l'amorçage du système.) Quoiqu'il en soit, certains administrateurs préfèrent utiliser `ldconfig` dans les situations suivantes (très courantes) :

1. L'administrateur doit supprimer `lib/<fichier>`,
2. L'administrateur ne retrouve pas le nom de la bibliothèque parce que `ls` est lié dynamiquement ; il est en train d'utiliser un shell qui n'a pas la commande interne `ls` et et il ne sait pas utiliser "`echo *`" à la place,
3. L'adminsitrateur constate qu'il y a un lien statique `sln`, mais il ne sait pas comment faire appel au lien.

– Divers :
`{ ctrlaltdel, kbdrate }`

Pour faire face au fait que certains claviers sont caractérisés par une vitesse de répétition très haute qui les rend inutilisables, `kbdrate` peut être installé dans `/sbin` sur certains systèmes.

Etant donné que l'action, par défaut sur le noyau, de la combinaison Ctrl-Atl-Del engendre un réamorçage instantané et brutal, il est généralement recommandé de désactiver ce comportement avant que ne soit monté le système de fichiers racine en mode lecture-écriture. Certaines configurations d'`init` permettent de désactiver Ctrl-Alt-Del, mais d'autres exigent d'employer le programme `ctrlaltdel`, qui peut être installé dans `/sbin`.

36.6.1.8 /usr/include : fichiers d'en-tête inclus dans les programmes en C.

Les liens symboliques ci-après sont nécessaires si un compilateur C ou C++ est installé et ce, seulement pour les systèmes qui ne sont pas basés sur glibc :

```
/usr/include/asm -> /usr/src/linux/include/asm-<arch>
/usr/include/linux -> /usr/src/linux/include/linux
```

36.6.1.9 /usr/src : code source.

Pour les systèmes basés sur la glibc, il n'y a pas de règles spécifiques pour ce répertoire. Pour les systèmes basés sur les révisions libc Linux antérieures à la glibc, les règles suivantes s'appliquent :

Le seul code source qui devrait être placé dans un endroit spécial est le code source du noyau Linux (`/usr/src/linux`) ;

Si un compilateur C ou C++ est installé, mais que le code source du noyau Linux complet n'est pas installé, les fichiers "include" doivent se trouver dans les répertoires que voici :

```
/usr/src/linux/include/asm-<arch>
/usr/src/linux/include/linux
```

`<arch>` est le nom de l'architecture système.

Note : `/usr/src/linux` *peut être un lien symbolique pointant vers l'arborescence du noyau.*

Il est important que les fichiers "include" du noyau soient localisés dans `/usr/src/linux` et non dans `/usr/include` de manière à ne pas avoir de problèmes quand les administrateurs système mettent à jour leur version du noyau pour la première fois.

36.6.1.10 /var/spool/cron : cron et tâches.

Ce répertoire contient les données variables des programmes `cron` et `at`.

36.7 Annexe.

36.7.1 Liste de diffusion du SFN.

La liste de diffusion du Système de Fichiers Nomalisés (*Filesystem Hierarchy Standard* ou FHS) est accessible sur <fhs-discuss@uscd.edu>. Pour souscrire à la liste, envoyez un courriel à <listserv@ucsd.edu> avec pour message "ADD fhs-discuss".

Nous remercions Network Operations à l'Université de Californie – San Diego, qui nous a autorisés à utiliser leur excellent serveur de listes de diffusion.

Comme noté dans l'introduction, n'envoyez pas de courriel à la liste de diffusion avant d'avoir au préalable contacté l'éditeur du SFN ou un contributeur répertorié.

36.7.2 L'origine du SFN.

Le processus de développement d'une norme relative au système de fichiers a débuté en août 1993 pour restructurer l'arborescence des répertoires et fichiers de Linux (FSSTND). Une norme spécifique à Linux a été publiée pour la première fois en février 1994. Des révisions ont suivi le 9 octobre 2004 et le 28 mars 1995.

Au début 1995, fût adoptée l'idée de développer une version plus complète de la norme FSSTND pour traiter non seulement Linux mais aussi les autres systèmes de type Unix . Une aide a été apportée par les membres de la communauté de développement de BSD. En conséquence, un effort concerté a été entrepris pour se focaliser sur les problèmes généraux aux systèmes de type Unix. En reconnaissance de cet élargissement d'intérêt, le nom de la norme a changé pour devenir officiellement "Filesystem Hierarchy Standard" (ou FHS).

Les noms des volontaires qui ont contribué très largement à cette norme sont repris à la fin de la présente annexe. Cette norme est le fruit d'un consensus de ces contributeurs mais aussi d'autres personnes.

36.7.3 Lignes de conduite générales.

Voici quelques lignes de conduite qui ont été adoptées durant le développement de cette norme :

– résoudre des problèmes techniques en limitant les difficultés transitoires,
– rendre la norme raisonnablement stable,

- obtenir l'approbation des contributeurs, des développeurs, et des autres décideurs dans les groupes de développement pertinents et encourager leur participation,
- fournir une norme qui soit attractive pour les personnes mettant en oeuvre divers systèmes de type UNIX.

36.7.4 Perspectives.

Le présent document décrit une norme pour le système de fichiers en indiquant l'emplacement des fichiers et des répertoires et aussi, le contenu de certains fichiers.

La norme a été construite par des personnes élaborant des distributions, des développeurs de paquets logiciels et des administrateurs systèmes impliqués dans l'élaboration et la gestion de systèmes de fichiers conformes au SFN. Elle est initialement prévue pour constituer une référence et non pour être une introduction à la gestion d'un système de fichiers conforme.

Le SFN s'est développé sur base d'un travail antérieur appelé FSSTND, une organisation normalisée du système de fichiers basé sur le système d'exploitation Linux. Elle s'appuie sur le FSSTND pour traiter des problèmes d'interopérabilité non seulement pour la communauté Linux mais aussi pour une communauté plus large incluant les personnes utilisant les systèmes à base de BSD-4.4. Elle tire parti d'enseignements provenant du monde BSD et d'ailleurs, à propos des supports multi-architectures et des demandes associées à l'exploitation de réseaux hétérogènes.

Bien que cette norme soit plus étendue que les tentatives antérieures de normalisation des systèmes de fichiers, les mises-à-jour périodiques peuvent devenir nécessaires en fonction des exigences technologiques nouvelles. Il est également possible que des solutions meilleures aux problèmes traités présentement soient découvertes de sorte que les solutions avancées ici ne soient plus optimales. Cependant, la rétro-compatibilité sur un sujet spécifique est garantie d'une version à l'autre.

Les commentaires associés à cette norme sont bienvenus. Tout commentaire ou suggestion pour des modifications peut être adressé à l'éditeur original (Daniel Quinlan <quinlan@pathname.com>) ou à la liste de diffusion du SFN/FHS. Les commentaires relatifs à la grammaire et à l'orthographe devraient être adressés à l'éditeur du SFN/FHS.

Avant d'envoyer un courriel à la liste de diffusion, veuillez contacter en tout premier lieu l'éditeur du SFN/FHS de manière à éviter la rediscussion d'anciens sujets.

Les questions relatives à l'interprétation de certains points de ce document peuvent surgir. Si vous en ressentez la nécessité, contactez l'éditeur du SFN / FHS. Si vous ressentez la nécessité d'une quelconque clarification, contactez l'éditeur. Etant donné que la norme représente un consensus entre de nombreux participants, il est important de s'assurer que toute interprétation représente leur opinion collective. Pour cette raison, il se peut qu'il ne soit pas possible d'obtenir de réponse immédiate sauf si la demande de renseignement a déjà fait l'objet d'une discussion.

36.7.5 Remerciements.

Les développeurs du SFN/FHS souhaitent remercier les développeurs, les administrateurs système et les utilisateurs dont les contributions furent essentielles à l'élaboration de la norme. Nous souhaitons adresser nos remerciements à chaque contributeur qui a aidé à rédiger, compiler et composer cette norme.

Le Groupe de Normalisation souhaite remercier les développeurs de Linux qui ont aidé à l'élaboration du FSSTND, le prédécesseur de la présente norme. S'ils n'avaient pas montré que le FSSTND était essentiel, le SFN/FHS n'aurait jamais évolué.

36.7.6 Contributeurs.

Brandon S. Allbery	<bsa@kf8nh.wariat.org>
Keith Bostic	<bostic@cs.berkeley.edu>
Drew Eckhardt	<drew@colorado.edu>
Rik Faith	<faith@cs.unc.edu>
Stephen Harris	<sweh@spuddy.mew.co.uk>
Ian Jackson	<ijackson@cus.cam.co.uk>
John A. Martin	<jmartin@acm.org>
Ian McCloghrie	<ian@uscd.edu>
Chris Metclaf	<metclaf@lcs.mit.edu>
Ian Murdock	<imurdock@debian.org>
David C. Niemi	<niemidc@clark.net>
Daniel Quinlan	<quinlan@pathname.com>
Eric S. Raymond	<esr@thyrsus.com>
Rusty Russell	<rusty@rustcorp.com.au>
Mike Sangrey	<mike@sojurn.lns.pa.us>
David H. Silber	<dhs@glowworm.firefly.ac.uk>
Thomas Sippel-Dau	<t.sippel-dau@ic.ac.uk>
Theodore Ts'o	<tytso@athena.mit.edu>
Stephen Tweedie	<sct@dcs.ed.ac.uk>
Fred N. van Kempen	<waltje@infomagic.com>
Bernd Warken	<bwarken@mayn.de>

Chapitre 37

httpd : serveur web Apache.

Dans ce chapitre, nous montrons comment installer un serveur web gérant des domaines virtuels et des pages web dynamiques CGI.[1] Le langage HTML n'est pas couvert, et vous êtes supposé comprendre comment ce langage fonctionne, ou, du moins comment trouver de la documentation à son propos.

37.1 Bases à propos des serveurs web.

Dans la section 27.2, nous avons vu un exemple de session HTTP au cours de laquelle nous avons utilisé la commande `telnet`. En réalité, un *serveur web* n'est rien d'autre qu'un programme qui lit un fichier situé sur le disque dur chaque fois qu'un requête `GET /<nom_de_fichier>.html HTTP/1.0` arrive sur le port 80. Dans cette section, nous montrons un cas de serveur web écrit avec un script de shell [ce programme provient d'un auteur qui n'a pas signé le code source ; aussi, si vous vous reconnaissez, n'hésitez pas à envoyer un mél]. Vous devrez ajouter la ligne suivante :

```
www stream  tcp  nowait  nobody  /usr/local/sbin/sh-httpd
```

dans votre fichier `/etc/inetd.conf`. Si vous exécutez `xinetd`, vous devrez plutôt ajouter un fichier contenant :

```
service www
{
     socket_type       = stream
     wait              = no
     user              = nobody
     server            = /usr/local/sbin/sh-httpd
}
```

dans votre répertoire `/etc/xinetd.d/`. Ensuite, vous pourrez arrêter tous les serveurs web déjà en cours et redémarrer `inetd` (ou `xinetd`). Vous devrez éga-

[1] NdT : à l'heure où la traduction était entreprise, Apache version 2 pouvait déjà être installé. D'autre part, la flexibilité du langage dynamique PHP concurrence désormais très fortement le langage CGI.

lement créer un fichier de journalisation (/usr/local/var/log/sh-httpd.log) et au moins une page web (/usr/local/var/sh-www/index.html) que votre serveur acheminera. Elle pourrait contenir ceci, par exemple :

```
<HTML>
 <HEAD>
  <TITLE>Mon premier document</TITLE>
 </HEAD>
 <BODY bgcolor=#CCCCCC text=''#000000''>
Ceci est mon premier document<P>
Visitez-le.
  <A HREF=''http ://rute.sourceforge.net/''
   The Rute Home Page
  </A>
pour plus d'information.</P>
 </BODY>
</HTML>
```

Notez que le serveur fonctionne sous l'utilisateur nobody, si bien que le fichier de journalisation doit être en écriture pour cet utilisateur ; le fichier index.html doit être en lecture. Notez également l'utilisation de la commande getpeername, qui peut être modifiée en PEER='''' si vous n'avez pas installé le paquet netpipes. [Il n'est possible que les autres commandes utilisées soient disponibles sur d'autres systèmes Unix].

```
#!/bin/sh
VERSION=0.1
NAME=''ShellHTTPD''
DEFCONTENT=''text/html''
DOCROOT=/usr/local/var/sh-www
DEFINDEX=index.html
LOGFILE=/usr/local/var/log/sh-httpd.log

log() {
   local REMOTE_HOST=$1
   local REFERRER=$2
   local CODE=$3
   local SIZE=$4

   echo ``$REMOTE_HOST $REFERRER - [$REQ_DATE] \
\''${REQUEST}\'' ${CODE} ${SIZE}'' >> ${LOGFILE}
}

print_header() {
   echo -e ``HTTP/1.0 200 OK\r''
   echo -e ``Server : ${NAME}/${VERSION}\r''
   echo -e ``Date : `date`\r''
}
```

```
print_error() {
   echo -e "HTTP/1.0 $1 $2\r"
   echo -e "Content-type : $DEFCONTENT\r"
   echo -e "Connection : close\r"
   echo -e "Date : `date`\r"
   echo -e "\r"
   echo -e "$2\r"
   exit 1
}

guess_content_type() {
   local FILE=$1
   local CONTENT

   case ${FILE##*.} in
      html) CONTENT=$DEFCONTENT ;;
      gz) CONTENT=application/x-gzip ;;
      *) CONTENT=application/octet-stream ;;
   esac

   echo -e "Content-type : $CONTENT"
}

do_get() {
   local DIR
   local NURL
   local LEN

   if [ ! -d $DOCROOT ]; then
      log ${PEER} - 404 0
      print_error 404 "No such file or directory"
   fi

   if [ -z "${RL##*/}" ]; then
      URL=${URL}${DEFINDEX}
   fi

   DIR="`dirname $URL`"
   if [ ! -d ${DOCROOT}/${DIR} ]; then
      log ${PEER} - 404 0
      print_error 404 "Directory not found"
   else
      cd ${DOCROOT}/${DIR}
      NURL="`pwd`/`basename ${URL}`"
      URL=${NURL}
   fi

   if [ ! -f ${URL} ]; then
      log ${PEER} - 404 0
      print_error 404 "Document not found"
   fi
```

```
    print_header
    guess_content_type ${URL}
    LEN="`ls -l ${URL} | tr -s ' ' | cut -d ' ' -f 5`"
    echo -e "Content-length : $LEN\r\n\r"
    log ${PEER} - 200 ${LEN}
    cat ${URL}
    sleep 3
}

read_request() {
    local DIRT
    local COMMAND

    read REQUEST
    read DIRT

    REQ_DATE="`date +"%d/%b/%Y :%H :%M :%S %z"`"
    REQUEST="echo ${REQUEST} | tr -s [ :blank :]`"
    COMMAND="`echo ${REQUEST} | cut -d ' ' -f 1`"
    URL="`echo ${REQUEST} | cut -d' ' -f 2`"
    PROTOCOL="`echo ${REQUEST} | cut -d ' ' -f 3`"

    case $COMMAND in
       HEAD)
           print_error 501 "Not implemented (yet)"
          ;;
       GET)
           do_get
          ;;
       *)
           print_error 501 "Not implemented"
          ;;
       esac
}

#
# It was supposed to be clean -- without any non-standard utilities
# but I want some logging where the connexions come from, so
# I use just this one utility to get the peer address
#
# This is from the netpipes package
PEER="`getpeername | cut -d ' ' -f 1`"

read_request

exit0
```

A présent, exécutez `telnet localhost 80`, comme dans la section 27.2. Si cela fonctionne, et que vos fichiers de journalisation se remplissent correctement (`tail -f ...`), vous pouvez essayer de vous connecter à http ://localhost

avec un navigateur web comme Firefox, Mozilla, Galeon, Netscape.

Notez encore que la commande `getsockname` (qui vous indique les adresses IP [parmi les vôtres] auxquelles un client distant se connecte) pourrait permettre au script de servir des pages web à partir d'un répertoire différent pour chaque adresse IP. Voici en bref en quoi consiste les *domaines virtuels* [...].

37.2 Installation et configuration d'Apache.

Du fait que chaque distribution empaquette Apache à sa manière, nous supposerons ici qu'Apache a été installé à partir des sources, plutôt quà partir des paquets `.deb` ou `.rpm`. Vous pouvez recourir à la section 25.1 pour déterminer comment installer Apache depuis ses sources sous forme d'un paquet `tar.gz`, ce dernier étant accessible comme paquet GNU. Vous pouvez même l'installer sous MS-Windows, Windows NT, ou OS/2. Le paquet source est, bien entendu, accessible depuis *Apache Home Page* à l'adresse http ://www.apache.org. Nous supposerons aussi que l'installation a été faite avec `--prefix=/opt/apache`. Au cours de l'installation, Apache aura déversé une grande quantité de documentation dans `/opt/apache/htdocs/manual`.

37.2.1 `httpd.conf`.

Apache possède plusieurs fichiers de configuration hérités des premiers développements : `access.conf` et `srm.conf` en sont deux exemples. Ces fichiers sont désormais obsolètes et devraient rester vides. Un fichier de configuration élémentaire `/opt/apache/conf/httpd.conf` devrait contenir au minimum :

```
ServerType standalone
ServerRoot "/opt/apache"
PidFile /opt/apache/logs/httpd.pid
ScoreBoardFile /opt/apache/logs/httpd.scoreboard
Port 80
User nobody
Group nobody
HostnameLookups Off
ServerAdmin webmaster@cranzgot.co.za
UseCanonicalName On
ServerSignature On
DefaultType text/plain
ErrorLog /opt/apache/logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /opt/apache/logs/access_log common
DocumentRoot "/opt/apache/htdocs"
DirectoryIndex index.html
AccessFileName .htaccess
<Directory />
   Options FollowSymLinks
   AllowOverride None
   Order Deny,Allow
```

```
   Deny from All
</Directory>
<Files ~ "^\.ht">
   Order allow,deny
   Deny from all
</Files>
<Directory "/opt/apache/htdocs">
   Options Indexes FollowSymLinks MultiViews
   AllowOverride All
   Order allow,deny
   Allow from all
</Directory>
<Directory "/opt/apache/htdocs/home/*/www">
   Options Indexes MultiViews
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>
UserDir /opt/apache/htdocs/home/*/www
```

Une fois le fichier de configuration prêt, vous pouvez déplacer le fichier `index.html` dans `/opt/apache/htdocs/`. Vous verrez qu'il y a un manuel complet d'Apache et une page de démonstration déjà installée. A présent, vous pouvez les déplacer dans un autre répertoire et exécutez :

```
/opt/apache/bin/httpd -X
```

37.2.2 Directives communes.

Les options sont décrites ci-dessous. Chaque option est appelée "directive" dans la terminologie d'Apache. La liste complète des directives fondamentales se trouve dans `/opt/apache/htdocs/manual/mod/core.html`.

ServerType Comme discuté dans la section 30.2, certains services peuvent être exécutés en mode autonome (ou *standalone*), via `inetd` ou via `xinetd`. La présente directive peut être `standalone` ou `inetd`. Si vous choisissez `inetd`, vous devrez ajouter une ligne idoine dans le fichier de configuration d'`inetd`, bien qu'un serveur web fonctionnee presque toujours en mode autonome.

ServerRoot Cette option représente la superstructure du répertoire [voir à la page 160] sous lequel Apache est installé. Il s'agira toujours de la même valeur que celle passée à `--prefix=`.

PidFile De nombreux services du système enregistrent l'ID de processus dans un fichier à des fins de contrôle ou d'arrêt. Sur la plupart des distributions, le fichier s'appelle `/var/run/httpd.pid`.

ScoreBoardFile Cette option est utilisée pour des communications entre le processus-parent et les processus-enfants d'Apache sur certains systèmes non-UNIX.

Port Ceci est le port TCP d'écoute des serveurs fonctionnant en mode autonome (*standalone*).

User, Group Cette option est importante en terme de sécurité. Elle force `httpd` à utiliser les privilèges de l'utilisateur `nobody`. Si le serveur web venait à être "craqué", l'attaque ne pourrait permettre d'accèder à d'autres privilèges que ceux de `nobody`.

HostNameLookups Pour forcer une requête inverse de DNS sur chaque hôte se connectant, activez cette directive avec `on`. Pour forcer une requête directe sur chaque requête inverse, activez cette directive avec `double`. Cette option concerne les connexions vu que le contrôle de l'accès effectue une requête inverse et directe si cela est requis. Si vous voulez réduire le temps de latence, mettez `off`.

ServerAdmin Les messages d'erreur incluront cette adresse de courriel.

UseCanonicalName Si Apache doit retourner une URL pour quelque raison que ce soit, il s'agira normalement du nom complet du serveur. Le fait de mettre `off` permet d'utiliser le propre nom d'hôte envoyé par le client.

ServerSignature Tous les fichiers retournés au client ont un champ spécifiant la manière dont le fichier devrait être affiché. Si Apache ne peut en déduire le type, il supposera que le type MIME est `text/plain`. Voir la sous-section 13.6.2 pour une discussion sur les types MIME.

ErrorLog Cette directive indique où les erreurs doivent être consignées ; usuellement, il s'agit de `/var/log/httpd/error_log`.

Loglevel définit la quantité d'information à journaliser.

Logformat définit un nouveau format de journal. En l'occurrence, nous définissons et appelons le format de journal `common`. Les lignes multiples sont autorisées. De nombreuses informations peuvent être consignées : voir `/opt/apache/htdocs/manual/mod/mod_log_config.html` pour une description complète.

CustomLog Le nom du fichier de journalisation et son format (tel que définit dans la directive précédente).

DocumentRoot Cette directive spécifie le répertoire de plus haut niveau auquel un client peut accéder. La chaîne `/opt/apache/htdocs` est d'office la cible des requêtes, et donc, une URL comme http ://localhost/manual/index.html.en retournera le fichier `/opt/apache/htdocs/manual/index.html.en`.

DirectoryIndex Cette directive donne le fichier par défaut pour essayer de répondre à des URLs qui contiennent seulement un nom de répertoire. Si le fichier index.html n'existe pas sous ce répertoire, un index des répertoires est transmis au client. D'autres configurations usuelles employent `index.htm` ou `default.html`.

AccessFileName Avant de fournir un fichier à un client, Apache lit des directives supplémentaires à partir du fichier `.htaccess` se trouvant dans le même répertoire que le fichier demandé. Si le répertoire parent contient un fichier `.htaccess`, ce dernier a la priorité. Le fichier `.htaccess` contient des directives limitant l'accès au répertoire (voir ci-dessous).

Ce qui précède concerne seulement la configuration générale d'Apache. Pour servir réellement des pages web, il est nécessaire de définir des répertoires, chacun avec un thème particulier et contenant des fichiers graphiques ou HTML. Le fichier de configuration d'Apache est très semblable à celui d'un document HTML. Les sections débutent par <*paramètre-de-section*> et se termine par </*section*>.

La directive la plus commune de cette sorte est `<Directory /répertoire>`. Avant de définir quelque répertoire que ce soit, vous devez limiter l'accès au répertoire racine. Ce contrôle est très critique pour la sécurité du système.

```
<Directory />
   Options FollowSymLinks
   Deny from All
   Order Deny,Allow
   AllowOverride None
</Directory>
```

Cette configuration déclare le répertoire racine à Apache, de manière à restreindre l'accès aux clients de façon très stricte. Ces directives sont [certaines proviennent directement du manuel d'Apache] :

Options La directive `Options` contrôle quelles sont les caractéristiques du serveur disponibles dans un répertoire particulier. Il existe aussi une syntaxe *+option* ou *-option* permettant d'inclure les options du répertoire parent, par exemple, `Options +FollowSymLinks -Indexes`.

FollowSymlinks Le serveur suivra tous les liens symboliques sous le répertoire. Soyez prudent avec `FollowSymLinks`. Vous pouvez –par exemple– donner accès à n'importe qui au répertoire racine, en ayant un lien `../../../` sous htdocs, ce qui n'est pas ce que vous cherchez.

ExecCGI L'exécution de scripts CGI est permise.

Includes Les commandes exécutées par le serveur (*server-side includes*) peuvent être lancées (plus d'information est donnée ci-après).

IncludesNOEXEC Les commandes exécutées par le serveur (*server-side includes*) sont activées, mais les commandes `# exec` et `# include` des scripts CGI sont désactivées.

Indexes Si un client demande nommément un répertoire et qu'aucun fichier `index.html` n'est présent (ou que vous n'avez spécifié aucun fichier `DirectoryIndex`), une liste du contenu de ce répertoire est créée et fournie. Pour des raisons de sécurité, il se peut que vous souhaitiez que cette option soit désactivée.

MultiViews Les vues multiples sont autorisées (plus d'information est donnée ci-après).

SymLinksOwnerMatch Le serveur suit seulement les liens symboliques dont le fichier ou le répertoire cible est la propriété du même ID d'utilisateur que celui du lien (plus d'information est donnée ci-après).

All Toutes les options sont activées sauf `MultiViews`. C'est le réglage par défaut.

Deny Les hôtes qui ne sont pas autorisés à se connecter. Vous pouvez indiquer un nom d'hôte particulier ou une adresse IP, par exemple, comme ceci :

```
Deny from 10.1.2.3
Deny from 192.168.5.0/24
Deny from cranzgot.co.za
```

Ceci aura pour effet de refuser l'accès à `10.1.2.3`, à toutes les machines dont les IP commencent par `192.168.5.` et à toutes les machines dont l'adresse se termine par `cranzgot.co.za`, ce qui inclut aussi l'hôte `cranzgot.co.za`.

Allow Les hôtes autorisés à se connecter. Cette directive utilise la même syntaxe que `Deny`.

Order si `Order` a pour valeur `Deny,Allow`, les directives `Deny` sont d'abord consultées. Tout client qui n'est pas répertorié sous `Deny` et correspond à la directive `Allow` est *autorisé* à accéder au serveur.
Si `Order` a pour valeur `Allow,Deny`, les directives `Allow` sont d'abord consultées et tout client qui n'y correspond pas et correspond à une directive `Deny` verra son accès au serveur *refusé*.

AllowOverride En plus des directives indiquées ici, d'autres seront lues depuis le fichier spécifié par `AccessFileName`, usuellement appelé `.htaccess`. Habituellement, ce fichier est présent parmi vos fichiers `.html` ou, à défaut, dans le répertoire parent. Si ce fichier existe, son contenu est lu dans la directive `<Directory ...>`. L'option `AllowOverride` indique quelles directives le fichier `.htaccess` peut écraser. La liste complète peut être trouvée dans `/opt/apache/htdocs/manual/mod/core/html`.

Vous pouvez remarquer que nous donnons des `Options` très restrictives sur le répertoire racine ainsi que des accès très restreints. La seule caractéristique du serveur que nous autorisons est `FollowSymLinks`, nous refusons (`Deny`) tout accès et nous ôtons au fichier `.htaccess` la possibilité de passer outre nos restrictions.

La directive `<Files ...>` pose des restrictions sur tous les fichiers correspondants à une expression rationnelle particulière. En terme de mesures de sécurité, nous l'utilisons pour empêcher l'accès à tous les fichiers `.htaccess` de la manière suivante :

```
<Files ~ "^\.ht">
   Order allow,deny
   Deny from all
</Files>
```

Nous sommes finalement prêt à ajouter nos répertoires contenant les vraies pages web. Sur ce sujet, il y a moins de restrictions quant à l'accès :

```
<Directory "/opt/apache/htdocs">
   Options Indexes FollowSymLinks MultiViews
   AllowOverride All
   Order allow,deny
   Allow from all
</Directory>
```

37.2.3 Répertoires HTML d'utilisateurs.

Vos utilisateurs voudront sans doute qu'Apache ait accès à leurs pages web privées `~/www/`. Cette requête est facile à satisfaire à l'aide de la directive spéciale `UserDir` :

```
<Directory "/opt/apache/htdocs/home/*/www"
   Options Indexes MultiViews
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>
UserDir /opt/apache/htdocs/home/*/www
```

Pour que cette fonctionnalité soit activée, vous devez faire pointer un lien symbolique `/opt/apache/htdocs/home` vers `/home`, et créer un répertoire `www/` dans chacun des répertoire personnel (*home*) de chaque utilisateur. Le fait d'entrer l'URL `http ://localhost/~jacques/index.html` permet de retrouver le fichier `/opt/apache/htdocs/home/jacques/www/index.html`. Si Apache émet un message d'erreur intitulé `Forbidden` lorsque vous essayez cela, c'est probablement parce que les droits sur le répertoire *home* appelé `jacques` sont trop restrictifs. A présent, vous devez choisir une politique entre deux extrêmes : rendre l'accès au répertoire `jacques` moins restrictif ou élargir les droits sur Apache. Exécutez Apache sous le groupe `www` en utilisant `Group www`, et exécutez :

```
groupadd -g 65 www
chown jacques :www /home/jacques /home/jacques/www
chmod 0750 /home/jacques /home/jacques/www
```

Il s'agit là d'un compromis faisant preuve d'une bonne responsabilité en terme d'administration.

37.2.4 Création de synonymes.

Parfois, les documents HTML font référence à un fichier, à un graphique ou à un dessin en utilisant un simple préfixe, plutôt que le nom étendu du répertoire. En d'autres occasions, nous souhaiterons disposer de deux références distinctes dont la source est un même fichier. La directive `Alias` crée des liens virtuels entre les répertoires. Par exemple, le fait d'ajouter la ligne qui suit aura pour effet qu'une URL `/icons/bomb.gif` servira le fichier `/opt/apache/icons/bomb.gif` :

```
Alias /icons/ "/opt/apache/icons/"
```

Naturellement, nous devons informer Apache de cela :

```
<Directory "/opt/apache/icons"
   Options None
   AllowOverride None

   Order allow,deny
   Allow from all
</Directory>
```

37.2.5 Icônes de fantaisie.

Les listes de répertoires engendrées par la configuration que nous venons d'établir sont plutôt d'aspect strict. La directive :

```
IndexOptions FancyIndexing
```

permet d'afficher de jolies icônes descriptives à gauche du nom des fichiers. Quelles icônes pour quels fichiers ? Voici une question qui n'est pas simple à résoudre. Vous pouvez démarrer avec ceci :

```
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/images2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
AddIcon /icons/compressed.gif .Z .z .tag .gz .zip
AddIcon /icons/a.gif .ps .eps
AddIcon /icons/layout.gif .html .shtml .htm
```

Ceci nécessite aussi que la directive `Alias` décrite ci-dessus soit présente. La configuration par défaut d'Apache contient une liste bien plus grande de types de fichiers que ce qui est décrit ici.

37.2.6 Encodage et négociations relatives à la langue.

Apache peut servir des fichiers compressés par l'utilitaire `gzip` grâce à :

```
AddEncoding x-compress Z
AddEncoding x-gzip gz
```

A présent, si un client demande un fichier `index.html`, mais que seul `index.html.gz` existe, Apache décompresse le fichier à la volée. Notez bien que l'option `MultiViews` doit avoir été activée.

Les options suivantes permettent à Apache de servir des pages `index.html.`*code-de-langues* lorsqu'`index.html` est sollicité, en remplisant les pages avec la langue utilisée par le navigateur web. L'ajout de ces directives fait que le manuel

d'Apache et les pages qui ne sont pas en anglais s'affichent correctement dans la langue du navigateur. Dans ce cas aussi, l'option `MultiViews` doit être activée.

```
AddLanguage en .en
AddLanguage da .dk
AddLanguage nl .nl
AddLanguage et .ee
AddLanguage fr .fr

AddLanguage de .de
AddLanguage el .el
AddLanguage ja .ja
AddLanguage ru .ru
LanguagePriority en da nl et fr de el ja ru
```

La directive `LanguagePriority` indique le langage préféré si le navigateur n'en indique aucune.

Certains fichiers pourraient contenir une extension `.koi8-r`, indiquant qu'un jeu de caractères russes encode ce fichier. Beaucoup de langues ont un jeu de caractères personnalisés. Les fichiers russes ont une extension `.html.ru.koi8-r`. Apache doit indiquer au navigateur web le type d'encodage (c'est à cela que sert l'extension). Voici les directives pour le japonais, le russe et UTF-8 [UTF-8 est un jeu unicode utile dans chaque langue] :

```
AddCharset ISO-2022-JP .jis
AddCharset KOI8-R      .koi8-r
AddCharset UTF-8       .utf8
```

A nouveau, la configuration par défaut d'Apache contient bien davantage de langues et de jeu de caractères.

37.2.7 Portions de code incluses – SSI.

Apache possède un langage interne qui interprète les fichiers `.shtml` comme des scripts. Le résultat de tels scripts est retourné au client. La plupart des fichiers `.shtml` ne contiennent que du HTML et sont livrés sans modification. Cependant, des lignes comme celle-ci :

```
<!--#echo var="DATE_LOCAL" -->
```

seront interprétées, et leur résultat sera inclus dans l'HTML –d'où le nom de *server-side includes* (portions de code incluses) pour désigner des commandes de scripts exécutées en interne par le serveur. Ces *SSI* sont idéaux pour les pages HTML qui contiennent principalement de l'HTML statique avec peu de contenu dynamique. Pour se rendre compte de cela, ajoutez ce qui suit à votre fichier `http.conf` :

```
AddType text/html .shtml
AddHandler server-parsed .shtml
<Directory "/opt/apache/htdocs/ssi"
   Options Includes
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>
```

Créez un répertoire `/opt/apache/htdocs/ssi` avec un fichier `index.shtml` :

```
<HTML>
The date today is <!--#echo var="DATE_LOCAL" -->.<P>
Here is a directory listing :<br>
 <PRE>
  <!--#exec cmd="ls -al" -->
 </PRE>
<!--#include virtual="footer.html" -->
</HTML>
```

et un fichier `footer.html` contenant ce que vous voulez. L'utilité de cette méthode ressort lorsqu'il s'agit de créer de nombreux documents avec la même bannière, au moyen de la déclaration `# include`. Si vous vous demandez quelles autres variables vous pouvez afficher à côté de `DATE_LOCAL`, essayez donc :

```
<HTML>
 <PRE>
   <!--#printenv-->
 </PRE>
</HTML>
```

Visitez l'adresse http ://localhost/manual/howto/ssi.html pour consulter d'autres exemples.

37.2.8 CGI - Interface passerelle commune.

CGI –*Common Gateway Interface*– permet à une URL de pointer vers un script. Cependant, les données envoyées sur votre navigateur sont le résultat du script et non le contenu du script lui-même. Pour essayer cela, créez un fichier nommé `/opt/apache/htdocs/test.cgi` :

```
#!/bin/bash

echo 'Content-type : text/html'
echo
echo '<HTML>'
echo ' <HEAD>'
echo '  <TITLE>Mon premier CGI</TITLE>'
echo ' </HEAD>'
echo ' <BODY bgcolor=#CCCCCCC text="#000000">'
echo 'Ceci est mon premier CGI<P>'
echo 'Visitez s'il vous plaît'
echo '   <A HREF="http ://rute.sourceforge.net/">'
echo '    The Rute Home Page'
echo '   </A>'
echo 'pour plus d'information.</P>'
echo ' </BODY>'
echo '</HTML>'
```

Rendez ce script exécutable avec `chmod a+g test.cgi` et testez son résultat en l'exécutant en ligne de commandes. Ajoutez la ligne :

```
AddHandler cgi-script .cgi
```

à votre fichier `httpd.conf`. Ensuite, modifiez `Options` dans le répertoire `/opt/apache/htdocs` de manière à inclure `ExecCGI`, comme ceci :

```
<Directory "/opt/apache/htdocs">
   Options Indexes FollowSymLinks MultiViews ExecCGI
   AllowOverride All
   Order allow,deny
   Allow from all
</Directory>
```

Après avoir redémarrer Apache, vous devriez pouvoir visiter l'URL http ://localhost/test.cgi. Si des problèmes apparaissent, n'oubliez pas de lancer `tail /opt/apache/logs/error_log` de manière à obtenir un rapport complet.

Pour obtenir une liste exhaustive des variables d'environnement disponibles pour vos programmes CGI, essayez le script suivant :

```
#!/bin/sh

echo 'Content-type : text/html'
echo
echo '<HTML>'
echo '<PRE>'
set
echo '</PRE>'
echo '</HTML>'
```

Ce script révèlera les variables ordinaires de bash mais, il y a moyen d'accèder aussi à des variables plus intéressantes comme QUERY_STRING, par exemple. Modifiez un peu votre script :

```
#!/bin/sh

echo 'Content-type : text/html'
echo
ecch '<HTML>'
echo '<PRE>'
echo $QUERY_STRING
echo '</PRE>'
echo '</HTML>'
```

et rendez vous à l'URL http ://localhost/test/test.cgi?xxx=2&yyy=3. Il est aisé de voir comment les variables peuvent être passées à un script.

L'exemple précédent n'est toutefois pas très attractif. Cependant, il s'avère utile quand les scripts présentent une logique complexe ou quand ils peuvent donner accès à des informations auxquelles Apache ne peut accéder de lui-même. Dans le chapitre 39, nous verrons comment déployer une base de données SQL. Lorsque vous aurez étudié SQL, vous pourrez revenir ici et remplacer votre script CGI par :

```
#!/bin/bash

echo 'Content-type : text/html'
echo

pqsl -d template1 -H -c "SELECT * FROM pg_tables;"
```

Ce script videra la liste des tables de la base de données template1 si cette dernière existe. Apache devra fonctionner en tant qu'utilisateur pouvant accèder à la base de données, ce qui signifie qu'il faudra changer User nobody en User postgres [notez que pour des raisons de sécurité, nous devriez vraiment limiter les utilisateurs pouvant se connecter à la base de données postgres. Voir la section 39.4].

37.2.9 Formulaires et CGI.

Pour créer un formulaire fonctionnel, utilisez le marqueur <FORM> d'HTTP de la manière suivante. Un fichier /opt/apache/htdocs/form.html :

```
<HTML>
 <FORM name="myform" action="test.cgi" method="get">
   <TABLE>
     <TR>
       <TD colspan="2" align="center">
         Please enter your personnal details :
       </TD>
     </TR>
     <TR>
       <TD>Name :</TD><TD><INPUT type="text" name="name"></TD>
     </TR>
     <TR>
       <TD>Email :</TD><TD><INPUT type="text" name="email"></TD>
     </TR>
     <TR>
       <TD>Tel :</TD><TD><INPUT type="text" name="tel"></TD>
     </TR>
     <TR>
       <TD colspan="2" align="center">
         <INPUT type="submit" value="Submit"
       </TD>
     </TR>
   </TABLE>
 </FORM>
</HTML>
```

devrait donner :

Notez la manière dont ce formulaire appelle notre script existant `test.cgi`. Voici un script qui ajoute les données entrées, dans notre la table `postgres` SQL :

```
#!/bin/sh

echo 'Content-type : text/html'
echo

opts=`echo "$QUERY_STRING"` | \
   sed -e 's/[^A-Za-a0-9 %&+,.\/ :=@_~-]//g' -e 's/&/ /g'    -e q`

for opt in $opts ; do
   case $opt in
       name=*)
           name=${opt/name=/}
           ;;
       email=*)
           email=${opt/email=/}
           ;;
       tel=*)
           tel=${opt/tel=/}
           ;;
   esac
done

if psql -d template1 -H -c "\
INSERT INTO people (name, email, tel) \
VALUES ('$name', '$email', '$tel')" 2>&1 | grep -q '^INSERT ' ; then
   echo "<HTML>Your details \$name\", \"$email\" and \"$tel\"<BR>"
   echo "have been successfully recorded.</HTML>
else
   echo "<HTML>Database error, please contact our webmaster.</HTML>"
fi

exit 0
```

Remarquez comment les premières lignes du script éliminent tous les caractères non-désirés de `QUERY_STRING`. Un tel traitement est obligatoire pour des raisons de sécurité parce que les scripts de shell peuvent aisément exécuter des commandes si des caractères comme `$` et ` sont présent dans une chaîne.

Pour utiliser la méthode "POST" alternative, modifiez le marqueur `FORM` en :

```
<FORM name="myform" action="test.cgi" method="post">
```

La méthode POST envoie le texte de la requête via l'entrée standard (stdin) du script CGI. Par conséquent, vous devez aussi modifier votre ligne `opts=` en :

```
opts=`cat | \
  sed -e 's/[^A-Za-z0-9 %&+,.\/ :=@_~-]//g' -e 's/&/ /g' -e q`
```

37.2.10 setuid des scripts CGI.

Le fait d'éxécuter Apache en tant qu'utilisateur privilégié a des conséquences en terme de sécurité. Une autre façon d'exécuter le script précédent en tant qu'utiliseur `postgres` consiste à créer un binaire setuid. Pour faire cela, créez un fichier `test.cgi` en compilant le programme **C** suivant, similaire à celui de la section 34.2.
i

```
#include <unistd.h>

int main(int argc, char *argv[])
{
   setreuid (getuid (), geteuid ());
   execl ("/opt/apache/htdocs/test/test.sh", "test.sh", 0);
   return 0;
}
```

Ensuite, exécutez `chown postgres :www test.cgi` et `chmod 4550 test.cgi` (ou `chmod a-w,o-rx,u+s test.cgi`). Recréez vos scripts de shell comme `test.sh` et allez à nouveau à l'URL. Apache exécute `test.cgi`, devient utilisateur `postgres`, et donc exécute le script en tant qu'utilisateur `postgres`. Même si Apache reste utilisateur `noboby`, le script fonctionnera. Observez bien combien le programme setuid est insécurisé : il ne prend aucun argument et ne réalise qu'une seule fonction. En revanche, il prend des variables d'environnement (ou des entrées de stdin) qui peuvent influencer son comportement. Si un utilisateur pouvait exécuter le script, il pourrait envoyer des données via ces variables ce qui pourrait amener le script à se comporter d'une manière imprévue. Il existe une alternative :

```
#include <unistd.h>

int main(int argc, char *argv[])
{
   char *envir[] = {0};
   seteruid (geteuid (), geteuid ());
   execle ("/opt/apache/htdocs/test/test.sh", "test.sh", 0, envir);
   return 0;
}
```

Ce script annule l'environnement avant que le script CGI ne démarre, ce qui force l'utilisation exclusive de la méthode POST. Etant donné que la seule information qui peut être passée au script est une simple ligne de texte (ceci via l'option `-e q` de `sed`) et parce que cette ligne est passée au peigne fin concernant les caractères illicites, vous augmentez la sécurité d'utilisation du script.

37.2.11 Modules Apache et PHP.

L'exécution CGI est extrêmement lente si Apache doit invoquer un script de shell à chaque sollicitation. Apache possède un certain nombre de fonctionnalités pour des interpréteurs internes qui analysent des fichiers de scripts avec

une grande efficacité. Un langage de programmation bien connu et développé spécialement pour le web est PHP. PHP peut être téléchargé en tant que programme source depuis la *PHP Home Page* à l'adresse http ://www.php.net. Il contient les instructions GNU usuelles.

Apache présente la possibilité d'ajouter des fonctionnalités à l'exécution en utilisant les fichiers DSO (*Dynamic Shared Object*). Cette caractéristique intéresse surtout les distributions qui préparent des installations fragmentées d'Apache de sorte que les utilisateurs puissent n'installer que les parties qui les intéressent. Ceci revient en somme à ce qui a été décrit à la section 24.1 : pour donner à vos programmes des caractéristiques supplémentaires fournies par diverses bibliothèques, vous pouvez *soit* lier statiquement des bibliothèques à votre programme *ou* compiler les bibliothèques en tant que fichiers partagés `.so` de manière à ce qu'elles soient liées à l'exécution. La différence, en ce cas, est que les fichiers de bibliothèques sont usuellement appelés `mod_`*nom* et qu'ils sont enregistrés dans `/opt/apache/libexec/`. Ils sont seulement chargés si `LoadModule` *nom*`_module` apparaît dans le fichier `httpd.conf`. De manière à permettre le support DSO, reconstruisez et ré-installez Apache avec :

```
./configure --prefix=/opt/apache --enable-module=so
```

Tout paquet source qui crée un module Apache peut, à présent, exploiter l'utilitaire `/opt/apache/bin/apxs`, ce qui fait que vous devriez vous assurer que cet exécutable est dans votre `PATH`.

Dès maintenant, vous pouvez suivre les instructions pour installer PHP, en commençant par `./configure --prefix=/opt/php --with-paws=/opt/apache/bin/apxs --with-pgsql=/usr`. (Ceci suppose que vous souhaitiez permettre le support pour la base de données `postgres` SQL et que `postgres` a été installé au préalable sous `/usr`.) Enfin, vérifiez qu'un fichier `libphp4.so` a bien abouti dans `/opt/apache/libexec/`.[2]

Votre fichier `httpd.conf` doit contenir une information sur la présence des scripts PHP. Ajoutez les lignes suivantes :

```
LoadModule php4_module /opt/apache/libexec/libphp4.so
AddModule mod_php4.c
AddType application/x-httpd-php .php
```

et créez un fichier `/opt/apache/htdocs/hello.php` contenant :

```
<html>
<head>
<title>Example</title>
</head>
<body>
<?php echo "Hi, I'm a PHP script!"; ?>
</body>
</html>
```

Testez en visitant l'URL http ://localhost/hello.php.

[2] NdT : au moment de la traduction, la version 5 de PHP était disponible.

La programmation en PHP sort du cadre de cet ouvrage.

37.2.12 Hôtes virtuels.

L'utilisation d'un serveur web pour acheminer des pages web vers des domaines multiples est appelé en anglais *virtual hosting* (hébergement virtuel). Bien que le navigateur web semble être connecté à un site web qui est une entité isolée, ce site web peut être hébergé parmi de nombreux autres sites sur la même machine.

L'hébergement virtuel est assez simple à configurer. Disons que vous avez trois domaines `www.domaine1.com`, `www.domaine2.com` et `www.domaine3.com`. Nous souhaitons que les domaines `www.domaine1.com` et `www.domaine2.com` partagent la même adresse IP `196.123.45.1`, tandis que `www.domaine3.com` possède sa propre adresse IP, `196.123.45.2`. Le partage d'une adresse unique s'appelle l'hébergement virtuel basé sur le nom (*base-named virtual hosting*) et l'utilisation d'une adresse différente pour chaque domaine donne lieu à l'hébergement virtuel basé sur l'IP (*IP-based virtual hosting*).

Si notre machine ne possède qu'une IP, `196.123.45.1`, nous devrons configurer une adresse IP séparée sur la même carte réseau comme ceci (voir la section 26.9) :

```
ifconfig eth0 :1 196.123.45.2 netmask 255.255.255.0 up
```

Pour chaque domaine `/opt/apache/htdocs/www.domain?.com/`, nous créons un répertoire de référence (c'est-à-dire *top-level*). Nous devons dire à Apache que nous avons l'intention d'utiliser l'adresse IP `196.123.45.1` pour différents hôtes. Pour cela, nous utilisons la directive `NameVirtualHost`. Pour chaque hôte, nous devons spécifier le répertoire de référence (ou *top-level directory*) comme suit :

```
NameVirtualHost 196.123.45.1

<VirtualHost 196.123.45.1>
   ServerName www.domaine1.com
   DocumentRoot /opt/apache/htdocs/www.domaine1.com/
</VirtualHost>

<VirtualHost 196.123.45.1>
   ServerName www.domaine2.com

   DocumentRoot /opt/apache/htdocs/www.domaine2.com/
</VirtualHost>

<VirtualHost 196.123.45.2>
   ServerName www.domaine3.com
   DocumentRoot /opt/apache/htdocs/www.domaine3.com/
</VirtualHost>
```

Il ne reste plus qu'à configurer une zone DNS correcte pour chaque domaine de sorte que les requêtes sur `www.domaine1.com` et `www.domaine2.com` retournent

196.123.45.1 tandis que celles associées à www.domaine3.com retournent 196.123.45.2.

Vous pouvez alors ajouter les fichiers index.html à chaque répertoire.

Chapitre 38

crond et atd.

`crond` et `atd` sont deux services à la fois très simples et très importants qui devraient être familiers à tout un chacun. Le démon `crond` effectue des travaux en exécutant des commandes de manière périodique (quotidiennement, hebdomadairement, mensuellement) alors qu'`atd` présente la caractéristique de n'effectuer ultérieurement une commande qu'une seule fois.

Ces deux services sont si simples que nous n'irons pas dans le détail du contenu de leurs paquets.

38.1 Le fichier de configuration `/etc/crontab`.

Le fichier `/etc/crontab` contient la liste des travaux périodiques à effectuer – comme, par exemple, la mise à jour des bases de données de `locate` (voir la page 75) et de `whatis` (voir la page 69), les journalisations tournantes (voir la sous-section 22.4.9) et éventuellement la réalisation de tâches de sauvegarde. Si une tâche quelconque doit être effectuée périodiquement, vous pouvez la programmer dans ce fichier. `/etc/crontab` est lu par le démon `crond` au démarrage.

Après avoir modifié `/etc/crontab`, vous devez redémarrer crond avec les commandes `/etc/rc.d/init.d/crond restart` ou `/etc/init.d/crond restart` ou `/etc/init.d/cron restart`.

Le fichier `/etc/crontab` consiste en une séquence de lignes simples comprenant la définition du temps en jour/semaine/mois en vis-à-vis de laquelle une ligne de commande est spécifiée. Chaque ligne a une forme telle que :

```
<moment> <utilisateur> <executable>
```

où `<moment>` est une manière de représenter le moment auquel la commande doit être exécutée, `<utilisateur>` indique sous quel utilisateur la commande est exécutée et `<executable>` est la commande qui doit être mise en oeuvre.

La structure de `<moment>` comprend respectivement les minutes, les heures, le jour du mois, le mois et le jour de la semaine. Au moment prévu, `crond` exécute la commande. Voici un exemple :

```
50 13 2 9 6 root /usr/bin/play /etc/theetone.wav
```

où sera joué (play) theetone.wav le samedi 2 septembre de chaque année à 13 :50 :00 alors que :

```
50 13 2 * * root /usr/bin/play /etc/theetone.wav
```

jouera le fichier theetone.wav le 2ème jour de chaque mois à 13 :50 :00. Par ailleurs, la ligne :

```
50 13 * * 6 root /usr/bin/play /etc/theetone.wav
```

jouera (play) le fichier theetone.wav chaque samedi à 13 :50 :00. Notez aussi que la ligne :

```
50 13,14 * * 5,6,7 root /usr/bin/play /etc/theetone.wav
```

jouera le fichier theetone.wav à 13 :50 :00 *et* 14 :50 :00 chaque vendredi, samedi et dimanche alors que :

```
*/10 * * * 6 root /usr/bin/play /etc/theetone.wav
```

jouera le fichier theetone.wav, toutes les 10 minutes lors de chaque samedi. La notation / dans la description du moment est particulière et signifie "par pas de".

Observez bien que dans les commandes précédentes, play est exécutée en tant que root.

Ce qui suit est un cas concret de fichier /etc/crontab :

```
# Environment variables first
SHELL=/bin/bash
PATH=/sbin :/bin :/usr/sbin :/usr/bin
MAILTO=root
HOME=/

# Time specs
30 20 * * * root /etc/cron-alarm.sh
35 19 * * * root /etc/cron-alarm.sh
58 18 * * * root /etc/cron-alarm.sh
01  * * * * root run-parts /etc/cron.hourly
02  4 * * * root run-parts /etc/cron.daily
22  4 * * 0 root run-parts /etc/cron.weekly
42  4 1 * * root run-parts /etc/cron.monthly
```

Notez que le caractère # est utilisé pour les commentaires comme d'habitude. crond permet aussi de spécifier des variables d'environnement sous lesquels les commandes sont exécutées.

N'oubliez pas d'entrer les trois commandes qui vous permettront, comme ici, d'être averti des trois derniers métros du jour.

Les quatre dernières entrées appellent des scripts "tout faits". La commande run-parts est un script simple pour effectuer toutes les commandes listées sous

/etc/cron.hourly, /etc/cron.daily, etc. Ainsi, si vous avez un script qui doit être exécuté chaque jour mais pas à un moment précis, vous ne devez pas forcément éditer /etc/crontab : au lieu de cela, placez ce script parmi d'autres dans /etc/cron.<intervalle>.

A titre d'exemple, un répertoire /etc/cron.daily pourrait contenir :

```
total 14
drwxr-xr-x    2 root     root     1024 Sep  2 13 :22 .
drwr-xr-x    59 root     root     6144 Aug 31 13 :11 ..
-rwxr-xr-x    1 root     root      140 Aug 13 16 :16 backup
-rwxr-xr-x    1 root     root       51 Jun 16   1999 logrotate
-rwxr-xr-x    1 root     root      390 Sep 14   1999 makewhatis.cron
-rwxr-xr-x    1 root     root      459 Mar 25   1999 radiusd.cron.daily
-rwxr-xr-x    1 root     root       99 Jul 23 23 :48 slocate.cron
-rwxr-xr-x    1 root     root      103 Sep 25   1999 tetex.cron
-rwxr-xr-x    1 root     root      104 Aug 30   1999 tmpwatch
```

Il est conseillé de vérifier chacun des scripts de manière à savoir ce que le système fait.

38.2 La commande at.

at exécute une, et une seule, fois une commande programmée. at est un frontal (*front-end*) du démon atd qui, comme crond est très vraisemblablement déjà en cours sur votre système.[1]

Essayez de faire jouer le fichier .wav cité dans les sections précédentes, en tenant en compte le fait que presser 

permet d'obtenir un <EOT> (End-Of-Text) :

```
[root@cericon /etc]# at 14 :19
at> /usr/bin/play /etc/theetone.wav
at> <EOT>
warning : commands will be executed using /bin/sh
job 3 at 2000-09-02 14 :19
```

Vous pouvez taper atq pour obtenir une liste des travaux en cours :

```
3       2000-09-02  14 :19 a
```

a désigne le nom de la file, 3 est le numéro du travail, et 2000-09-02 14 :19 représente le moment programmé de l'exécution. Alors que play est en cours, l'éxécution d'atq retourne :

```
3       2000-09-02  14 :19 =
```

Les pages de man d'at et atd fournissent des informations supplémentaires. Notez qu'atd devrait être désactivé pour des raisons de sécurité.

[1] NdT : sur Gentoo-Linux, le paquet se trouve dans /usr/portage/sys-apps.

38.3 Les autres paquets cron.

Il y a de nombreuses mises en oeuvre de crond. Certaines possèdent des fichiers de configuration plus ou moins souples à manipuler alors que d'autres ont des fonctionnalités venant à bout de travaux à programmer qui sont effectués alors que les machines sont éteintes (ce qui peut être le cas de PCs domestiques). Votre distribution aura peut-être choisi d'office une implémentation donnée.

Chapitre 39

Serveur postgres SQL.

Ce chapitre décrit la méthode d'installation d'un serveur libre SQL.

39.1 Langage de recherche structuré (SQL).

Le langage de recherche structuré SQL (*Structured Query Language*) est un langage de programmation spécialement développé pour accéder à des données organisées sous forme de tables contenant des rangées et des colonnes –comme dans le cas des bases de données– mais aussi pour effectuer des recherches, du classement et du référencement croisé de ces données. Typiquement, les tables d'une base de données siègent dans des fichiers gérés par le *serveur* SQL (processus démon). Le serveur SQL écoute sur une *socket* SQL pour des requêtes entrantes provenant de machines clientes et il répond à ces requêtes.

SQL est devenu *de facto* un standard de l'industrie. Cependant, les protocoles (sur TCP/IP) par lesquels ces requêtes SQL sont envoyées diffèrent d'une implémentation à l'autre. Les requêtes SQL sont usuellement saisies en ligne de commande. Ceci représente une difficulté pour la plupart des utilisateurs si bien qu'une interface graphique (GUI pour *Graphic User Interface*) masque usuellement ces commandes à l'utilisateur. Les serveurs SQL et les logiciels de support SQL constituent un ensemble majeur. La gestion des tables des bases de données est un problème complexe. Un bon serveur SQL devra traiter efficacement des requêtes simultanées et multiples qui peuvent accéder à des éléments d'une même table pour les modifier. La bonne réalisation ce type d'opérations est d'autant plus complexe qu'elle doit inclure plusieurs types de recherches complexes ainsi que du référencement croisé tout en garantissant l'intégrité des données.

39.2 `postgres`.

`postgres` (*PostGreSQL*) est un serveur SQL libre écrit sous la licence BSD. `postgres` est basé sur SQL92 [le standard définitif]. Il effectue des opérations très astucieuses qu'apparemment d'autres bases ne réalisent pas. Le seul équivalent commercial capable de concurrencer `postgres` est un très coûteux logiciel provenant d'une industrie de pointe en la matière.

La documentation de postgres annonce fièrement :

> Le système de gestion de bases de données relationnelles objets, connu désormais sous le nom de PostgreSQL (et aussi appelé Postgres95), est dérivé du paquet logiciel Postgres écrit à Berkeley. Avec plus d'une décennie de développement, PostgreSQL est la base de données librement disponible la plus avancée à ce jour, offrant le contrôle multi-version, supportant presque toutes les constructions SQL (y compris les sous-sélections, les transactions, les types et les fonctions définies par l'utilisateur) et ayant une large gamme de connexions à de nombreux langages (C, C++, Java, Perl, Tcl et Python).

postgres est aussi d'une présentation vraiment minimaliste. De nombreux utilisateurs sont surpris par l'absence d'interface graphique. En fait, cette base de données fonctionnent sur tellement de plate-formes différentes qu'il est logique qu'elle soit un programme de base. [...].

Le paquet postgres consiste en fichiers qui sont décrits dans les deux sections qui suivent.

39.3 Contenu du paquet postgres.

Les paquets postgres consistent en programmes utilisateurs :

createdb	dropdb	pg_dump	psql
createlang	droplang	pg_dumpall	vacuumdb
createuser	dropuser	pg_id	

et en paquets serveurs :

initdb	pg_ctl	pg_upgarde	postgresql-dump
initlocation	pg_encoding	pg_version	postmaster
ipcclean	pg_passwd	postgres	

Chacun de ces programmes possède une page de man que vous devriez consulter.

D'autres pages de man fournissent des références relatives aux commandes SQL proprement-dites. Essayez man 1 select (expliqué un peu plus loin) :

```
SELECT (1)                                                SELECT(1)

NAME

          SELECT -- Retrieve rows from a table or view.

SYNOPSIS
          SELECT [ ALL ] DISTINCT [ ON ( expression [, ...] ) ] ]
               expression [ AS name ] [, ...]
               [ INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table ]
               [ FROM table [ alias ] [, ...] ]
               [ WHERE condition ]
               [ GROUP BY column [, ...] ]
```

```
[ HAVING condition [, ...] ]
[ { UNION [ ALL ] | INTERSECT | EXEPT } select ]
[ ORDER BY column [ ASC | DESC | USING operator ] [, ...] ]
[ FOR UPDATE [ OF class_name [, ...] ] ]
LIMIT { count | ALL } [ { OFSSET | , } start ]
```

Plus importante encore, la documentation extrêmement abondante en HTML qui accompagne postgres. Pointez votre navigateur web vers /usr/doc/postgresql-?.?.? (ou /usr/share/doc/...), et plongez-vous dans les répertoires admin, user, programmer, tutorial et postgres.

Enfin, il y a les scripts de démarrage et d'arrêt dans /etc/rc.d/init.d/ (ou /etc/init.d/) et le répertoire dans lequel la base de données, elle-même, siège : /var/lib/pgsql/.

39.4 Installation et initialisation de postgres.

postgres est disponible sous forme de paquets préparés dans votre distribution favorite. Installez simplement le paquet en utilisant rpm ou apt-get et suivez les instructions données ci-dessous.

Arrêtez le serveur postgres s'il fonctionne déjà; le script init.d peut agir sur postgres ou postgresql (commande Debian entre parenthèses) :

```
/etc/rc.d/init.d/postgres stop
( /etc/init.d/postgresql stop )
```

Modifiez le script init.d pour le support des requêtes TCP. Vous devez trouver une ligne comme suit, à laquelle il faut ajouter l'option -i. Elle peut ressembler à :

```
su -l postgres -c "/usr/bin/pg_ctl -D $PGDATA \
      -p /usr/bin/postmaster -o '-i -o -e' start >/dev/null 2>&1"
```

Le format de date 28/4/2005 peut remplacer le format 4/28/2005 si on utilise les options -o -e. Notez que les hôtes ne seront pas à même de se connecter tant que vous n'éditez pas /var/lib/pgsql/data/pg_hba.conf (ou /etc/postgresql/pg_hba.conf, sur Debian) et que vous n'ajoutez pas :

```
host    mydatabase  192.168.4.7  255.255.255.0   trust
```

Dans les deux cas, vous devrez vérifier que ce fichier garantit que seuls les hôtes autorisés peuvent se connecter à votre base de données. Sinon, enlevez tout-à-fait l'option -i si vous vous connectez seulement en local sur la machine. Jusqu'à un certain point, vous pouvez aussi restreindre les utilisateurs pouvant se connecter à ce fichier.

Il serait intéressant que la connexion (*socket*) de domaine UNIX que postgres écoute (c'est-à-dire /tmp/.s.PGSQL.5432) ait pour droits 0770 au lieu de 0777. De cette manière, vous pourriez restreindre les connexions aux groupes d'utilisateurs appartenant au groupe postgres. Vous pouvez ajouter cette caractéristique en recherchant la commande **C** chmod dans /src/backend/libpq/pgcom-

m.c à l'intérieur des sources de postgres-7.0. Il se peut que les versions ultérieures aient déjà adapté les droits à 0770.

Pour exécuter postgres, vous devez avoir un utilisateur ayant ce nom. Si ce n'est déjà fait, entrez :

```
/usr/sbin/useradd postgres
```

et redémarrez le serveur avec :

```
/etc/rc.d/init.d/postgresql restart
```

A la première exécution, le script postgres init.d initialise une base de données modèle, de sorte que vous pouvez effectuer un deuxième démarrage.

A présent, vous êtes en mesure de créer votre propre base de données. L'exemple suivant crée une base appelée finance et un utilisateur postgresql nommé finance. Ceci est fait en tant qu'utilisateur postgres (et cela est réalisé avec l'option -U). Vous devrez exécuter ces commandes en tant qu'utilisateur root ou comme utilisateur postgres sans le motif -U postgres.

```
[root@cericon]# /usr/sbin/useradd finance
[root@cericon]# createuser -U postgres --adduser --createdb finance
CREATE USER
[root@cericon]# createdb -U finance finance
CREATE DATABASE
[root@cericon]#
```

39.5 Recherches avec psql.

Maintenant que la base existe, il devient possible d'effectuer des recherches SQL.

```
[root@cericon]# psql -U finance
Welcome to psql, the PostgreSQL interactive terminal.

Type :  \copyright for distribution terms
        \h for help with SQL commands
        \? for help on internal slash commands
        \g or terminate with semicolon to execute query
        \q to quit

finance=# select * from pg_tables;
    tablename  | tableowner | hasindexes | hashrules | hastriggers
---------------+------------+------------+-----------+------------
pg_type        | postgres   | t          | f         | f
pg_attribute   | postgres   | t          | f         | f
pg_proc        | postgres   | t          | f         | f
pg_class       | postgres   | t          | f         | f
pg_group       | postgres   | t          | f         | f
```

```
pg_database    | postgres   | f          | f         | f
pg_variable    | postgres   | f          | f         | f
pg_log         | postgres   | f          | f         | f
pg_xactlock    | postgres   | f          | f         | f
pg_attrdef     | postgres   | t          | f         | f
pg_relcheck    | postgres   | t          | f         | f
pg_trigger     | postgres   | t          | f         | f
pg_inherits    | postgres   | t          | f         | f
pg_index       | postgres   | t          | f         | f
pg_statistic   | postgres   | t          | f         | f
pg_operator    | postgres   | t          | f         | f
pg_opclass     | postgres   | t          | f         | f
pg_am          | postgres   | t          | f         | f
pg_amop        | postgres   | t          | f         | f
pg_amproc      | postgres   | f          | f         | f
pg_language    | postgres   | t          | f         | f
pg_aggregate   | postgres   | t          | f         | f
pg_ipl         | postgres   | f          | f         | f
pg_inheritproc | postgres   | f          | f         | f
pg_rewrite     | postgres   | t          | f         | f
pg_listener    | postgres   | t          | f         | f
pg_description | postgres   | t          | f         | f
pg_shadow      | postgres   | f          | f         | f
(28 rows)
```

Les rangées (*rows*) qui précèdent désignent des tables internes de `postgres`. Certaines d'entre elles sont des tables à strictement parler, d'autres sont des *vues* de tables [c'est-à-dire une représentation d'une table réelle.]

Pour obtenir une liste des bases de données, essayez :

```
finance=# select * from pg_database ;

  datname   | datdba  | encoding | datpath
------------+---------+----------+----------
template1   |      24 |        0 | template1
finance     |      26 |        0 | finance
(2rows)
```

39.6 Introduction à SQL.

Ce qui suit représente 99% des commandes que vous utiliserez. (Notez que toutes les commandes SQL doivent être terminées par un point-virgule –vous ne seriez pas la première personne qui se demande pourquoi rien ne se passe en pressant la touche 

alors que la commande n'est pas terminée par un point-virgule).

39.6.1 Création de tables.

Pour créer une table nommée `people`, avec trois colonnes, exécutez :

```
CREATE TABLE people ( name text, gender bool, address text ) ;
```

La table ainsi créée intitule les colonnes : `name`, `gender` et `address`. Ces colonnes sont caractérisées par un *type*. Ceci signifie que seul le type de données qui a été spécifié lors de la création de la table peut être accepté. Dans le cas de `gender`, le type est `bool`éen et les valeurs ne peuvent valoir que *true* ou *false*, ce qui indique que le genre des personnes est soit homme ou femme. Il n'y a pas de raisons formelles à utiliser le type booléen ; l'usage d'un entier ou d'un champ "texte" est souvent plus évocateur. Dans le cas de `name` et `address`, le *type* est `text`, ce qui veut dire que ces colonnes peuvent abriter des données très variables (`text` est un type très général englobant la majorité des autres types).

Notez que dans la documentation relative à `postgres`, une "colonne" est appelée un "attribut" (et ce, pour des raisons historiques).

Vous devriez essayer de choisir des types selon le genre de *recherche* que vous ferez et *non* selon la nature des données qu'une colonne contient. Le tableau 39.1 affiche une liste des types utiles et de leur équivalent SQL92. Les types qui sont écrits en gras doivent être utilisés préférentiellement à d'autres types similaires pour une meilleure précision.

39.6.2 Afficher une table.

La déclaration `SELECT` est la plus utilisée en SQL. Elle retourne les données à partir des tables et peut effectuer des recherches :

```
finance=# SELECT * FROM PEOPLE ;
  name  | gender  | address
--------+---------+--------
(0 rows)
```

39.6.3 Ajouter une colonne.

La commande ALTER permet de faire une modification :

```
finance=# ALTER TABLE people ADD COLUMN phone text ;
ALTER
finance=# SELECT * FROM people ;
  name  | gender  | address  | phone
--------+---------+----------+------
(0 rows)
```

39.6.4 Supprimer une colonne.

Vous ne pouvez pas supprimer de colonnes dans `postgres`(*dropping*) ; vous devez créer une nouvelle table à partir de l'ancienne sans la (ou les) colonne(s) en

TAB. 39.1 – Types usuels de postgres.

Postgres Type	Type SQL92 ou SQL3	Description
bool	boolean	booléen logique (vrai/faux)
box		rectangle sur un plan
char(n)	character(n)	chaîne de caractères de longueur fixe
cidr		IP de réseau version 4 ou adresse d'hôte
circle		cercle dans un plan
date	date	date du calendrier sans le jour
decimal	decimal(p,s)	nombre exact pour p |= 9, s = 0
float4	float(p),p | 7	nombre à virgule flottante de précision p
float8	float(p),7 |=p | 16	nombre à virgule flottante de précision p
inet		IP version 4 ou adresse d'hôte
int2	smallint	entier signé à 2-octets
int4	int, integer	entier signé à 4-octets
int8		entier signé à 8-octets
interval	interval	période à usage général
line		ligne de longueur infinie
lseg		segment de ligne
money	decimal(9,2)	monnaie en style US
numeric	numeric(p,s)	nombre exact pour p == 9, s = 0
path		courbe ouverte ou fermée dans un plan
point		point géometrique
polygon		polygone régulier ou irrégulier
serial		ID simple pour faire un indice ou une référence croisée
time	time	temps courant
text		texte de longueur arbitraire (jusqu'à 8k dans le cas de postgres 7)
timetz	time with time zone	temps avec zone horaire
timestamp	timestamp with time zone	date/temps avec zone horaire à grande précision
varchar(n)	character varying(n)	chaîne de caractères de longueur quelconque

question. Ceci deviendra simple à réaliser lorsque vous aurez pris connaissance de la sous-section 39.6.9.

39.6.5 Supprimer une table.

Utilisez la commande DROP pour supprimer une table :

```
DROP TABLE people;
```

39.6.6 Insertion de lignes, objet relationnel.

Insérez une ligne de la manière suivante (vous pouvez continuer à encoder sur de multiples lignes) :

```
finance=# INSERT INTO people (name, gender, address, phone)
finance=# VALUES ('Paul Sheer', true, 'Earth', '7617224');
INSERT 20280 1
```

La valeur retournée est l'oid (*Object ID*) de la rangée. postgres est une base de données relationnelle objet (*Object Relational DB*). Ce terme signifie que chaque table a une colonne cachée appelée colonne oid qui stocke un numéro d'identité unique pour chaque rangée. Le numéro d'identité est unique dans toute la base de données. Du fait qu'il identifie de manière non-ambiguë chaque rangée parmi toutes les tables, vous qualifiez les rangées, "d'objets". La caractéristique oid est très utile aux programmeurs.

39.6.7 Localisation de lignes.

L'oid de la rangée précédente est 20280. Pour retrouver cette rangée :

```
finance=# SELECT * FROM people WHERE oid = 20280;
    name    | gender  | address  | phone
------------+---------+----------+--------
 Paul Sheer | true    | Earth    | 7617224
(1 row)
```

39.6.8 Afficher une colonne sélectionnée, et la colonne oid.

Pour afficher les colonnes sélectionnées, essayez :

```
SELECT name, address FROM people;
SELECT oid, name FROM people;
SELECT oid, * FROM people;
```

Le résultat de ces commandes est très explicite.

39.6.9 Créer des tables à partir d'autres tables.

A présent, nous créons une nouvelle table et remplisons deux de ses colonnes par les colonnes de la table originale :

```
finance=# CREATE TABLE sitings (person text, place text, siting text);
CREATE
finance=# INSERT INTO sitings (person, place) SELECT name, address FROM
people;
INSERT 20324 1
```

39.6.10 Suppression de rangées.

On supprime des rangées sélectionnées de la manière suivante :

```
finance=# DELETE FROM people WHERE name = 'Paul Sheer';
DELETE 1
```

39.6.11 Recherches.

La recherche la plus simple que vous pouvez réaliser avec `postgres` est :

```
SELECT * FROM people WHERE name LIKE '%Paul%';
```

Si vous recherchez un motif indépendamment de la casse et ce, dans le champ `address` :

```
SELECT * FROM people WHERE lower(name) LIKE '%paul%' OR lower(address) LIKE
'%paul%';
```

Le premier signe `%` est un caractère de remplacement qui correspond à tout texte avant `Paul` et le `%` terminal correspond à tout texte se trouvant après `Paul`. Il s'agit d'une méthode de recherche tout-à-fait usuelle préférable à la recherche d'une correspondance exacte.

Les possibilités de recherche sont extrêmement vastes :

```
SELECT * FROM people WHERE gender = true AND phone = '8765432';
```

39.6.12 Migration à partir d'une autre base de données ; transfert et restauration de tables sous forme de texte.

La commande :

```
COPY people TO '/tmp/people.txt';
```

transfère la table `people` vers `/tmp/people.txt`, avec les tabuluations comme délimiteurs et des rangées se terminant par une mise à la ligne.

La commande :

```
COPY people WITH OIDS to '/tmp/people.txt' DELIMITERS ',' WITH NULL AS
'(null)';
```

transfère la table `people` vers `/tmp/people.txt`, avec des virgules comme délimiteurs, des rangées terminées par une mise à la ligne et l'indication `(null)` là où il peut y avoir un octet zéro.

De manière analogue, la commande

```
COPY people FROM '/tmp/people.txt';
```

insère dans la table `people` les rangées enregistrées dans `/tmp/people.txt`. Cela suppose une ligne par rangée de la table et une tabulation entre chaque cellule.

Notez que les caractères non-imprimables doivent être précédés du caractère d'échappement `\` *à la fois en sortie et dans l'interprétation des données en entrée.*

Donc, il est simple d'obtenir des données provenant d'autres bases de données. Vous ne devez vous occuper que de la manière de les transférer sous forme de texte.

39.6.13 Transfert d'une base de données complète.

La commande `pg_dump <nom_base_de_donnees>` transfère votre base de données sous forme de texte (*plain text*). Si vous essayez de pratiquer ainsi sur votre base de données, vous noterez que le résultat contient des commandes SQL. Votre base peut être reconstruite *ab initio* en envoyant par un tube (*pipe*) cette sortie via l'entrée standard (stdin) de la commande `psql`. En d'autres termes, `pg_dump` produit simplement la séquence exacte de commandes SQL nécessaires à la reconstruction de votre base de données.

Il peut arriver qu'une nouvelle version de `postgres` réalise une commutation des données dans un format de fichiers incompatible avec des fichiers anciens. Dans ce cas, il est prudent d'utiliser `pg_dumpall` avant de faire une mise-à-jour (et de sauver le résultat). Le résultat de `pg_dumpall` peut alimenter la stdin de la commande `psql`. Elle contient aussi toutes les commandes nécessaires pour recréer toutes vos bases de données ainsi que pour formater correctement les données contenues.

39.6.14 Recherches plus poussées.

Quand vous êtes face à un jeu de tables complexes, vous pouvez envisager d'opérer des fusions, des sélections, des recherches et du référencement croisé de différentes manières pour obtenir l'information que vous voulez en retirer.

La véritable puissance de SQL, c'est de vous offrir la possibilité de réaliser des recherches de manière efficace. Dans le présent ouvrage, nous n'en sommes qu'à une introduction. La documentation `postgres` citée ci-dessus contient les informations pour vous permettre de devenir un expert.

39.7 Projets réels de base de données.

Les scientifiques spécialisés en informatique étudient des sujets comme la *modélisation d'objets*, l'*algèbre relationnel* et la *normalisation des bases de données*. Il s'agit de méthodes académiques sur lesquelles les bonnes bases de données sont developpées. Vous ne devriez pas vous aventurer sur le terrain de l'élaboration de bases de données complexes sans connaître ces méthodes.

La plupart des bibliothèques universitaires possèdent des ouvrages permettant d'apprendre la théorie sur laquelle se fondent les bases de données.

Chapitre 40

smbd – le projet NT Samba.

L'introduction qui est présentée ci-dessous est extraite de la documentation en ligne de Samba.

40.1 Une introduction par Christopher R. Hertel

L'accent a souvent été mis sur la nécessité d'une coexistence pacifique entre UNIX et Windows. Malheureusement, les deux systèmes proviennent de cultures différentes et ont des difficultés de communication à moins qu'un subterfuge ne soit employé. C'est là qu'intervient Samba. *Samba* http ://samba.org/ est exécuté sur des plate-formes UNIX, tout en communicant avec des clients Windows, de manière native. Ce logiciel permet à un système UNIX d'interagir dans un "environnement réseau" Windows sans provoquer de troubles. Les utilisateurs Windows accèdent aux fichiers et aux services d'impression sans se rendre compte qu'en réalité, ceux-ci sont fournis par un serveur UNIX.

Ces services sont gérés grâce à une suite de protocoles communément connus sous le terme "Common Internet File System", ou *CIFS* http ://www.cifs.com. Ce nom a été introduit par Microsoft et il donne une idée de la vision qu'ont les membres de cette compagnie pour le futur. Au coeur du CIFS se trouve la dernière incarnation du protocole SMB (***S****erver* ***M****essage* ***B****lock*), qui a une histoire à la fois longue et difficile. Samba est une implémentation libre du CIFS ; elle est disponible gratuitement sur les sites miroirs d'http ://samba.org.

Samba et Windows ne sont pas les seuls systèmes à fournir un réseau CIFS. OS/2 supporte les fichiers SMB et le partage d'imprimantes ; d'autres produits CIFS existent pour MacIntosh et d'autres plate-formes (en ce compris, divers UNIX). Samba a été porté sur une série de systèmes d'exploitation non-UNIX, comme VMS, AmigaOS et NetWare. CIFS est aussi supporté sur des plate-formes spécifiques, serveurs de fichiers. Il s'agit donc d'un système très répandu.

40.1.1 Historique – version (supposée) non-ennuyeuse.

L'histoire commence, il y a bien longtemps, dans les premiers temps mêmes des PCs, lorsqu'IBM et Sytec co-développent un système réseau destiné à de petits réseaux locaux (LANs). Le système comprend un dispositif appelé NetBIOS (***N****etwork* ***B****asic* ***I****nput* ***O****utput* ***S****ystem*). NetBIOS est une grosse pièce logicielle chargée en mémoire de manière à fournir une interface entre les programmes et le matériel réseau. Il comprend un schéma d'adressage utilisant des noms à 16 octets pour identifier les stations de travail et les applications réseau. Par la suite, Microsoft ajoute des propriétés à DOS

pour permettre aux E/S des disques d'être *redirigées* vers l'interface NetBIOS, ce qui rend l'espace disque partageable sur le réseau. A l'époque, le protocole de partage de fichiers utilisé par Microsoft reçoit le nom de SMB puis devient CIFS.

De nombreux autres logiciels sont écrits afin d'utiliser l'interface de programmation d'applications du NetBIOS (Application Programmer's Interface), ce qui signifie que tous ces développements n'auraient jamais –au grand jamais– dû disparaître. Cependant, au fil du temps, les mécanismes sous-jacents à l'API sont habilement "étripés" et remplacés. NetBEUI (***NetB****IOS* ***E****nhanced* ***U****ser* ***I****nterface*), introduit par IBM, fournit alors un mécanisme pour passer des paquets NetBIOS sur Token Ring et Ethernet. D'autres développent une émulation NetBIOS LAN incluant des protocoles de haut-niveau comprenant DECnet, IPX/SPX et, bien sûr, TCP/IP.
NetBIOS et TCP/IP constituent un tandem intéressant. TCP/IP peut être routé entre divers réseaux interconnectés (*internetworks*), mais NetBIOS a, dès le départ, été élaboré pour les LANs isolés. L'astuce consiste donc à convertir les noms à 16 octets de NetBIOS en adresses IP de telle sorte que les messages puissent vraiment trouver leur chemin dans un réseau IP routé. Les documents RFC1001 et RFC1002 relatent le mécanisme en détail. A mesure que Windows évolue, Microsoft ajoute deux éléments au paquet SMB. Il s'agit du "browsing" ainsi que du service d'authentification et d'autorisation centralisées, connu sous le nom de *Windows NT Domain Control*.

40.1.2 Pendant ce temps, de l'autre côté de la planète...

Andrew Tridgell, un australien, rencontre un petit problème. Il doit monter un espace disque d'un serveur UNIX sur son PC fonctionnant sous DOS. En fait, ceci n'est pas vraiment un problème puisqu'il dispose d'un client NFS (***N****etwork* ***F****ile* ***S****ystem*) pour DOS, fonctionnant bien. Malheureusement, il a aussi une application qui requiert l'interface NetBIOS. Toute personne qui a tenté de faire fonctionner de multiples protocoles sous DOS sait que cela peut être .*rè..biz*rre.
Aussi, Andrew choisit-il une solution très logique. Il écrit un renifleur de paquets, effectue de l'ingénierie inverse sur le protocole SMB et met le résultat en oeuvre sur son serveur UNIX. Ce faisant, il permet au système UNIX d'apparaître comme un serveur de fichiers pour PC. Ceci lui permet de monter les systèmes de fichiers partagés depuis le serveur UNIX tout en exécutant simulatanément les applications NetBIOS. Andrew publie son code au début 1992. S'en suit alors un rapide débogage et le projet est mis de côté. Parfois, des méls parviennent à Andrew mais souvent il les ignore. Pratiquement deux ans plus tard, il entreprend de relier le PC (sous Windows) de son épouse à sa station de travail LINUX. En l'absence d'une meilleure solution, il se sert à nouveau de son code serveur et est agréablement surpris de constater que cela fonctionne.
Parmi ses contacts méls, Andrew apprend que NetBIOS et SMB sont désormais bien documentés. En possession de cette nouvelle information, il se remet au travail sur son code mais il rencontre rapidement un autre problème. Il est contacté par une entreprise réclamant des droits sur le nom qu'il a choisi pour son logiciel serveur. Plutôt que de s'engager dans une polémique, il consulte le dictionnaire à la recherche des mots contenant les lettres "smb". Samba est de la liste. Curieusement, le même mot n'est pas dans le fichier dictionnaire qu'il utilise aujourd'hui.

Depuis cette époque, le projet Samba s'est puissamment développé. Andrew dirige maintenant toute une équipe de programmeurs répartis dans le monde entier pour continuer le développement de Samba. Lorsqu'une nouvelle version est annoncée, des milliers de copies sont téléchargées journellement. Les entreprises commerciales, comme Silicon Graphics, empaquettent Samba dans leurs produits. Il existe même des tee-shirts Samba. La meilleure indication du succès de ce logiciel est sans doute d'avoir été retenu parmi les "Documents d'Halloween", une série de mémos internes à Microsoft qui sont parvenus jusque dans la Communauté du Libre. Ces mémos contiennent une série de noms de logiciels libres que Microsoft considère compétitifs. Cependant, le plus

important de toute cette affaire, c'est qu'Andrew peut toujours partager l'imprimante de son épouse.

40.1.3 Ce que fait Samba.

Samba consiste en deux programmes-clés, et une série d'autres logiciels dont nous reparlerons plus tard. Les deux programmes fondamentaux sont `smbd` et `nmbd`. Leur mission consiste à mettre en oeuvre les quatre services de base de CIFS dans sa version moderne, c'est-à-dire :

- serveurs de fichiers et d'impression,
- authentification et autorisation,
- résolution de noms,
- annonces de services ("browsing").

Bien sûr, les services relatifs aux fichiers et à l'impression constituent, la pierre angulaire de la suite CIFS. Ils sont fournis par `smbd`, le démon de SMB. Le démon `smbd` gère aussi l'authentification et l'autorisation en "mode utilisateur" et en "mode partagé". Vous pouvez protéger les services de partage de fichiers et d'imprimantes en exigeant des mots de passe. En mode partagé, qui est le schéma le plus simple, un mot de passe peut être attribué à un répertoire partagé ou à une imprimante (appelé dans le jargon anglophone : "*share*", c'est-à-dire "partage"). Ce simple mot de passe est alors donné à toute personne autorisée à utiliser le partage. Avec l'authentification en mode utilisateur, chaque utilisateur possède ses propres mots de passe et identifiant de connexion. L'administrateur système peut accorder ou refuser l'accès sur une base individuelle.

Le système *Windows NT Domain* fournit un niveau supplémentaire en terme d'authentification pour CIFS. L'idée de base est qu'un utilisateur ne devrait pouvoir se connecter qu'une seule fois pour accéder à tous les services sur le réseau. Le système *NT Domain* gère ceci avec un serveur d'authentification appelé contrôleur de domaine (*Domain Controller*). Un *domaine NT* (qui ne devrait pas être confondu avec un ***D****omain* ***N****ame* ***S****ystem* –DNS) est un groupe de machines qui partage le même *Contrôleur de domaine*.

Le système *domaine NT* mérite une remarque parce qu'avant la version 2 de Samba, Microsoft possédait le code des protocoles d'authentification du *domaine NT*. Avec la version 2, Samba a introduit son premier code propre d'authentification de *domaine NT*, non-dérivé de Microsoft. Bien sûr, le but est de mimer complètement un contrôleur de domaine Windows NT.

Les deux autres composantes de CIFS, la résolution de nom et la navigation, sont gérés par `nmbd`. Ces deux services impliquent fondamentalement la gestion et la distribution des listes de noms de NetBIOS.

La résolution de noms prend deux formes : la diffusion (*broadcast*) et le point-à-point. Une machine peut utiliser l'une ou l'autre méthode ou les deux, selon sa configuration. La résolution par diffusion est la plus proche du mécanisme originel de NetBIOS. Un client cherchant un service nommé `Trillian` appele : "`Hé, Trillian ! Ou êtes-vous ?`", et attend que lui réponde la machine ayant ce nom, en lui fournissant son adresse IP. Cette technique peut engendrer un certain volume de trafic réseau (comme c'est le cas avec de nombreux cris dans la rue), toutefois comme ceci est restreint au réseau LAN, il n'en résulte pas une forte gêne.

L'autre méthode de résolution comprend l'usage d'un serveur NBNS (acronyme pour ***N****et****B****IOS* ***N****ame* ***S****ervice*). (L'implémentation NBNS de Microsoft s'appelle WINS, pour Windows Internet Name Service ; cet acronyme est couramment utilisé de nos jours.) Le serveur NBNS fonctionne un peu à la manière des murs d'un ancienne cabine téléphonique [...]. Les machines peuvent laisser aux autres le soin de voir leur nom et leur numéro IP.

Bonjour, je suis Voomba. Appelez-moi au 192.168.100.101

Voici comment cela fonctionne : les clients envoient leur nom NetBIOS et leur adresse

IP au serveur NBNS qui conserve l'information dans une base de données. Quand un client veut entrer en communication avec un autre, il envoie le nom de ce dernier au serveur NBNS. Si ce nom est dans la base, le serveur NBNS renvoie une adresse IP.
Les clients situés sur divers sous-réseaux peuvent aussi partager le serveur NBNS ; aussi, sans diffusion, le mécanisme point-à-point n'est-il pas restreint au réseau local LAN. Par de nombreux aspects, le serveur NBNS présente des similitudes avec un DNS. Toutefois, la liste de noms de NBNS est presque totalement dynamique et, il y a peu de contrôle quant au fait que seuls les clients autorisés peuvent s'enregistrer. Des conflits peuvent –et, de fait, doivent inévitablement– se produire.
Enfin, il y a la navigation (*browsing*). [...] Samba gère cet aspect. Naturellement, il ne s'agit pas de la navigation web que nous connaissons bien, mais d'une liste de services (fichiers et imprimantes) qui peut être parcourue, et qui est fournie par les ordinateurs sur le réseau.
Sur un réseau local (LAN), les ordinateurs participants sont l'objet d'une élection pour déterminer lequel sera le navigateur-maître, Local Master Browser (LMB). L'élu s'identifie en indiquant aux autres un nom de NetBIOS spécial (en plus d'autres noms qu'il peut éventuellement avoir). Le travail de ce LMB est de conserver une liste des services disponibles. C'est cette liste qui apparaît lorsqu'on clique sur l'icône de Windows appelée "Network Neighborhood" (voisinage réseau).

En plus du LMB, il y a des navigateurs-maîtres de domaine, Domain Master Browsers (DMBs). Les DMBs coordonnent les listes de navigateur dans les domaines NT, même sur des réseaux avec routage. En utilisant les DMBs, le LMB localise son DMB pour échanger et combiner les listes de navigateurs. Donc, la liste se propage sur tous les hôtes du domaine NT. Malheureusement, les durées de synchronisation sont critiques. Il arrive que cela prenne plus d'une heure pour qu'une modifiaction sur un sous-réseau distant soit prise en considération dans le "voisinage réseau".

40.1.4 Autres cas.

Samba comprend une série d'utilitaires dont les plus courants sont :

smbclient Il s'agit d'un simple client SMB avec une interface similaire à celle d'un client FTP. Il peut être utilisé depuis un système UNIX pour se connecter à un partage SMB distant, transférer des fichiers et envoyer des fichiers à des partages d'impression (c'est-à-dire des imprimantes).

nmblookup Il s'agit d'un client du service de noms de NetBIOS. Nmblookup peut être utilisé pour trouver des noms NetBIOS sur le réseau, faire des recherches sur leur adresse IP, et demander à une machine distante la liste des machines qu'elle pense posséder.

swat C'est l'acronyme pour ***S****amba* ***W****eb* ***A****dministration* ***T****ool*. Swat vous permet de configurer Samba à distance, en utilisant un navigateur web.

Il y a bien plus de fonctionnalité, mais pour en comprendre l'intérêt, il faudrait approfondir divers aspects de CIFS, SMB et Samba. C'est là où les choses deviennent plus fastidieuses et donc, pour l'heure, nous les laissons de côté.

40.1.5 Système de fichiers SMB pour LINUX.

Il est agréable avec un ordinateur sous Windows d'utiliser le partage de fichiers SMB comme si ces fichiers étaient sur votre propre disque dur. Le périphérique N : peut sembler se comporter comme votre propre espace disque mais il s'agit en réalité de l'espace disque d'une autre machine se trouvant sur le réseau.

Les systèmes Linux peut aussi fonctionner de la même manière, en utilisant le système de fichiers smbfs. Construit avec le code Samba, smbfs (qui signifie ***SMB***

***F**ile **S**ystem*) permet à Linux de répartir les partages SMB distants dans une structure propre de répertoires. Ainsi, par exemple, le répertoire /mnt/zarquon pourrait être un partage SMB. Dans ce répertoire, vous pourriez lire, écrire, modifier, supprimer et copier des fichiers de la même manière que vous le faites avec des fichiers locaux.

Le système de fichiers smbfs est asticieux, mais il ne fonctionne qu'avec Linux. En fait, il ne fait pas partie de la suite Samba et est distribué gratieusement mais seulement comme une fonctionnalité supplémentaire. Le nouveau paquet smbsh constitue une solution plus génerale (***SMB sh**ell* était encore en développement lors de la préparation de ce texte –version anglaise originale). C'est un paquet très intéressant. Il fonctionne comme un shell UNIX mais il effectue des opérations subtiles avec des appels à des bibliothèques UNIX. En interceptant ces appels, smbsh donne l'impression que les partages SMB sont montés. Toutes les opérations d'écriture, lecture, etc. sont disponibles pour l'utilisateur smbsh. Une autre caractéristique de smbsh est qu'il fonctionne sur une base utilisateur ou shell alors que le montage d'un système de fichiers concerne le système dans son ensemble. Dès lors, avec smbsh, le contrôle des accès est plus fin.

40.1.6 Mise en place et gestion.

Le fichier central de configuration de Samba est `smb.conf`. Il s'agit d'un simple fichier texte ressemblant aux fichiers *.ini de Windows. Le but est de procurer aux administrateurs habitués à Windows un fichier aisé à manipuler. Cependant, avec le temps, le nombre de fonctionnalités à configurer a augmenté et le nombre d'administrateurs réseau prêt à éditer un fichier *.ini a considérablement diminué. Pour certaine personne, la gestion du fichier `smb.conf` est quelque peu intimidante.
Ceci dit, l'étude détaillée de `smb.conf` est une tâche du plus haut intérêt, chaque réglage de variables permettant d'arriver à un contrôle très fin. Chaque élément du fichier est largement documenté de manière à aider les administrateurs au mieux. Le fichier `smb.conf` peut être modifié à l'aide de `swat`, qui le formate joliment.

40.1.7 De la version 2.0 à la version 3.0.

> ***Note de traduction :*** *la traduction stricte du document de C. R. Hertel, tel que contenu dans* Rute User's Tutoriel and Exposition *et, accessible directement à l'adresse http ://us4.samba.org/samba/docs/SambaIntro.html aurait pu laisser penser au lecteur que les modifications* actuelles *portant sur les nouvelles versions de Samba sont celles associées au passage à la version Samba-2.0. Or, en 2005, nous en sommes à Samba-3.0. Les sous-sections intitulées "Le présent" et "Le futur" de l'article original de C. R. Hertel ont donc été traduites de manière abrégée. Nous les avons complétées avec les sections "Ce qu'il y a de nouveau dans Samba-2.2" et "Ce qu'il y a de nouveau dans Samba-3.0" du document internet intitulé* Using Samba *– 2nd edition de Jay Ts, Robert Eckstein et David Collier-Brown (http ://us4.samba.org/samba/docs/using_samba/toc.html).*

40.1.7.1 Samba-2.0.

La version Samba-2.0 est sortie en janvier 1999. Une des caractéristiques majeures était l'amélioration de la vitesse de traitement. Ziff-Davis Publishing a réalisé un test comparatif Samba/Linux – Windows NT4 sur le même matériel PC. Les résultats montraient que la rapidité de traitement par Samba sous charge, été améliorée d'un facteur deux au moins par rapport à celle de NT4. Samba était déjà fourni avec toutes les distributions Linux majeures et Ziff-Davis en avait testé trois.
Une autre étape a été franchie lorsque Silicon Graphics (SGI) est devenu le premier

vendeur d'UNIX supportant Samba. [...], SGI affirmait que les serveurs de la série Origin exécutant Samba 2.0 constituaient la ligne de serveurs disponible la plus puissance pour les clients Windows. Depuis, SGI offre un support commercial à Samba comme le font d'autres fournisseurs (voir le site Samba http ://samba.org/). Le support traditionnel est naturellemement resté disponible via comp.protocols.smb et la liste de diffusion samba@samba.org.

L'équipe de Samba a continué son travail avec l'inclusion des listes d'accès, NT ACL (***A**ccess **C**ontrol **L**ists*), le support de LDAP (***L**ightweith **D**irectory **A**ccess **P**rotocol*), le contrôle de domaine NT (*NT Domain Control*) et le système de fichiers distribués de Microsoft (DSF, ***D**istributed **F**ile **S**ystem*).

Ensuite, Windows 2000 sortit. En raison de sa domination sur les ordinateurs de bureau, Microsoft dût décider de l'avenir de CIFS. En fonction des informations relatives aux versions beta et des commentaires de Microsoft, voici une liste d'éléménts qui fût prise en considération à l'époque :

CIFS sans NetBIOS Microsoft essaie de découpler CIFS et NetBIOS. NetBIOS ne sera pas écarté mais il ne sera plus réquis pour le réseau CIFS. Le protocole SMB sera fourni nativement sur TCP/IP. Les requêtes de noms se feront via le DNS.

DNS dynamique Microsoft mettra en oeuvre le DNS Dynamique, un système encore en cours d'élaboration par l'IETF (***I**nternet **E**ngineering **T**ask **F**orce*). Le DNS dynamique permet aux noms d'être ajoutés à la volée sur un serveur DNS.

Kerberos V Microsoft a l'intention d'utiliser Kerberos V avec un PAC (***P**rivilege **A**ttribute **C**ertificate*) : http ://www.unsenix.org/publications/login/1997-11/embraces.html, qui comprendra une information ID de groupe et d'utilisateur dans le répertoire actif (*Active Directory*). Les serveurs rechercheront ce PAC lorsqu'ils autoriseront l'accès aux services qu'ils fournissent. Kerberos servira pour l'authentification et les autorisations.

Répertoire actif Le répertoire actif (Active Directory) semble être au coeur du réseau Windows 2000. Il est vraisemblable que les services NetBIOS enregistreront leurs noms dans le répertoire actif.

Domaines_NT_hiérarchiques Au lieu des contrôleurs de domaines isolés, le système NT Domain deviendra hiérarchique. Le système d'attribution de nom sera modifié en un système remarquablement similaire à DNS.

40.1.7.2 Samba-2.2.

Dans la version 2.2, Samba présente davantage de support pour la gestion réseau de Windows y compris la capacité de réaliser des tâches plus lourdes pour agir dans un domaine NT. En outre, Samba-2.2 présente un support amélioré pour les technologies que Microsoft a introduit avec Windows 2000, bien que l'équipe de Samba ait sauvé le support de répertoire actif (Active Directory) pour la version 3.0.

Support PDC pour client Windows 2000/XP. Samba pouvait jusqu'ici être considéré comme un PDC (contrôleur de domaine principal ou ***P**rimary **D**omain **C**ontroller*) capable d'authentifier des sytèmes Windows 95/98/Me et NT 4. Cette fonctionnalité a été étendue de manière à inclure les systèmes Windows 2000 et XP. Il est donc possible de disposer d'un serveur Samba fonctionnant avec tout l'éventail de clients Windows. Ceci assure une grande stabilité, une grande performance et un réseau plus sécurisé. Cela vous procure donc les avantages associés aux systèmes plus récents sans devoir se procurer la licence d'accès client de Windows (***C**lient **A**ccess **L**icence* ou CAL, qui donne le droit à une machine d'accéder aux services de fichiers et d'impression d'un serveur NT).

Support Dfs de Microsoft. Le support Dfs permet de recueillir (afin de les partager) des ressources dispersées parmi un certain nombre de serveurs du réseau. Il permet de les faire apparaître à un utilisateur comme si elles étaient regroupées dans une arborescence unique sur un seul serveur. Cette technique rend l'utilisation des ressources plus conviviale. Au lieu de devoir naviguer dans le réseau à la recherche de la ressource à localiser, les utilisateurs peuvent directement aller sur le serveur Dfs et y collecter ce qu'ils recherchent. Samba 2.2 offre un serveur Dfs, si bien qu'un serveur Dfs Windows n'est plus nécessaire.

Support d'impression de Windows NT/2000/XP. L'interface d'impression basée sur RPC (***R**emote **P**rocedure **C**all*)[1] n'est plus la même sous Windows NT/2000/XP que sous Windows 95/98/Me. Avec Samba-2.2, l'interface pour Windows NT/2000/XP est désormais supportée. Par ailleurs, l'équipe Samba a introduit la possibilité de charger automatiquement le pilote d'une imprimante à partir du serveur Samba, lorsqu'une nouvelle imprimante est ajoutée.

ACLs. Désormais, Samba supporte les ACLs (***A**ccess **C**ontrol **L**ists*) de l'hôte UNIX, du moins pour les variantes UNIX qui en offre le support. Ceci comprend Solaris 2.6, 2.7 et 2.8, Irix, AIX, Linux (soit via le correctif ACL des systèmes de fichiers ext2 et ext3 [http ://acl.bestbits.at], soit via le système de fichiers XFS) et FreeBSD 5.0 (et versions supérieures). Quand le support ACL est utilisé, Samba traduit les ACLs Windows NT/2000/XP en ACLs UNIX, ce qui rapproche encore davantage –du point de vue d'un client Windows– le serveur Samba d'un serveur Windows.

Support pour les outils d'administration clients de Windows. Windows contient des outils pouvant être utilisés par un client pour gérer des ressources partagées à distance, sur un serveur Windows. Samba-2.2 permet également à ces outils de fonctionner sur les partages du serveur Samba.

Intégration avec Winbind. Winbind est une fonctionnalité permettant aux utilisateurs dont le compte est enregistré dans une base de données d'un domaine Windows d'authentifier un système UNIX. Ceci conduit à un environnement relativement unifié dans lequel un compte utilisateur peut être conservé soit sur un système UNIX, soit sur un contrôleur de domaine Windows NT/2000/XP. Il en résulte une plus grande facilité de gestion des comptes car les administrateurs ne doivent plus faire attention à la synchronisation des systèmes. Par ailleurs, il devient ainsi possible aux utilisateurs dont le compte est conservé sur un domaine Windows de s'authentifier lors de leurs accès aux partages Samba.

Extension CIFS d'Unix. Les extensions CIFS d'Unix ont été développées par Hewlett-Packard et introduite dans Samba-2.2.4. Elles permettent aux serveurs Samba de supporter les attributs des systèmes de fichiers UNIX. Samba peut donc être utilisé comme alternative au partage de fichiers NFS ayant lieu entre machines UNIX. En l'occurrence, l'avantage de Samba est qu'il authentifie de manière individuelle les utilisateurs alors que NFS n'authentifie que les clients sur base de leur adresse IP (ce qui est faible en terme de sécurité). De plus, la haute configurabilité de Samba lui permet de jouer un rôle important en terme de sécurité, les systèmes UNIX pouvant donc être des clients Samba.

[1] NdT : il s'agit d'une technique utilisée dans un cadre "client-serveur". Le client appelle des procédures qui sont exécutées sur un ordinateur distant grâce à un serveur d'applications. Le protocole RPC gère les interactions entre le client et le serveur.

40.1.7.3 Samba-3.0.

La différence majeure de cette version est qu'elle comprend le support pour l'authentification par Kerberos 5 et LDAP (***L**ightweigth **D**irectory **A**ccess **P**rotocol* : protocole d'annuaire sur TCP/IP) qui sont nécessaires pour agir en tant que clients dans un domaine du répertoire actif. On trouve aussi le support pour l'Unicode qui simplifie grandement l'accès aux langages internationaux.

Dans les versions ultérieures de type 3.0.x, l'équipe de développement projette de développper la réplication de WINS ce qui permettra à Samba d'agir comme serveur WINS secondaire ou comme serveur WINS primaire avec des serveurs WINS Windows ou Samba secondaires. Il est aussi envisagé de faire agir Samba comme BDC de Windows NT et de supporter les rapports de confiance dans un domaine Windows NT.

40.2 Configuration de Samba.

A vrai dire, la configuration de `smbd` est vraiment facile. Un réseau LAN typique contient une machine UNIX permettant de partager les répertoires `/home/*` vis-à-vis des clients Windows, chaque utilisateur pouvant se connecter avec le même nom que celui du répertoire `/home/` correspondant. La machine UNIX agit comme partage d'imprimante en redirigeant les travaux d'impression via la commande `lpr`, et via PostScript, une méthode que nous aimons particulièrement. Considérons une machine Windows appellée `divinian.cranzgot.co.za` sur un réseau local LAN `192.168.3.0/24`. L'utilisateur de cette machine a pour identifiant de connexion (*login*) UNIX : `psheer`, sur le serveur `cericon.cranzgot.co.za`.

Le fichier de configuration de Samba est `/etc/samba/smb.conf` sur la majorité des distributions. Un fichier minimaliste pour réaliser les opérations citées ci-dessus sera :

```
[global]
   workgroup = MYGROUP
   server string = Samba Server
   hosts allow = 192.168.127.
   printcap name = /etc/printcap
   load printers = yes
   printing = bsd
   log file = /var/log/samba/%m.log
   max log size = 0
   security = user
   socket options = TCP_NODELAY SO_RCVBUF=8192 + SO_SNDBUF=8192
   encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd
[homes]
   comment = Home Directories
   browseable = no
   writable = yes
[printers]
   comment = All Printers
   path = /var/spool/samba
```

```
browseable = no
guest ok = no
printable = yes
```

Le protocole SMB enregistre les mots de passe de manière différente que sous UNIX. Il requiert donc un fichier de mots de passe, usuellement `/etc/samba/smbpasswd`. Il y a aussi une table de correspondance des noms de connexion UNIX et Samba dans `/etc/samba/smbusers`, mais pour des raisons de simplicité, nous utiliserons les mêmes *login* sous Samba et UNIX. Nous ajoutons un nouvel utilisateur UNIX et Samba et, créons les mots de passe de cette manière :

```
smbadduser psheer :psheer
useradd psheer
smbpasswd psheer
passwd psheer
```

Rappelez-vous qu'avec SMB, il existe des problèmes d'interprétation de la casse –un mot de passe saisi incorrectement pourrait fonctionner avec Samba mais pas avec UNIX.

Pour démarrer Samba, lançons les commandes :

```
/etc/init.d/smbd start
( /etc/rc.d/init.d/smbd start )
( /etc/init.d/samba start )
```

Pour faire bonne mesure, la configuration DNS devrait être correctement faite avec des requêtes inverses et directes sur toutes les machines clientes.

A ce stade, vous pouvez tester votre serveur Samba depuis un site UNIX. LINUX possède un support natif pour les partages SMB avec le système de fichiers `smbfs`. Essayez maintenant de monter un partage servi par la machine locale :

```
mkdir -p /mnt/smb
mount -t smbfs -o username=psheer,passwd=12345 //cericon/psheer /mnt/smb
```

Vous pouvez exécuter `tail -f /var/log/samba/cericon.log`. Cela devrait vous retourner des messages comme :

```
cericon (192.168.3.2) connect to service psheer as user psheer (uid=500, gid=500) (pid
942)
```

où le terme "service" désigne soit une partie de répertoire soit un partage d'imprimante.

L'utilitaire `smbclient` est un outil générique pour lancer des requêtes SMB, mais il est principalement utile pour les impressions. Assurez-vous que le démon d'impression est en cours d'exécution (et fonctionnel) et, ensuite lancez :

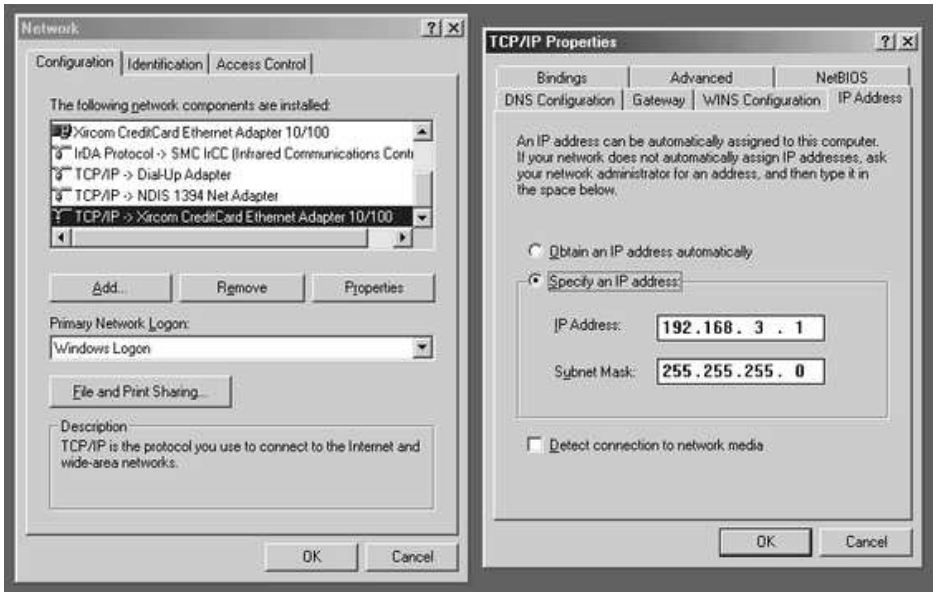
```
echo hello | smbclient //cericon/lp 12345 -U psheer -c 'print -'
```

ce qui créera une entrée sur la file d'impression. A votre fichier de journalisation s'ajoutera le commentaire :

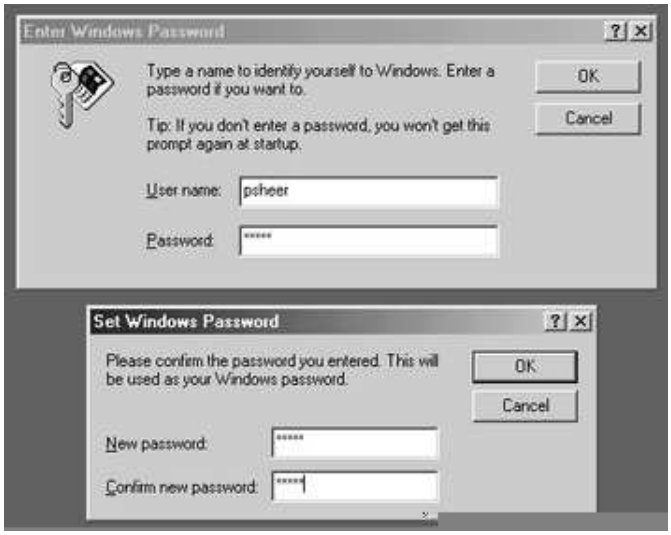
```
cericon (192.168.3.2) connect to service lp as user psheer (uid=500, gid=500) (pid 1014)
```

40.3 Configuration de Windows.

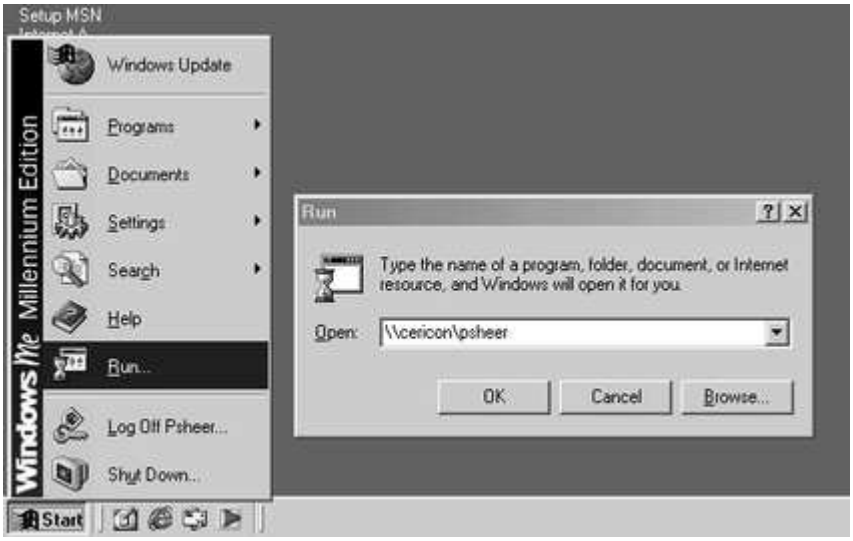
La configuration d'un poste Windows commence par la configuration de TCP/IP :



Ensuite, il faut que vous vous **déconnectiez** depuis le menu **Start** et que vous vous connectiez à nouveau en tant qu'utilisateur Samba :



Enfin, rendez-vous à l'onglet **Run...** dans le menu **Start** et entrez : `\\cericon\psheer`. Une invite pour saisir votre mot de passe vous sera proposée et vous y répondrez comme dans le cas de la commande `smbpasswd` (voir la section précédente).



Vous devriez alors accéder à votre répertoire personnel et le voir sous un jour nouveau.

40.4 Configuration d'une imprimante Windows.

Sous l'onglet **Settings** de votre menu **Start**, il vous est possible d'ajouter des imprimantes. Votre file d'imprimante UNIX `lp` est visible en tant qu'imprimante réseau `\\cericon\lp` et, elle devrait être reprise comme telle par l'assistant de configuration. Concernant le type de pilote d'imprimante, vous devriez choisir "Apple Color Laserwriter" étant donné que ce pilote s'accorde de documents PostScript. Vous ferez votre choix parmi les options du pilote d'imprimante afin d'accèder à une optimisation pour la "portabilité".

40.5 Configuration de `swat`.

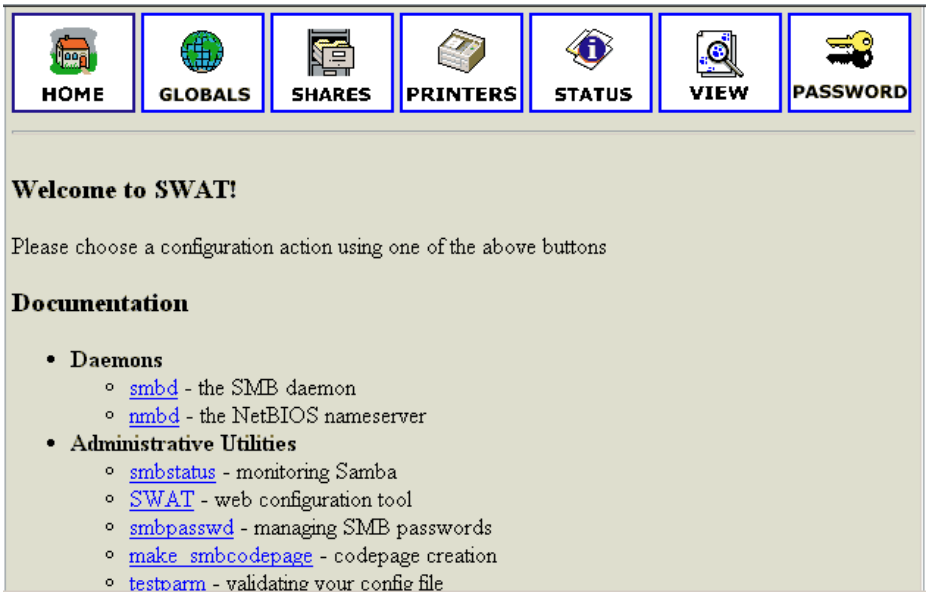
`swat` est un service, lancé depuis `inetd` et qui écoute les connexions HTTP sur le port `901`. Il permet la gestion complète à distance de Samba au moyen d'un navigateur web. Pour configurer ce service, ajoutez `swat 901/tcp` à votre fichier `/etc/services`, et la ligne qui suit, au fichier `/etc/inetd.conf` :

```
swat stream tcp nowait root /usr/sbin/tcpd /usr/sbin/swat
```

Soyez *très attentif à propos des utilisateurs que vous autorisez à se connecter.* Si vous utilisez `xinetd`, créez un fichier `/etc/xinetd.d/swat` :

```
service swat
{
   port            = 901
   socket_type     = stream
   wait            = no
   only_from       = localhost 192.168.0.0/16
   user            = root
   server          = /usr/sbin/swat
   server_args     = -s /etc/samba/smb.conf
   log_on_failure  += USERID
   disable         = no
}
```

Après avoir redémarré `inetd` (ou `xinetd`), vous pouvez pointer votre navigateur web vers http ://cericon :901/. Netscape vous demandera un nom d'utilisateur et le mot de passe correspondant. Vous devriez vous connecter en tant que `root` (`swat` n'utilise pas `smbpasswd` pour authentifier cet utilisateur). L'interface de la page web est intuitive :



Comme elle a été écrite par les développeurs mêmes de Samba, vous pouvez l'utiliser en toute confiance pour effectuer des configurations de manière routinière. La page web fournit aussi une interface d'accès à la documentation. Notez que la configuration réalisée via cette interface écrasera toute configuration déjà existante.

40.6 Windows NT – avertissement.

Les serveurs SMB de Windows sont en concurrence pour jouer le rôle de serveur de noms de leur domaine, par le numéro de version et le temps écoulé

depuis le dernier (re)démarrage de la machine (*uptime*). A nouveau, nous parlons en terme de services de noms Windows et non en terme de services DNS. Bien qu'il ne soit pas dans notre intention de considérer les mécanismes sous-jacents, nous devons garder à l'esprit que la configuration d'un serveur Samba sur un réseau constitué de nombreuses machines NT ainsi que le maintien du service peuvent constituer un véritable cauchemar. Une solution consiste parfois à éteindre toutes les machines du réseau LAN, à en sélectionner une comme serveur du domaine et la réactiver en tout premier lieu en attendant une heure. Après avoir vérifié que tout fonctionne correctement sur cette machine, procédez au réamorçage des autres ordinateurs.

Naturellement, n'oubliez pas la commande `nmblookup`.

Chapitre 41

named – serveur de noms de domaine.

Au chapitre 28, nous avons traité la partie "cliente" de DNS. Dans le présent chapitre, nous configurons un serveur de noms qui répond aux requêtes des clients.

Il semble que très souvent, le serveur de noms soit considéré comme mystérieux. En fait, la configuration et l'activation d'un serveur de noms est un exercice simple et ordinaire. Un démon serveur de noms n'est pas un processus lourd : l'exécutable `named` représente environ 500 ko et ne consomme que très peu de CPU.

Le paquet à la base d'un serveur de noms s'appelle `bind`. Ce chapitre traite de `bind-8.2` ou des versions supérieures. Le terme `bind` est l'acronyme de *Berkeley Internet Name Domain*. La difficulté lors de la mise en service d'un serveur de noms provient des fichiers de configuration qui sont pratiquement impossibles à construire *ex nihilo* sans que ne se glissent des erreurs de frappe. Donc, il ne faut jamais créer un serveur de noms par cette méthode. Il faut *toujours* partir d'un serveur de noms existant, connu pour fonctionner. Nous insisterons davantage sur les fichiers de configuration que sur le mécanisme de fonctionnement. Donc, vous pourrez vous inspirer des exemples rapportés ici, pour créer votre propre serveur de noms.

Avant de démarrer votre serveur, gardez en tête que `bind` présente des vulnérabilités. Si vous exécutez une ancienne version, votre système sera l'objet d'une attaque. Nombreux sont ceux qui sont même sceptiques quant aux versions plus récentes (`bind-9.1` au moment de la rédaction du texte anglais ; `9.2.2` au moment de la traduction) même si aucun trou de sécurité n'a été rapporté. `djbdns` constitue une alternative souvent considérée comme le server DNS le plus performant[1].

Avant que vous ne commenciez à travailler sur le fichier de configuration, vous devriez démarrer une nouvelle fenêtre de terminal avec la commande (l'alternative Debian est indiquée entre parenthèse) :

[1] NdT : au moment de la traduction, la version la plus récente était djdns-1.05.

```
tail -f /var/log/messages
( tail -f /var/log/syslog )
```

Conservez cette fenêtre ouverte durant toute l'installation et la procédure de vérification. Désormais, lorsque nous nous réferrerons aux *messages*, il s'agira de messages affiché dans cette fenêtre.

41.1 Documentation.

Les pages de man relatives à named sont hostname(7), named-xfer(8), named-(8) et ndc(8). Ces pages se rapportent à un document intitulé "Guide des opérations du serveur de noms BIND" sous forme d'un fichier Postscript intitulé /usr/[share/]doc/bind-<version>/bog/file.psf (ou /usr/share/doc/bind/bog.ps).

Le problème avec une partie de cette documentation est qu'elle est encore basée sur le fichier de configuration désormais obsolète named.boot. Il existe un script /usr/doc/bind-<version>/named-bootconf/named-bootconf (ou encore /usr/sbin/named-bootconf) qui lit named.boot depuis stdin et écrit un fichier name.conf vers stdout. Il est utile d'effectuer un echo "old config line" | named-bootconf pour voir en quoi consiste l'équivalent sous le nouveau style.

Le répertoire /usr/[share/]doc/bind[-version>]/html/ contient l'information à caractère général la plus importante. Il s'agit d'une référence complète relative à la configuration de bind. Les répertoires en parallèle contiennent aussi des documents de type FAQ et diverses discussions sur la sécurité. Un fichier nommé style.txt contient la disposition recommandée des fichiers de configuration, en termes de fiabilité et de lisibilité. Enfin, le répertoire rfc/ contient les RFCs d'intérêt (voir la section 14.6).

41.2 Configuration de bind.

Il n'y a qu'un fichier de configuration principal relatif à named, il s'agit de /etc/named.conf (ou sur Debian : /etc/bind/named.conf ; dans le présent chapitre, nous nous référons à /etc/named.conf pour faciliter la lecture). Naguère, le service named a eu recours à /etc/named.boot mais c'est désormais terminé. S'il y a un fichier named.boot dans /etc, il n'est plus utilisé, sauf peut-être par une ancienne version de bind.

Dans ce qui suit, nous donnons des exemples de configurations utiles à divers scenarii d'utilisation d'un serveur de noms.

41.2.1 Exemple de configuration

Le fichier named.conf devrait contenir la ligne : directory "/var/named" (ou directory"/etc/named" ou directory "/var/cache/bind"). Ce répertoire renferme les divers fichiers contenant les listes littérales de correspondance "nom – adresse IP" que bind aura pour mission de servir. L'exemple suivant est un serveur de noms pour une entreprise qui a reçu une gamme d'adresses IP 196.28.144.16/29 (c'est-à-dire 196.28.144.16-23) et une adresse unique

160.123.181.44. Une gamme d'adresses IP internes 192.168.2.0-255 doit également être supportée. L'astuce consiste de ne pas se préoccuper des détails de fonctionnement. Si vous copiez et éditez les fichiers de manière cohérente en lisant attentivement les commentaires, bind fonctionnera très bien. Citons à présent les fichiers nécessaires.

– ***Configuration client local*** : `/etc/resolv.conf`

```
domain localdomain
nameserver 127.0.0.1
```

– ***Fichier de configuration de haut-niveau*** : `/etc/named.conf`

```
/*
 * The "directory" line tells named that any further file name's
 * given are under the /var/named/ directory
 */
options {
        directory "/var/named";
        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below. Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
         // query-source address * port 53;
};

/* The list of root servers : */
zone "." {
         type hint;
         file "named.ca";
};

/* Forward lookups of the localhost : */
zone "localdomain" {
         type master;
         file "named.localdomain";
};

/* Reverse lookups of the localhost : */
zone "1.0.0.127.in-addr.arpa" {
         type master;
         file "named.127.0.0.1";
};

/* Forward lookups of hosts in my domain : */
```

```
zone "cranzgot.co.za" {
/* Forward lookups of hosts in my domain : */
zone "cranzgot.co.za" {
        type master;
        file "named.cranzgot.co.za";
};

/* Reverse lookups of local IP numbers : */
zone "2.168.192.in-addr.arpa" {
        type master;
        file "named.192.168.2";
};

/* Reverse lookups of 196.28.144.* Internet IP numbers : */
zone "144.28.196.in-addr.arpa" {
        type master;
        file "named.196.28.144";
};

/* Reverse lookup of 160.123.181.44 only : */
zone "44.181.123.160.in-addr.arpa" {
        type master;
        file "named.160.123.181.44";
};
```

– ***Liste de noms de serveur root*** : `/var/named/named.ca`

```
; Get the original of this file from ftp ://ftp.rs.internic.net/domain/named.root
;
; formely ns.internic.net
.                       3600000  IN  NS   a.root-servers.net.
a.root-servers.net.     3600000      A    198.41.0.4
.                       3600000      NS   b.root-servers.net.
b.root-servers.net      3600000      A    128.9.0.107
.                       3600000      NS   c.root-servers.net.
c.root-servers.net.     3600000      A    192.33.4.12
.                       3600000      NS   d.root-servers.net.
d.root-servers.net.     3600000      A    128.8.10.90
.                       3600000      NS   e.root-servers.net.
e.root-servers.net.     3600000      A    192.203.230.10
.                       3600000      NS   f.root-servers.net.
f.root-servers.net.     3600000      A    192.5.5.241
.                       3600000      NS   g.root-servers.net.
g.root-servers.net.     3600000      A    192.112.36.4
.                       3600000      NS   h.root-servers.net.
```

```
h.root-servers.net.     3600000      A     128.63.2.53
.                       3600000      NS    i.root-servers.net.
i.root-servers.net.     3600000      A     192.36.148.17
.                       3600000      NS    j.root-servers.net.
j.root-servers.net.     3600000      A     198.41.0.10
.                       3600000      NS    k.root-servers.net.
k.root-servers.net.     3600000      A     193.0.14.129
.                       3600000      NS    l.root-servers.net.
l.root-servers.net.     3600000      A     192.32.64.12
.                       3600000      NS    m.root-servers.net.
m.root-servers.net.     3600000      A     202.12.27.33
```

– ***Recherche directe locale*** : `/var/named/named/localdomain`

```
$TTL 259200
@           IN      SOA    localhost.localdomain. dns-admin.localhost.localdomain. (
                2000012101      ; Serial number
                10800           ; Refresh every 3 hours
                3600            ; Retry every hour
                3600000         ; Expire after 42 days
                259200 )        ; Minimum Time to Live (TTL) of 3 days

                IN        NS    localhost.localdomain.

localhost       IN        A     127.0.0.1
```

– ***Recherche inverse locale*** : `/var/named/named.127.0.0.1`

```
$TTL 259200
@               IN       SOA    localhost. dns-admin.localhost. (
                200012101       ; Serial number
                10800           ; Refresh every 3 hours
                3600            ; Retry every hour
                3600000         ; Expires after 42 days
                259200 )        ; Minimum Time to Live (TTL) of 3 days

                IN       NS     localhost.

                IN       PTR    localhost.
```

– ***Fichier de domaine autoritaire*** : `/var/named/named/cranzgot.co.za`

```
$TTL 259200
@               IN       SOA    nsl.cranzgot.co.za. dns-admin.nsl.cranzgot.co.za (
                2000012101      ; Serial number
                10800           ; Refresh every 3 hours
```

```
                3600            ; Refresh every hour
                3600000         ; Expire after 42 jours
                259200 )        ; Minimum Time to Live (TTL) of 3 days

                IN      NS      ns1.cranzgot.co.za.
                IN      NS      ns2.cranzgot.co.za.

                IN      A       160.123.181.44
                IN      MX      10 mail1.cranzgot.co.za.
                IN      MX      20 mail2.cranzgot.co.za.
; We will use the first IP address for the name server itself :
ns1             IN      A       196.28.144.16

; our backup name server is faaar away :
ns2             IN      A       146.143.21.88

;
 : FTP server :
ftp             IN      A       196.28.144.17

; Aliases :
www             IN      CNAME   cranzgot.co.za.
mail1           IN      CNAME   ns1.cranzgot.co.za.
mail2           IN      CNAME   ns2.cranzgot.co.za.
gopher          IN      CNAME   ftp.cranzgot.co.za.
pop             IN      CNAME   mail1.cranzgot.co.za.
proxy           IN      CNAME   ftp.cranzgot.co.za.

; Reserved for future web servers :
unused18        IN      A       196.28.144.18
unused19        IN      A       196.28.144.19
unused20        IN      A       196.28.144.20
unused21        IN      A       196.28.144.21
unused22        IN      A       196.28.144.22
unused23        IN      A       196.28.144.23

; local LAN :
pc1             IN      A       192.168.2.1
pc2             IN      A       192.168.2.2
pc3             IN      A       192.168.2.3
pc4             IN      A       192.168.2.4
; and so on... to 192.168.2.255
```

– ***Recherche inverse de LAN*** : `/var/named/named/192.168.2`

```
$TTL 259200
@               IN      SOA     ns1.cranzgot.co.za.
dns-admin.ns1.cranzgot.co.za. (
```

```
                2000012101      ; Serial number
                10800           ; Refresh every 3 hours
                3600            ; Retry every hour
                3600000         ; Expire after 42 days
                259200 )        ; Minimum Time to Live (TTL) of 3 days

                IN        NS    nsl.cranzgot.co.za.

1               IN        A     pc1.cranzgot.co.za.
2               IN        A     pc2.cranzgot.co.za.
3               IN        A     pc3.cranzgot.co.za.

4               IN        A     pc4.cranzgot.co.za.
; and so on... to 255
```

– ***Recherche inverse autoritaire(1)*** : `/var/named/named.196.28.144`

```
$TTL 259200
@               IN       SOA    nsl.cranzgot.co.za. dns-admin.nsl.cranzgot.co.za. (
                2000012101      ; Serial number
                10800           ; Refresh every 3 hours
                3600            ; Retry every hour
                3600000         ; Expire after 42 days
                259200 )        ; Minimum Time to Live (TTL) of 3 days

                IN        NS    dns.big-isp.net.

0               IN        NS    dns.big-isp.net.
1               IN        NS    dns.big-isp.net.
2               IN        NS    dns.big-isp.net.
3               IN        NS    dns.big-isp.net.
4               IN        NS    dns.big-isp.net.
5               IN        NS    dns.big-isp.net.
6               IN        NS    dns.big-isp.net.
7               IN        NS    dns.big-isp.net.
8               IN        NS    dns.big-isp.net.
9               IN        NS    dns.big-isp.net.
10              IN        NS    dns.big-isp.net.
11              IN        NS    dns.big-isp.net.
12              IN        NS    dns.big-isp.net.
13              IN        NS    dns.big-isp.net.
14              IN        NS    dns.big-isp.net.
15              IN        NS    dns.big-isp.net.

16              IN        PTR   ns1.cranzgot.co.za.
17              IN        PTR   ftp.cranzgot.co.za.
18              IN        PTR   unused18.cranzgot.co.za.
19              IN        PTR   unused19.cranzgot.co.za.
20              IN        PTR   unused20.cranzgot.co.za.
21              IN        PTR   unused21.cranzgot.co.za.
22              IN        PTR   unused22.cranzgot.co.za.
23              IN        PTR   unused23.cranzgot.co.za.

24              IN        NS    dns.big-isp.net.
25              IN        NS    dns.big-isp.net.
26              IN        NS    dns.big-isp.net.
; and so on... up to 255
```

– ***Recherche inverse autoritaire(2)*** : `/var/named/named.160.123.181.44`

```
$TTL 259200
@             IN       SOA    nsl.cranzgot.co.za. dns-admin.nsl.cranzgot.co.za. (
              2000012101     ; Serial number
              10800          ; Refresh every 3 hours
              3600           ; Retry every hour
              3600000        ; Expire after 42 days
              259200 )       ; Minimum Time to Live (TTL) of 3 days

              IN        NS    nsl1.cranzgot.co.za.
              IN        NS    nsl2.cranzgot.co.za.

              IN        PTR    cranzgot.co.za
```

41.2.2 Démarrage du serveur de noms.

Si vous avez créé une configuration analogue à celle décrite ci-dessus, vous pouvez lancer les commandes d'initialisation de `bind`. Ces commandes (avec leur alternative entre parenthèses) sont :

```
/etc/rc.d/init.d/named start
( /etc/init.d/named start )
( /etc/init.d/bind start )
/etc/rc.d/init.d/named stop
/etc/rc.d/init.d/named restart
/etc/rc.d/init.d/named status
```

Vous devriez obtenir des messages de ce type :

```
Jul  8 15 :45 :23 ns1 named[17656] : status. named 8.2.2-P5 sat Aug 5 13 :21 :24 DET 2000 ^I
Jul  8 15 :45 :23 ns1 named[17656] : hint zone "" (IN) loaded (serial 0)
Jul  8 15 :45 :23 ns1 named[17656] : master zone "localhost" (IN) loaded (serial 2000012101)
Jul  8 15 :45 :23 ns1 named[17656] : master zone "1.0.0.127.in-addr.arpa" (IN) loaded (serial
Jul  8 15 :45 :23 ns1 named[17656] : master zone "cranzgot.co.za" (IN) loaded (serial 20000121
Jul  8 15 :45 :23 ns1 named[17656] : master zone "myisp.co.za" (IN- loaded (serial 20'0012101
Jul  8 15 :45 :23 ns1 named[17656] : master zone "2.168.192.in-addr.arpa" (IN) loaded (serial
Jul  8 15 :45 :23 ns1 named[17656] : master zone "144.28.196.in-addr.arpa" (IN) loaded (serial
Jul  8 15 :45 :23 ns1 named[17656] : master zone "44.181.123.160.in-addr.arpa" (IN) loaded (se
Jul  8 15 :45 :23 ns1 named[17656] : listening on [127.0.0.1].53 (lo)
Jul  8 15 :45 :23 ns1 named[17656] : listening on [196.28.144.16].53 (eth0)
Jul  8 15 :45 :23 ns1 named[17656] : Forwarding source address is [0.0.0.0].1041
Jul  8 15 :45 :23 ns1 named : named startup succeeded
Jul  8 15 :45 :23 ns1 named[17657] : group = 25
Jul  8 15 :45 :23 ns1 named[17657] : user = named
Jul  8 15 :45 :23 ns1 named[17657] : Ready to answer queries.
```

Si vous avez commis des erreurs de typographie ou nommé des fichiers incorrec-

tement, vous obtiendrez des messages d'erreur. *Les administrateurs débutants ne modifieront pas les fichiers de configuration de named et ne redémarreront pas named sans avoir vérifié /var/log/messages (ou /var/log/syslog). N'agissez* JAMAIS *autrement.*

41.2.3 Configuration détaillée.

Si vos fichiers de configuration ne contiennent pas d'erreurs évidentes, vous devez examiner attentivement le contenu de ces fichiers.

41.2.3.1 named.conf.

Le fichier de configuration de haut-niveau /etc/named.conf a un format de type **C**. Les commentaires sont entourés par /* */ ou précédés de //.

Dans notre cas, la section intitulée options indique un seul paramètre : le répertoire de localisation des fichiers. Le fichier options.html sous les répertoires de la documentation de bind renferme une liste d'options. Certaines sont ésotériques tandis que d'autres sont d'usage courant.

Les lignes zone "." {... seront présentes dans toutes les configurations des serveurs de noms. Elles indiquent à named qu'Internet, dans son ensemble, est gouverné par le fichier named.ca. De son côté, named.ca contient une liste des serveurs de noms principaux (ou *root*).

Les lignes zone "localdomain" {... sont usuelles. Elle indiquent que les requêtes directes pour host.localdomain sont contenues dans le fichier /var/named/named.localdomain. Ce fichier donne un résultat correct dans le cas des requêtes relatives à localhost. Beaucoup d'applications recherchent ce nom auprès du serveur de noms et une configuration appropriée doit nécessairement le retourner. Notez qu'une telle recherche fonctionne toujours avec resolv.conf. Ce fichier présente une ligne search localdomain, de sorte qu'une requête pour localhost donne le même résultat qu'une requête pour localhost.localdomain.

Les lignes zone "1.0.0.127.in-addr.arpa" {... resolvent les requêtes inverses pour l'adresse IP 127.0.0.1 (enregistrée dans le fichier named.127.0.0.-1). Notez que 1.0.0.127, c'est 127.0.0.1 écrit dans l'ordre inverse. En fait, les requêtes inverses sont juste des requêtes directes sous le domaine .in-addr.arpa. De nombreuses applications effectuent une requête inverse sur les connexions reçues pour vérifier leur authenticité, même concernant localhost. Dès lors, vous souhaiterez certainement que ces lignes soient présentes afin d'empêcher ces applications de bloquer.

La suite du fichier correspond à la configuration spécifique à notre domaine.

Les lignes zone "cranzgot.co.za" {... indiquent que l'information relative aux requêtes inverses se trouve dans le fichier named.cranzgot.co.za.

Les lignes zone "1.168.192.in-addr.arpa" {... indiquent que l'information pour les requêtes inverses sur la gamme d'adresses IP 192.168.1.0-255 se trouve dans le fichier named.192.168.1.

Les lignes zone "44.182.124.160.in-addr.arpa" {... indiquent que l'information à propos des requêtes inverses sur l'adresse 160.124.182.44 se trouve dans le fichier named.160.124.182.44.

41.2.3.2 Enregistrement SOA de domaines.

Chacun des autres fichiers `named.` présente un format similaire. Ils commencent par la ligne `$TTL` suivie de l'expression `@ IN SOA`. TTL signifie *Time To Live*, il s'agit de la durée du délai d'expiration pour toutes les entrées subséquentes. Cette ligne n'empêche pas seulement l'apparition d'un message d'alerte `No default TTL set...`, mais en réalité, elle indique la durée pendant laquelle il faut mettre en *cache* une entrée. `SOA` est l'acronyme pour *Start Of Authority*.[2] Le nom d'hôte de la seconde ligne indique l'autorité pour ce domaine et le terme `<user>.<hostname>` spécifie l'adresse courriel de la personne responsable.

Les quelques lignes qui suivent dans le fichier contiennent des informations d'arrêt pour les données en *cache*, et la transmission de données sur le net. Il s'agit d'indications par défaut, raisonnables ; cependant, il vous est loisible de modifier ces valeurs. Pour cela, consultez la documentation adéquate donnée à la section 41.1. Toutes les valeurs sont exprimées en secondes.

Le *numéro de série* du fichier (c'est-à-dire `200012101`) sert à indiquer qu'un changement a été opéré et donc, que de nouvelles données devraient être transmises sur d'autres serveurs. Lorsque le fichier est mis à jour quelle que soit la méthode, vous devez incrémenter ce numéro de série. Par convention, le format est *AAAAMMJJxx* (exactement 10 chiffres). *xx* commence à `01` (disons) et est incrémenté à chaque changement effectué durant la journée.

Il est absolument essentiel que ce numéro de série soit mis à jour quand un fichier est modifié. Autrement, les changements ne seront pas répercutés sur le reste de l'Internet.

41.2.3.3 Noms d'hôtes pointés et non-pointés.

Si un nom d'hôte se termine par `.`, ce point indique un nom d'hôte pleinement qualifié. Si un nom ne se termine pas par un `.`, cela veut dire qu'un nom de domaine devra être ajouté au nom d'hôte. Cette technique n'est utilisé que pour conférer un peu d'élégance aux fichiers.

Par exemple, la ligne :

```
ftp                 IN      A       196.28.144.17
```

pourrait être écrite :

```
ftp.cranzgot.co.za. IN      A       196.28.144.17
```

Soyez toujours attentif à bien terminer les noms d'hôtes qualifiés avec un point, sinon cela conduira `named` à ajouter un domaine.

41.2.3.4 Noms d'hôtes vides.

Si un nom d'hôte est omis dans le début d'une ligne, le domaine est substitué. Cette manière d'écrire n'a d'autre but qu'apporter un peu d'élégance. Par exemple, la ligne :

[2] NdT : Il s'agit d'un enregistrement indiquant qu'un serveur DNS contient les informations « autorisées » sur un domaine particulier. En première approximation, on peut dire que ce serveur qui sait, et nul autre.

```
                    IN      NS      ns1.cranzgot.co.za.
```

est équivalente à celle-ci :

```
cranzgot.co.za.     IN      NS      ns1.cranzgot.co.za.
```

41.2.3.5 Enregistrements NS, MX, PTR et CNAME.

Chaque enregistrement DNS apparaît sur une ligne unique, en associant un domaine / de nom d'hôte ou une adresse IP avec un autre nom d'hôte ou une autre adresse IP. Donc, il est facile de construire un fichier qui renseigne l'internet à propos de votre entreprise.

Les types d'enregistrements de base sont A et PTR. Ils associent simplement un nom d'hôte à une adresse IP (A), ou un numéro IP à un nom d'hôte (PTR). Vous ne devriez pas faire correspondre plus d'un hôte à une adresse IP.

L'enregistrement CNAME indique d'un hôte qu'il est l'alias d'un autre hôte. Ainsi :

```
ns1                 IN      A       196.28.144.1
mail1               IN      CNAME   ns1.cranzgot.co.za
```

au lieu de :

```
ns1                 IN      A       196.28.144.1
mail1               IN      A       196.28.144.1
```

Finalement, les enregistrements NS et MX :

```
<domain>            IN      NS      <serveur_de_noms>
<domain>            IN      MX      <serveur_courriel>
```

déclarent que le domaine <domaine> a un serveur de nom appelé <serveur_de_noms> ou un serveur de courriel appelé <serveur_courriel>. En conséquence, les MTAs peuvent localiser leur serveur de courriel comme étant responsable d'adresses courriel de la forme *utilisateur*@cranzgot.co.za.

41.2.3.6 Configuration des recherches inverses.

Le fichier /var/named/named.196.28.144 contient les données de requêtes inverses pour les 255 adresses IP associées à 196.28.144.. Cependant, c'est notre fournisseur d'accès internet (appelé big-isp.net) qui est responsable de cette gamme d'adresses, et il est possible qu'il gère même les 65.536 adresses associées à 196.28.. Lorsqu'une requête inverse sur 196.28.144.? est tentée, celle-ci passe sur big-isp.net. Le problème est qu'il y a beaucoup d'entreprises qui utilisent la gamme 196.28.144.?, chacune avec leur propre serveur de noms, si bien qu'aucun serveur de noms simple ne peut être authoritaire pour le domaine 144.28.196.in-adrr.arpa. D'où la présence de lignes telles que celles-ci dans /var/named.196.28.144 :

```
5                       IN      NS      dns.big-isp.net.
```

L'adresse IP 196.28.144.5 n'est pas de notre responsabilité, et par conséquent nous rapportons toute requête sur ce type d'adresse à un serveur de noms plus authoritaire que le nôtre. Du côté du fournisseur d'accès internet, le serveur de noms dns.big-isp.net doit avoir un fichier /var/named/named.196.28.144 qui contient des lignes telles que :

```
$TTL 259200
@               IN      SOA     dns.dns.big-isp.net. dns-admin.dns.big-isp.net. (
                2000012101      ; Serial number
                10800           ; Refresh every 3
hours
                3600            ; Retry every hour
                3600000         ; Expire after 42 days
                259200 )        ; Minimum Time to Live (TTL) of 3 days

                IN      NS      dns.big-isp.net.

0               IN      NS      ns1.dali.co.za.
1               IN      NS      ns1.dali.co.za.
2               IN      NS      ns1.dali.co.za.
3               IN      NS      ns1.dali.co.za.
4               IN      NS      ns1.dali.co.za.
5               IN      NS      ns1.dali.co.za.
6               IN      NS      ns1.dali.co.za.
7               IN      NS      ns1.dali.co.za.
8               IN      NS      ns1.picasso.co.za.
9               IN      NS      ns1.picasso.co.za.
10              IN      NS      ns1.picasso.co.za.
11              IN      NS      ns1.picasso.co.za.
12              IN      NS      ns1.picasso.co.za.
13              IN      NS      ns1.picasso.co.za.
14              IN      NS      ns1.picasso.co.za.
15              IN      NS      ns1.picasso.co.za.

16              IN      PTR     ns1.cranzgot.co.za.
17              IN      PTR     ns1.cranzgot.co.za.
18              IN      PTR     ns1.cranzgot.co.za.
19              IN      PTR     ns1.cranzgot.co.za.
20              IN      PTR     ns1.cranzgot.co.za.
21              IN      PTR     ns1.cranzgot.co.za.
22              IN      PTR     ns1.cranzgot.co.za.
23              IN      PTR     ns1.cranzgot.co.za.

24              IN      NS      ns1.matisse.co.za.
25              IN      NS      ns1.matisse.co.za.
26              IN      NS      ns1.matisse.co.za.
27              IN      NS      ns1.matisse.co.za.
```

```
28          IN        NS    ns1.matisse.co.za.
29          IN        NS    ns1.matisse.co.za.
30          IN        NS    ns1.matisse.co.za.
31          IN        NS    ns1.matisse.co.za.
; and so on... up to 255
```

En l'occurrence, Matisse, Dali et Picasso sont les noms d'autres entreprises ayant acheté un petit bloc d'adresses IP à `big-isp.net`. Chacune des lignes permet de rediriger les requêtes au serveur de noms approprié.

41.3 Partage de charge par rotation.

Si vous avez plus d'un enregistrement `A` pour une machine particulière, `named` retournera de multiples adresses IP lors d'une requête. Le partage de charge entre différents serveurs web est donc possible –l'ordre d'apparition de ces adresses est aléatoire d'une requête à l'autre et, votre navigateur web ne fera que choisir la première adresse affichée. Par exemple, la commande `host cnn.com` retourne plusieurs adresses IP. Il pourrait s'agir de :

```
cnn.com              IN       A          207.25.71.5
cnn.com              IN       A          207.25.71.6
.
.
.
cnn.com              IN       A          207.25.71.29
cnn.com              IN       A          207.25.71.30
```

41.4 Configurer `bind` pour des connexions intermittentes.[3]

Si vous avez une connexion non-permanente (ou *dialup*), le serveur de noms devrait être configuré en tant que serveur de noms en *antémémoire seule* (ou *caching-only*).[4] Bien sûr, il n'y a pas de systèmes serveurs de noms en mémoire-tampon-*seule*, ce terme signifiant seulement que les fichiers `name.` ne contiennent que des enregistrements essentiels. L'idée d'un serveur de noms en mémoire-tampon est d'empêcher les requêtes DNS "bidons" qui consomment de la bande passante ou qui conduisent un serveur capable de faire des connexions à la demande (*dial-on-demand*, associée au paquet *diald*) à initier une connexion. Il s'agit aussi d'empêcher le blocage d'applications dans l'attente d'une requête DNS. (Des exemples typiques concernent `sendmail`, qui bloque durant plusieurs minutes quand une machine est allumée sans que le réseau ne soit branché et aussi `netscape 4`, qui essaie de faire une requête d'adresse IP sur `news.<localdomain>`.)

[3] NdT : Il s'agit de *dialup*. Ce terme décrit toute connexion non-permanente avec ou sans réseau téléphonique.

[4] NdT : ce qui pourrait aussi éventuellement s'appeler un serveur en mémoire-tampon seule.

41.4.1 Exemple d'un serveur de noms antémémoire.

Dans le cas des serveurs de noms à antémémoire, le fichier `/etc/name.conf` devrait ressembler à ce qui suit. Remplacez `<nameserver>` par l'adresse IP du serveur de noms que votre fournisseur d'accès internet vous a attribué. Le nom de votre machine locale est supposé être `cericon.prov.ate.` (le contenu est allégé des commentaires et des mises à la ligne pour des raisons de concision) :

```
options {
        forwarders {
                <nameserver>;
        };
        directory "/var/named";
};

zone "." { type hint; file "named.ca"; };
zone "localdomain" { type master; file "named.localdomain"; };
zone "1.0.0.127.in-addr.arpa" { type master; file "named.127.0.0.1"; };
zone "priv.ate" { type master; file "named.priv.ate"; };
zone "168.192.in-addr.arpa" { type master; file "named.192.168"; };
```

Le fichier `/var/named.priv.ate` devrait avoir allure suivante :

```
$TTL 259200
@           IN      SOA    cericon.priv.ate. root.cericon.priv.ate.
            ( 2000012101 10800 3600 3600000 259200 )
            IN      NS     cericon.priv.ate.
cericon     IN      A      192.168.1.1
news        IN      A      192.168.1.2
```

Le fichier `/var/named.192.168` devrait contenir ceci :

```
$TTL 259200
@           IN      SOA    localhost. root.localhost.
            ( 2000012101 10800 3600 3600000 259200 )
            IN      NS     localhost.
1.1         IN      PTR    cericon.priv.ate.
```

Les fichiers complémentaires sont identiques à ceux repris plus haut. De plus, vérifiez bien que votre nom d'hôte ait été configuré comme indiqué au chapitre 28.

41.4.2 Adresses IP dynamiques.

Dans le cas des machines à connexion intermittente, leurs adresses IP sont souvent attribuées de manière dynamique, de sorte que votre jeu d'adresses `192.168.` ne s'applique pas. Il est probable qu'il vous faudra essayer plusieurs connexions de manière à avoir une idée du type d'adresses obtenues. Supposons que votre FAI vous attribue toujours une adresse du type `196.26.`$x.x$. Vous pouvez alors effectuer une requête inverse sur `196.28.`. Ceci entraînera un échec des requêtes mais sans aucun blocage.

Une telle méthode ne sera probablement pas nécessaire. Il vaut toujours mieux identifier l'application qui provoque de fausses connexions ou qui provoque un blocage et, développer votre créativité selon les cas rencontrés.

41.5 Serveurs DNS secondaires ou esclaves.

`named` peut fonctionner comme serveur de sauvegarde pour un autre serveur, appelé serveur *esclave* ou *secondaire*.

Comme dans le cas des serveurs antémémoire *seule*, il n'y a pas –au sens strict– de serveurs *secondaires*. Il ne s'agit en fait que d'un serveur `named` fonctionnant à capacité réduite.

Supposons que nous voulions de `ns2.cranzgot.co.za` qu'il devienne le serveur secondaire de `ns1.cranzgot.co.za`. Le fichier de configuration `named.conf` sera :

```
options {
        directory "/var/named";
};

zone "." {
            type hint;
            file "named.ca";
};

zone "localdomain" {
            type master;
            file "named.localdomain";
};

zone "1.0.0.127.in-addr.arpa" {
            type master;
            file "named.127.0.0.1";
};

zone "cranzgot.co.za" {
            type slave;
            file "named.cranzgot.co.za";
            masters {
                    196.28.144.16;
            };
};

zone "2.168.192.in-addr.arpa" {
             type slave;
             file "named.192.168.2";
             masters {
                     196.28.144.16;
             };
};
```

```
zone "144.28.196.in-addr.arpa" {
              type slave;
              file "named.196.28.144";
              masters {
                      196.28.144.16;
              };
};

zone "44.181.123.160.in-addr.arpa" {
              type slave;
              file "named.160.123.181.44";
              masters {
                      196.28.144.16;
              };
};
```

Quand une entrée contient "master", vous devez fournir le fichier approprié. Quand une entrée contient "slave", named télécharge automatiquement le fichier depuis 196.28.144.12 (c'est-à-dire ns1.cranzgot.co.za), la première fois qu'une requête est nécessaire pour ce domaine.

Voilà en ce qui concerne DNS!

Chapitre 42

Protocole Point-à-Point - Réseau “dialup”.

L’exploitation du réseau téléphonique aux moyen de connexions intermittentes n’est pas fiable et entraîne des difficultés de configuration. Ceci est dû au fait que les téléphones n’ont pas été conçus pour le transfert de données (au sens informatique du terme). Cependant, étant donné que le réseau téléphonique est le plus développé des réseaux électroniques au monde, il est logique de vouloir en tirer parti. C’est pour cela que les modems ont été créés. En revanche, le RNIS n’est que légèrement plus coûteux et constitue un bien meilleur choix.[1] Voir la section 42.7 pour davantage d’informations.

42.1 Connexion de base.

Pour un usage domestique, un réseau dont les connexions sont intermittentes n’est pas difficile du tout à configurer. Le HOWTO PPP renferme de nombreuses informations (voir le chapitre 17). Les deux fichiers importants sont `/etc/ppp/chap-secrets` et `/etc/ppp/pap-secrets` qui, tous les deux contiennent la ligne :

```
<nom_utilisateur>   *   <mot_de_passe>   *
```

Seul un de ces fichiers sera réellement utilisé. Lancez la commande suivante derrière l’invite du shell [l’exemple qui suit suppose qu’une chaîne d’initialisation AT&F1est suffisante. Voir la section 4.5] :

[1] NdT : RNIS signifie Réseau Numérique à Intégration de Services. ISDN en est le terme équivalent anglais (*Integrated Services Digital Network*). Il s’agit d’un réseau permettant de transporter à la fois de la voix, des images, des données, à l’aide d’un téléphone, d’un fax, d’un PABX (Private Automatic Branch eXchange ; il s’agit d’un boîtier électronique auxquels sont reliés tous les postes téléphoniques d’une entreprise et toutes les lignes téléphoniques extérieures. Le terme français est “autocommutateur”). Le terme RNIS est opposé à RTC (Réseau Téléphonique Commuté, appelé *PSTN* en anglais pour *Public Switched Telephone Network*).

```
ppd connect \
          "chat -S -s -v \
          " 'AT&F1' \
          OK ATDT<tel-number> CONNECT " \
          name : <nom_utilisateur> assword : '\q<mot_de_passe>' \
          con : ppp" \
   /dev/<modem> 57600 debug crtscts modem lock nodetach \
   hide-password defaultroute \
   user <nom_utilisateur> \
   noauth
```

Ceci constitue une commande de connexion intermittente minimaliste qui spécifie seulement le fournisseur d'accès internet (FAI ou *ISP* en anglais). [...]

La ligne de commande est explicitée ci-dessous :

connect <script> indique le script que `pppd` doit utiliser pour le démarrage. Lorsque vous utilisez un modem de manière manuelle (comme nous le verrons ci-dessous), vous devrez parcourir les étapes d'initialisation du modem, initier une connexion, vous connecter à l'ordinateur distant et lui indiquer que vous voulez faire une connexion en mode "communication de données" appelée connexion *point-à-point* ou *PPP*. Le <script> est une automatisation de toute cette procédure manuelle.

chat -S -s -v <expect> <send> <expect> <send>... C'est la commande `<script>` en soi. `chat` possède une page de man et utilise des communications autres que par modem. `-S` impose de journaliser les messages vers le terminal et *non* vers `syslog` ; `-s` indique de journaliser vers stderr ; `-v` veut dire mode bavard. Après les options, on trouve une liste d'informations que le modem doit émettre et de réponses que fournit l'utilisateur. Il s'agit d'une séquence interactive (ou *expect-send sequence*). La séquence `AT&F1` est une chaîne pour l'initialisation du modem [l'exemple suppose qu'une chaîne d'initialisation AT&F1 est suffisante. Voir la section 4.5]. `\q` indique de ne pas imprimer le mot de passe parmi les messages de débogage (ce qui est très important).

dev/tty ? ? indique le périphérique qui doit être utilisé. Il s'agira le plus souvent de `/dev/ttyS0`, `/dev/ttyS1`, `/dev/ttyS2` ou `/dev/ttyS3`.

57600 définit la vitesse de transfert du modem. Il ne s'agit que de la vitesse de transmission PC/modem ; ceci n'a rien à voir avec le flux de données réelles. La valeur devrait être la plus élevée possible sauf dans le cas de vieilles machines dont les ports série ne manipulent pas plus que `38400`. Sauf si réellement cela ne fonctionne pas du tout, la valeur idéale est `115200`.

debug il s'agit de l'information de débogage. Cette option est utile pour diagnostiquer des problèmes.

crtscts utilise le contrôle de flux matériel.

modem utilise les lignes de contrôle du modem (c'est une option par défaut).

lock crée un fichier de verrouillage UUCP dans `/var/lock`. Comme expliqué à la section 35.4, il s'agit d'un fichier du type `/var/lock/LCK..tty??` qui indique que le périphérique série est en cours d'utilisation. C'est pour çà que vous ne *devez pas* appeler le périphérique `/dev/modem` ou `/dev/cua?`.

nodetach permet de toujours fonctionner comme processus en arrière-plan. Ceci permet de vérifier que `pppd` est en cours et de l'arrêter avec `^C`.

defaultroute crée une route IP après que PPP ait été exécuté. Donc, les paquets iront à la bonne adresse.

hide-password cache le mot de passe dans les journaux. Il s'agit d'une option très importante pour la sécurité.

user <**nom_utilisateur**> indique la ligne des fichiers `/etc/ppp/chap-secrets` et `/etc/ppp/pap-secrets` à utiliser. Pour un PC domestique, il n'y a généralement qu'une ligne.

42.1.1 Séquences interactives de votre script `chat`.

Pour déterminer la liste des séquences intéractives (*expect-send*), vous devrez établir une connexion manuellement. La commande :

```
dip -t
```

signifie *dial-IP* et converse directement avec votre modem.

La session qui suit illustre une connexion manuelle dans le cas de l'utilisateur `psheer`. L'utilisation de `dip` manuellement, comme ici, vous permet de visualiser les échanges qu'initie PPP. Lorsque vous avez obtenu ces informations, pressez `^C`. Copiez et collez votre session afin d'avoir un texte de référence.

```
[root@cericon]# dip -t
DIP : Dialup IP Protocol Driver version 3.3.7o-uri (8 Feb 96)
Written by Fred N. van Kempen, MicroWalt Corporation.

DIP> port ttyS0
DIP> speed 57600
DIP> term
[ Entering TERMINAL mode. Use CTRL-] to get back ]
AT&F1
OK
ATDT4068500
CONNECT 26400/ARQ/V34/LAPM/V42BIS
Checking authorization, please wait...
name :psheer
password :

c2-ctn-icon :ppp
Entering PPP mode.
Async interface address is unnumbered (FastEthernet0)
Your IP address is 196.34.157.148. MTU is 1500 bytes

~y}#A!}!e} }3}"}&} }*} } }~}&4}2Iq}'}"}(}"NS~~y}#A}!r} }4}"}&} }
[ BACK to LOCAL mode. ]
DIP > quit
[root@cericon]
```

A présent, vous pouvez modifier le script `chat` ci-dessus à votre guise. Les changements sont élémentaires. Par exemple, vous pouvez indiquer `login :` au lieu de `name :`. Certains systèmes vous demanderont aussi de saisir d'autres informations que `ppp`, tandis que d'autres ne vous demanderont plus rien après le mot de passe. Il arrive que certains systèmes ne requièrent pas de données après être entrés en mode PPP.

Notez que `dip` crée aussi des fichiers de verrouillage UUCP comme c'est expliqué à la section 35.4.

42.1.2 CHAP et PAP.

Vous pourriez vous demander la raison d'être des fichiers `/etc/ppp/chap-secrets` et `/etc/ppp/pap-secrets` si un nom d'utilisateur et un mot de passe sont déjà indiqués dans le script `chat`. *CHAP* (*Challenge Handshake Authentication Password*) et *PAP* (*Password Authentication Protocol*) sont des mécanismes d'authentification utilisés *après* la connexion, quelque part dans la séquence :

```
~y}#A!}!e} }3}"}&} }*} } }~}&4}2Iq}'}"}(}"NS~~y}#A}!r} }4}"}&} }
```

42.1.3 Exécution de `pppd`.

Si vous lancez la commande `pppd`, vous obtiendrez un retour comme celui-ci :

```
send (AT&F1 ^M)
expect (OK)
AT&F1^M^M
OK
 -- got it

send (ATDT40685OO^M)
expect (CONNECT)
^M
ATDT40685OO^M^M
CONNECT
 --got it
send(^M)
expect (name)
^M
ATDT40685OO^M^M
CONNECT
 --got it

send(^M)

expect (name)

 45333/ARQ/V90/LAPM/V42BIS^M

Checking authorization, Please wait...^M

username :

 -- got it


send (psheer^M)

expect (assword :)

psheer^M

password :

 -- got it
```

```
send (??????)
expect (con :)
^M
^M
c2-ctn-icon :
 -- got it

send (ppp^M)
Serial connection established.
Using interface ppp0
Connect : ppp0 <--> /dev/ttyS0
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x88c5a54f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x3d <asyncmap 0x0a0000> <magic 0x3435476c> <pcomp> <accomp>]
sent [LCP ConfAck id=0x3d <asyncmap 0xa0000> <magic 0x3435476c> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x88c5a54f> <pcomp> <accomp>]
sent [IPCP ConfReq id=0x1 <addr 192.168.3.9> <compress VJ 0f 01>]
sent [CCP ConfReq id=0x1 <deflate 15> <deflate(old#) 15> <bsd v1 15>]
rcvd [IPCP ConfReq id=0x45 <addr 168.209.2.67>]
sent [IPCP ConfAck id=0x45 <addr 168.209.2.67>]
rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfRej id=0x2 <addr 192.168.3.9>]
rcvd [IPCP ProtRej od=0x3e 80 fd 01 01 00 0f 1a 04 78 00 18 04 78 00 15 03 2f]
rcvd [IPCP ConfNak id=0x2 <addr 196.34.157.131>]
sent [IPCP ConfReq id=0x3 <addr 196.34.157.131>]
rcvd [IPCP ConfAck id=0x3 <addr 196.34.157.131>]
local IP address 196.34.25.95
remote IP addres 168.209.2.67
Script /etc/ppp/ip-up started (pid 671)
Script /etc/ppp/ip-up finished (pid 671), status =0x0
 Terminating on signal 2.
Script /etc/ppp/ip-down started (pid 701)
sent [LCP TermRep id=0x2 "User request"]
rcvd [LCP TermAck id=0x2]
```

Vous pouvez "voir" fonctionner les séquences interactives, si bien qu'il est aisé de les corriger si vous faites des erreurs quelque part.
A ce stade, vous voudrez peut-être obtenir le résultat de `route -n` et `ifconfig` dans un autre terminal :

```
[root@cericon]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
168.209.2.67    0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         168.209.2.69    0.0.0.0         UG    0      0        0 ppp0
[root@cericon]# ifconfig
lo      Link encap :Local Loopback
        inet addr :127.0.0.1 Mask :255.0.0.0
        UP LOOPBACK RUNNING MTU :3924 Metric :1
```

```
        RX packets :2547933 errors :0 dropped :0 overruns :0 frame :0
        TX packets :2547933 errors :0 dropped :0 overruns :0 carrier :0
        collisions :0 txqueuelen :0

ppp0    Link encap :Point-to-Point Protocol
        inet addr :196.34.25.95 P-t-P :168.209.2.67 Mask :255.255.255.255
        UP POINTTOPOINT RUNNING NOARP MULTICAST MTU :1500 Metric :1
        RX packets :7 errors :0 dropped :0 overruns :0 frame :0
        TX packets :7 errors :0 dropped :0 overruns :0 carrier :0
        collisions :0 txqueuelen :10
```

Ceci montre clairement que `pppd` a créé un périphérique réseau et une route allant vers ce dernier.

Si votre serveur de noms est configuré, vous deviez être à même de réussir un `ping metalab.unc.edu` ou un `ping` vers tout autre hôte bien connu.

42.2 Connexion à la demande ; mascarade d'adresses.

Les connexions à la demande (ou *dial-on demand*) requièrent essentiellement d'ajouter l'option `demand` à la ligne de commande `pppd` vue ci-dessus. L'autre manière de procéder consiste à utiliser le paquet logiciel `diald`, mais en l'occurrence nous ne discuterons que de `pppd`. Cependant, le paquet `diald` est de loin une solution plus générale et complète.

Avec l'option `demand`, vous noterez que de fausses connexions se produisent. Vous devez donc ajouter des règles de filtrages pour vous assurer que seuls les services qui vous intéressent produisent une connexion sortante. Ces services ne feront alors de connexions sortantes que lorsque cela s'avère nécessaire.

Nous allons donc voir comment établir des règles de filtrage ou de pare-feu (*firewall* en anglais). Notre application, basée sur `iptables`, est décrite à la sous-section 42.2.2.5, après une brève présentation des pare-feux basés sur les commandes anciennes : `ipfwadm` et `ipchains`.

Notre application se rapporte à un système représenté à la figure 42.1, où la machine connectée à l'internet par un modem agit comme passerelle (*gateway*) pour un réseau d'entreprise de classe C (`192.168.0.0/24`). Notez qu'il s'agit d'un cas très simple.

Notre cahier de charges pour l'établissement des règles est le suivant : nous voulons que les machines du réseau LAN puissent communiquer entre elles et avec la passerelle, que la passerelle puisse communiquer avec les machines du LAN, que les machines du LAN puissent avoir accès à l'internet et que les services DNS, SMTP et SSH soient disponibles sur la passerelle depuis internet.

42.2.1 D'ipfwadm à ipchains.

42.2.1.1 Mécanisme d'ipfwadm.

Toutes les fonctionnalités des pare-feux filtrants sont mises en oeuvre dans le noyau Linux. Jusqu'à la version `2.1.102`, il s'agit d'`ipfwadm`. A partir de

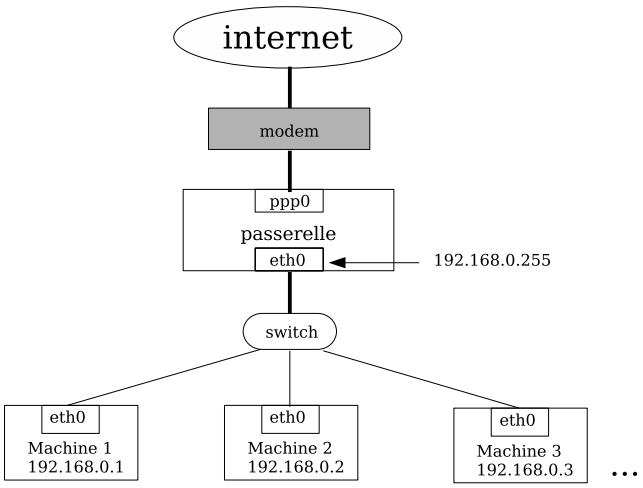


FIG. 42.1 – Schéma d'un réseau domestique (ou d'une petite entreprise) de classe C (192.168.0.0/24) avec une passerelle connectée à l'internet par un modem (via ppp0). Le modem peut être de type "câble" ou ADSL.

la version 2.1.102, ipchains est utilisé (éventuellement avec une compatibilité ipfwadm). Depuis le noyau 2.4, nous disposons en plus d'iptables (avec éventuellement une compatibilité avec ipchains et ipfwadm). Le lecteur se référera au Firewall-HOWTO pour davantage d'information sur l'élaboration d'un pare-feu. Lorsque le filtrage se fait sans suivi de connexion, le pare-feu est dit "stateless" ; s'il y a suivi de connexions (*conntrack*), le pare-feu est "statefull". Ces pare-feu travaillent au niveau de la couche de transport en analysant les *sockets* c'est-à-dire les couples d'adresses IP ↔ ports.

Si, par ailleurs, un filtrage applicatif est souhaité, il est nécessaire d'installer un serveur *proxy* (ou mandataire) qui permet de faire des requêtes au nom d'une machine se trouvant derrière le pare-feu (squid en est un exemple).[2] N'allez cependant pas croire qu'un réseau protégé par un indispensable pare-feu est devenu inviolable (voir le chapitre 45).

Comme indiqué dans la figure 42.2, le principe d'ipfwadm (un paquet réécrit à partir du logiciel ipfw de BSD) est le suivant :

- lorsqu'un paquet entre, il est analysé par les règles d'input,
- soit il est rejeté,
- soit il est accepté et alors :
  - il est routé vers les règles de forward s'il est destiné à une autre machine,
  - il est toujours analysé par les règles d'output quoiqu'il advienne,
  - il est émis à la machine destinataire.

42.2.1.2 Mécanisme d'ipchains.

ipchains est un module du noyau (voir le chapitre 43) qui effectue un filtrage des paquets IP. Il s'agit d'un remaniement d'IPv4, un paquet largement inspiré de BSD. Comme cela est schématisée dans la figure 42.3, ipchains réalise le filtrage selon trois chaînes : sur les paquets entrants (input), sur les paquets

[2] Le lecteur peut se référer au Squid-HOWTO à l'adresse http ://www.squid-cache.org.

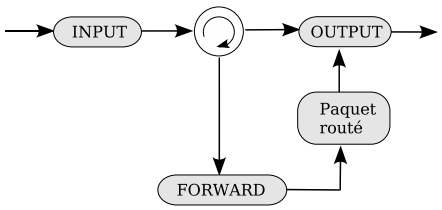


FIG. 42.2 – Schéma de fonctionnement d'ipfwadm.

sortants (output) et sur les paquets transmis (forward). Pour chacune des trois chaînes, il existe trois politiques : acceptation du paquet (ACCEPT), rejet du paquet en prévenant la source du paquet que ce dernier est rejeté (REJECT)[3] et élimination du paquet (DENY).

42.2.2 Netfilter avec Iptables.

42.2.2.1 Option du noyau.

Avant de voir les propriétés d'iptables, nous allons configurer le noyau (il doit s'agir au minimum d'un noyau linux-2.4) pour les utiliser. Les options suivantes –présentées comme dans le fichier /usr/src/linux/.config– doivent êtres activées en dur ou sous forme de modules (voir le chapitre 43 pour la compilation du noyau et l'installation de modules) :

```
CONFIG_PACKET
CONFIG_NETFILTER

CONFIG_IP_NF_CONNTRACK
CONFIG_IP_NF_FTP
CONFIG_IP_NF_IRC
CONFIG_IP_NF_IPTABLES
CONFIG_IP_NF_NAT
CONFIG_IP_NF_MATCH_STATE
CONFIG_IP_NF_TARGET_LOG
CONFIG_IP_NF_MATCH_LIMIT
CONFIG_IP_NF_TARGET_MASQUERADE
```

Si vous souhaitez assurer la compatibilité avec ipfwadm, ipchains, permettre l'installation de proxies et la correspondance avec les adresses MAC des cartes Ethernet, vous devriez activer les options suivantes, respectivement :

```
CONFIG_IP_NF_COMPAT_IPFWADM
CONFIG_IP_NF_COMPAT_IPCHAINS
CONFIG_IP_NF_TARGET_REDIRECT
CONFIG_IP_NF_MATCH_MAC
```

Cette étape précède la compilation du noyau (voir la section 43.11 pour les

[3] Il est conseillé de toujours utiliser DENY de manière à ne pas alerter l'expéditeur qui pourrait être un pirate.

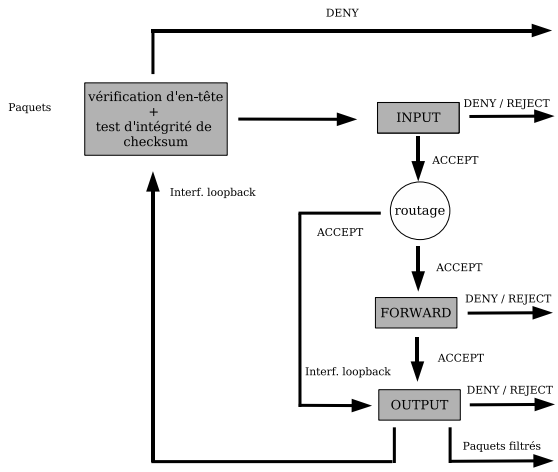


FIG. 42.3 – Principe simplifié de fonctionnement d'ipchains.

noyaux 2.4 et aussi, complémentairement la section 19.8.2 pour les noyaux 2.6).

42.2.2.2 Mécanisme d'iptables.

Même si la syntaxe d'iptables (voir iptables(8)) peut sembler proche de celle d'ipchains, l'architecture et le fonctionnement d'iptables s'éloignent assez de ceux d'ipchains. Ainsi, les chaînes INPUT et OUTPUT ne contrôlent pas tout ce qui entre et sort de la passerelle (y compris concernant le routage). Le contrôle ne concerne que ce qui entre en direction de la passerelle et ce qui sort de cette dernière.[4] Comme indiqué dans la figure 42.4, Netfilter présente cinq points d'accrochage (ou *hooks*) sur lesquels des modules de traitements de paquets se greffent (NF_IP_PRE_ROUTING, NF_IP_LOCAL_IN, NF_IP_LOCAL_OUT, NF_IP_FORWARD et NF_IP_POSTROUTING). La branche de gauche illustre le trajet des paquets entrant et sortant, vers et depuis, un processus fonctionnant de manière locale (FTP, HTTP, ...). La branche de droite représente le trajet des paquets traversant la passerelle dans sa fonction de routeur.

42.2.2.3 Tables.

Grâce aux cinq points d'accrochage, Netfilter est capable :
- d'effectuer des filtrages de paquets (fonction de pare-feu) de manière, par exemple, à interdire tous les paquets venant d'internet mais à autoriser les paquets s'adressant au port 80 (HTTP),
- d'effectuer de la traduction d'adresses réseau ou NAT (*Network Address Translation*), ce qui est utile quand on veut faire communiquer avec l'internet un réseau privé présentant des adresses du type 192.168.x.y.,
- de marquer des paquets de manière à leur appliquer un traitement spécial.

[4] Pour compléter l'information présentée ici, le lecteur devrait consulter le projet Netfilter/Iptables à l'adresse : http ://www.netfilter.org.

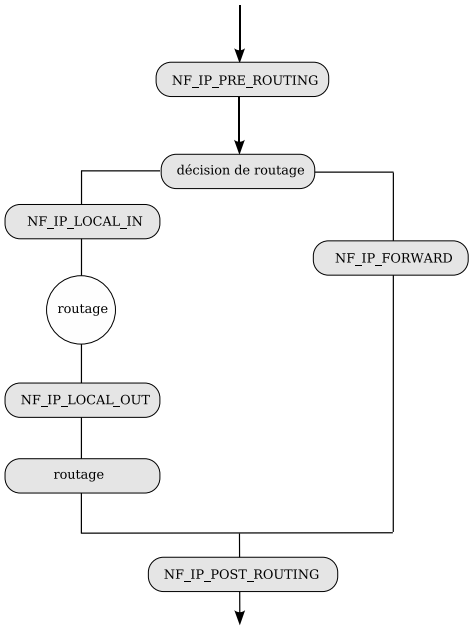


FIG. 42.4 – Principe simplifié de fonctionnement de `Netfilter / iptables`.

Netfilter peut agir ainsi grâce à la commande `iptables` qui écrit dans trois tables : `FILTER`, `NAT` et `MANGLE`.

La table `FILTER` contient les règles de filtrage des paquets; il y a trois chaînes :

- `INPUT` qui décide du sort des paquets entrant en local sur l'hôte,
- `OUTPUT` qui traite les paquets sortants,
- `FORWARD` qui filtre les paquets allant d'une interface à une autre, selon les routes existantes.

La table `NAT` effectue toutes les traductions d'adresses nécessaires :

- `PREROUTING` est une chaîne de traduction d'adresses IP (*IP translation*),
- `POSTROUTING` permet la traduction de l'adresse de la source,
- `OUTPUT` permet de modifier la destination des paquets produits localement (c'est-à-dire par la passerelle).

La table `MANGLE` opère des marquages de paquets entrants (`PREROUTING`) et produits localement (`OUTPUT`). Le marquage de paquets dans leur en-tête sert à autoriser un traitement spécial des paquets dans la table de routage.

Deux cas peuvent se présenter sur un pare-feu. Quelle que soit l'interface sur laquelle arrive un message, ce dernier passe par la fonction de décision de routage. Celle-ci détermine si le paquet est destiné à un processus local à l'hôte ou à une machine d'un autre réseau :

- soit le paquet est destiné à l'hôte local, alors il traverse la chaîne `INPUT`,
- soit le paquet est destiné à un autre hôte : il traverse la chaîne `FORWARD`.

Dans le premier cas de figure, si le paquet n'est pas rejeté, il est transmis au processus auquel il s'adresse. Le processus le traite et émet, le cas échéant, un nouveau paquet en réponse. Ce nouveau paquet traverse la chaîne `OUTPUT`. S'il n'est pas rejeté, il est dirigé vers la sortie.

Dans le second cas de figure, si le paquet n'est pas rejeté, il poursuit sa route.

42.2.2.4 Syntaxe.

Il est bien entendu nécessaire de visiter `iptables`(8), mais nous allons voir quelques éléments essentiels.

Les comportements adressés à l'endroit d'un paquet sont :
- `ACCEPT` : paquet autorisé,
- `REJECT` : le paquet est rejeté et il y a envoi d'un message à l'expéditeur,
- `DROP` : le paquet est ignoré,
- `LOG` : le paquet est journalisé.

Les principales options de manipulation des chaînes sont :
- `A` : ajoute une règle à une chaîne,
- `D` : supprime une règle à une chaîne,
- `F` : supprime les règles une à une (*flush*),
- `I` : insertion d'une règle à une position donnée dans une chaîne,
- `L` : affichage des règles,
- `N` : création d'une nouvelle chaîne par un utilisateur,
- `P` : politique par défaut à l'endroit d'une chaîne,
- `R` : remplacement d'une chaîne donnée par une autre,
- `X` : suppression d'une chaîne vide.

Les paramètres principaux sont :
- `c` : permet d'initialiser le compteur de paquets et d'octets d'une règle,
- `d` : spécifie la destination (voir le paramètre `-s`),
- `i` : le sémaphore `-i` désigne l'interface d'entrée et n'est fonctionnel que dans les chaînes `INPUT` et `OUTPUT`,
- `j` : le sémaphore `-j` signifie : "que faut-il faire du paquet ?". Il est typiquement suivi des termes `ACCEPT`, `DROP` ou `REJECT`,
- `o` : le sémaphore `-o` désigne l'interface de sortie et n'est fonctionnel que dans les chaînes `FORWARD` et `OUTPUT`.
- `p` : il s'agit du protocole (`tcp`, `udp`, `icmp` ou `all`),
- `s` : indique la source (nom de réseau, nom d'hôte, adresse d'un réseau ou adresse IP),
- `[!]` : un paquet auquel ce sémaphore s'applique ne correspondra à aucune des règles spécifiées.

Notez encore que :
- les ports TCP et UDP doivent être mentionnés avec les options `--source/--port/--sport` (ou `--destination-port/--dport`). Ils doivent être placés après les options `-p tcp` ou `-p udp`. Ceci permet de charger les extensions TCP et UDP, respectivement. Peut suivre : (i) le numéro du port (`iptables -A INPUT -p tcp --dport 110 -j DENY`), (ii) une série de ports décrits un à un (`iptables -A INPUT -p tcp -m multiport -dport 110,4242,119 -j DROP`) ou (iii) une série de ports sous forme d'une plage (`iptables -A INPUT -p tcp -sport 4925 :4633 -j ACCEPT`),
- grâce à `--tcp-flags`, la possibilité existe de spécifier une correspondance sur un sémaphore `tcp` (`SYN`, `ACK`, `FIN`, `RST`, `URG`, `PSH`, `ALL`, `NONE`) comme dans `iptables -A INPUT -p tcp --dport 42 --tcp-flags SYN,ACK -j ACCEPT`,
- grâce à `--icmp-type`, on peut indiquer une action à conduire sur un paquet `icmp` (`8` pour `ping`) : par exemple, `iptables -A INPUT -p icmp`

`--icmp-type 8 -j ACCEPT`,
– grâce à `--mac-address`, il est possible d'établir une règle pour une adresse MAC comme dans `iptables -A INPUT --mac-source 42.42.AA.42.AA -j DROP`.

42.2.2.5 Application à notre pare-feu.

Le fichier dans lequel les modifications de votre pare-feu Netfilter ont lieu s'appelle `/etc/sysconfig/iptables` (sous RedHat ou Debian) ou `/etc/conf.d/local.start` (sous Gentoo). Lorsque vous avez modifiez vos règles, n'oubliez pas de les sauvegardez et d'activer le pare-feu avec les commandes suivantes :

```
/etc/init.d/iptables save
/etc/init.d/iptables restart
```

La première opération consiste à effectuer de la redirection (ou *forwarding*) d'adresses :

```
# Permettre le "forwarding" et le changement dynamique d'adresses :
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv4/ip_dynaddr
```

Ensuite, nous nettoyons toutes les tables et fermons totalement le pare-feu. A l'issue de cette manipulation, si vous utilisez la commande `ping`, rien ne devrait se passer entre la passerelle et le réseau LAN ni entre les postes du réseau LAN et la passerelle. Les `ping`s de la passerelle vers l'internet (et vice-versa) ne fonctionneront pas. En outre, les `ping`s des hôtes du LAN vers l'internet ne répondront pas (d'une part, il n'y a pas de règle de `NAT` entre le réseau local et l'internet ; d'autre part, le `FORWARD` effectue un `DROP` sur tous les paquets).

```
# On vide les règles :
iptables -F
# Suppression des règles une à une :
iptables -X

# On fait pointer les règles sur DROP :
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# On vide des règles pour "nat" et "mangle" :
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F

iptables -t mangle -X
```

A présent, nous permettons à la passerelle de s'auto-`ping`uer sur 127.0.0.1. Nous autorisons aussi que, de la passerelle, un `ping` soit possible vers les hôtes du réseau privé et inversément. A ce stade, rien d'autre n'est permis (notez qu'*il n'est*

pas forcément vrai que le réseau local soit sûr).

```
# la passerelle peut émettre dans tous les sens :
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# le réseau est supposé sûr :
iptables -A INPUT -i eth0 -j ACCEPT
iptables -A OUTPUT -o eth0 -j ACCEPT
```

Autorisons le NAT, à présent, en effectuant une traduction d'adresses pour tout paquet qui traverse la passerelle en sortant via ppp0 :[5]

```
# établissement du NAT :
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Pour l'instant, les pings du réseau local vers l'internet ne sont pas fonctionnels car FORWARD effectue toujours un DROP sur tous les paquets. Nous allons donc accorder des autorisations sur le FORWARD. Pour réaliser cette opération, nous utilisons le mécanisme de conntrack (ou suivi de connexions). Quoiqu'un peu consommateur de ressources sur la passerelle, le suivi de connexion permet d'obtenir des informations sur toute connexion en cours :

NEW paquet demandant une nouvelle connexion,

ESTABLISHED paquet associé à une connexion déjà établie,

RELATED nouvelle connexion liée à une connexion établie (par exemple, dans le cas d'une connexion FTP active),

INVALID paquet associé à une connexion inconnue.

Il est possible de n'accepter –et ce dans les deux sens (depuis et vers l'internet)– que les connexions déjà établies ou en relation avec des connexions déjà établies et, de n'accepter de nouvelles connexions que depuis votre réseau vers l'internet. Avec cette manière de procéder, votre réseau local peut se connecter sur tout serveur de l'internet, mais aucune nouvelle connexion ne peut être établie depuis l'internet vers votre réseau.

Nous allons donc faire en sorte que toutes les connexions nouvelles, établies et associées à une connexion déjà établie qui entrent par eth0 (réseau local) et qui veulent sortir par ppp0 (vers l'internet) puissent passer. Nous allons aussi permettre que toutes les connexions établies ou associées à une connexion déjà établie qui entrent par ppp0 et veulent aller vers eth0 puissent passer.

[5] Notez que le NAT peut être restreint à une plage d'adresses IP de votre réseau LAN :
- `iptables -t nat -A POSTROUTING -s 192.168.0.0/255.255.255.0 -o ppp0 -j MASQUERADE`

ou à une liste d'adresses définies :
- `iptables -t nat -A POSTROUTING -s 192.168.0.10 -o ppp0 -j MASQUERADE`
- `iptables -t nat -A POSTROUTING -s 192.168.0.11 -o ppp0 -j MASQUERADE`

```
# toute connexion sortant du LAN vers internet= acceptée :
iptables -A FORWARD -i eth0 -o ppp0 -m state --state NEW,
ESTALISHED,RELATED -j ACCCEPT

# seules les connexions établies ou en relation = acceptées
(d'internet vers LAN) :
iptables -A FORWARD -i ppp0 -o eth0 -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

Remarquez donc, l'utilisation de l'option `--state`. A présent, si vous essayez la commande `ping -c 5 www.yahoo.fr`, vous devriez obtenir une réponse.

En résumé, à ce stade, les machines du réseau local accèdent à la passerelle. De la passerelle, vous accéderz au réseau LAN. De l'internet, nul n'accède à la passerelle (`DROP` en `INPUT` sur `ppp0`). En outre, depuis l'internet, il n'est possible d'accéder aux hôtes du LAN que via des connexions déjà établies et dont ils sont clients. Autrement dit, aucun serveur du réseau local n'est accessible depuis l'internet.

Lecture du suivi des connexions.

Avant d'ouvrir des ports, nous pouvons déjà apprendre comment décoder les messages de suivi de connexions (*conntrack*). Le fichier-clé est `/proc/ip_conntrack`. Il contient des informations analogues à celles-ci :

```
tcp 6 14 CLOSE_WAIT
         src=192.168.0.10 dst=213.186.35.33 sport=1102 dport=80
         src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1102 [ASSURED] use=1

tcp 6 431991 ESTABLISHED
         src=192.168.0.10 dst=213.186.35.33 sport=1103 dport=80
         src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1103 [ASSURED] use=1

tcp 6 73 TIME_WAIT
         src=192.168.0.10 dst=213.186.35.33 sport=1104 dport=80
         src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1104 [ASSURED] use=1

tcp 6 82 SYN_SENT
         src=192.168.0.10 dst=213.186.35.33 sport=1105 dport=80 [UNREPLIED]
         src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1105 use=1

tcp 6 51 CLOSE_WAIT
         src=192.168.0.10 dst=213.186.35.33 sport=1106 dport=80
         src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1106 [ASSURED] use=1
```

Pour comprendre ces informations, il faut savoir qu'une des machines du LAN possède l'adresse IP `192.168.0.10` et que la passerelle a pour adresse IP sur le LAN : `192.168.0.255`. L'adresse IP de la passerelle sur l'internet est `80.8.193.-97`. Le client navigue sur un serveur HTTP (`sport=80`) dont l'adresse IP est `213.186.35.33`. Le troisième champ de chaque nouvelle entrée est un "timer". L'entrée est effacée de la table `/proc/ip_conntrack` lorsque le nombre atteint zéro. Cette table étant en mémoire (sur le fichier virtuel `/proc`), son contenu évo-

lue constamment. Par ailleurs, sur un réseau domestique, l'occupation de la mémoire sera négligeable. Il en va autrement de l'utilisation du *suivi de connexions* dans les réseaux de grandes entreprises.

Nous pouvons maintenant reprendre la construction du pare-feu afin d'y ouvrir des ports sélectivement.[6]

Supposons que vous souhaitiez installer un serveur DNS sur votre passerelle, un SMTP (pour envoyer du courrier depuis le LAN sans se préoccuper des disponibilités du SMTP du FAI) et un accès SSH afin d'administrer à distante votre serveur depuis l'internet. Que devez-vous ajouter comme règles ?

S'il faut que votre serveur réponde aux requêtes DNS, il est nécessaire d'ouvrir une voie UDP (voir `/etc/services`). Votre serveur envoie une requête UDP à destination du port 53 d'un serveur DNS et attend une réponse sur un port supérieur à 1025 (en provenance de 53).

```
# Autoriser des requêtes DNS locales :
iptables -A OUTPUT -o ppp0 -p udp --sport 1024 : --dport 53 -m state --state \
                                                ! INVALID -j ACCEPT iptables -A
INPUT -i ppp0 -p udp --sport 53 --dport 1024 : -m state --state \
                                                 RELATED,ESTABLISHED -j ACCEPT
```

Notez l'utilisation de `!` qui ressemble à l'opérateur logique `NON` (qui veut dire en l'occurrence : sauf) et de `dport` signifiant port de destination.

A présent, concernant le serveur SMTP, nous savons qu'il écoute sur le port 25 et utilise TCP :

```
# Autoriser des envois SMTP locaux :
iptables -A OUTPUT -o ppp0 -p tcp --sport 1024 : --dport 25 -m state --state \
                                                ! INVALID -j ACCEPT iptables -A
INPUT -i ppp0 -p tcp --sport 25 --dport 1024 : -m state --state \
                                                RELATED,ESTABLISHED -j ACCEPT
```

Enfin, pour que vous puissiez accéder à votre passerelle depuis l'internet (en supposant que vous puissiez relever son adresse IP à tout instant), il faut garder à l'esprit qu'SSH écoute le port 22 et utilise TCP :

```
# Autoriser l'accès SSH depuis l'internet :
iptables -A INPUT -p tcp --dport ssh -i ppp0 -j ACCEPT
iptables -A OUTPUT -p tcp --sport ssh -o ppp0 -j ACCEPT
```

Sauvegardez vos règles :

```
/etc/init.d/iptables save
```

et activez votre pare-feu :

[6] Bien qu'il ne soit pas nécessaire, le service `auth` peut être ouvert de manière que les services se connectant reçoivent un message d'échec au lieu d'attendre le délai d'interruption. Cependant, pour des raisons de sécurité, vous devriez commentez la ligne `auth` dans `/etc/inetd.conf`.

```
/etc/init.d/iptables start
```

Ensuite, vérifiez que les ports devant rester ouverts le sont bel et bien.

Pour verrouiller le parefeu de sorte à ce que plus rien ne passe :

```
/etc/init.d/iptables panic
```

42.3 Revenons à pppd.

Le script `pppd` devient (notez que vous devez avoir une version supérieure à `pppd-2.3.11` pour que cela fonctionne :)[7]

```
ppd connect \
          "chat -S -s -v \
          " 'AT&F1' \
          OK ATDT<tel-number> CONNECT " \
          name : <nom_utilisateur> assword : '\q<mot_de_passe>' \
          con : ppp" \
   /dev/ttyS0 57600 debug crtscts modem lock nodetach \
   hide-password defaultroute \
   user <nom_utilisateur> \
   demand \
    :10.112.112.112 \
   idle 180 \
   holdoff 30
```

42.4 Connexions DNS.

Pour être utilisé sur un serveur à connexions intermittences (*dialup*), votre service DNS doit être personnalisé. Remplacez la section "hilippe féretoption" de la configuration DNS telle que décrite dans le chapitre 41, avec ce qui suit :

```
options {
        forwarders { 196.7.173.2 ; /* exemple only */ } ;
        listen-on { 192.168.2.254 ; } ;
        directory "/var/cache/bind" ;
        dialup yes ; notify no ; forward only ;
} ;
```

L'option `dialup yes ; notify no ; forward only ;` indique dans l'ordre à `bind` (i) d'utiliser la ligne aussi peu que possible, (ii) de ne pas envoyer de messages d'annonce (il n'y a pas de serveur esclave sur notre LAN pour faire des annonces) et (iii) de rediriger les requêtes vers 192.168.2.254 plutôt que d'essayer de leur répondre. L'option `listen-on` force le serveur de noms à se connecter à l'interface réseau `192.168.2.254`, exclusivement. Dans le présent exemple, l'interface `192.168.2.254` est notre carte Ethernet qui route les paquets depuis

[7] NdT : au moment de la traduction, la version la plus récente était `ppp-2.4.x`.

le réseau LAN. Ceci est important pour la sécurité, parce qu'on empêche ainsi toute connexion entrante tentée depuis l'extérieur du réseau.

Il existe aussi un paquet DNS écrit spécifiquement pour utiliser des serveurs à connexions intermittentes. Ce paquet, appelé `dnrd`, est beaucoup plus facile à configurer que `bind`.

42.5 Serveur d'appels entrants.

`pppd` constitue une manière d'initialiser un périphérique réseau sur un port série, sans se préoccuper du fait que vous initiez ou écoutez une connexion. Du moment qu'il y a une connexion série entre deux machines, `pppd` négocie une liaison.

Pour *écouter* une connexion *entrante* avec `pppd`, il suffit d'ajouter la ligne suivante au fichier `/etc/inittab` :

```
S0 :2345 :respawn :/sbin/mgetty -s 115200 ttyS0
```

et ensuite la ligne :

```
/AutoPPP/ - a_ppp /usr/sbin/pppd
```

au fichier `/etc/mgetty+sendfax/login.config` (ou sur Debian : `/etc/mgetty/-login.config`). Pour des raisons de sécurité, vous souhaiterez probablement exécuter un `chmod a-s /usr/sbin/pppd`, étant donné que `mgetty` exécute `pppd` en tant que `root`. Votre fichier `/etc/ppp/options` contiendra :

```
proxyarp mtu 552 require-chap <nom_hôte>
```

Notez que nous nous passons des options relatives à la ligne série (c'est-à-dire le contrôle et la vitesse de flux) parce que `mgetty` a déjà initialisé la ligne série. `<nom_hôte>` représente le nom de la machine locale. La configuration de proxyarp ajoute le client distant aux tables ARP.[8] Ceci permet à votre client de se connecter à l'internet, de l'autre côté de la ligne sans ajouter de route supplémentaire. Le fichier `/etc/ppp/chap-secrets` peut être complété par cette ligne :

```
dialup * <mot_de_passe> 192.168.254.123
```

afin d'indiquer l'adresse IP et le mot de passe de chaque utilisateur.

Ensuite, ajoutez un utilisateur `dialup` et donnez-lui comme mot de passe celui contenu dans le fichier `chap-secrets`. Maintenant, vous pouvez tester votre configuration depuis une machine distante avec `dip -t`, comme c'était le cas plus haut. Si cela fonctionne (c'est-à-dire si `mgetty` répond, et que vous captez des lignes comme celles indiquée en page 499), alors, les propres connexions entrantes de `pppd` doivent aussi fonctionner. Le fichier `/etc/ppp/chap-secrets` peut aussi contenir :

[8] NdT : ARP est l'acronyme d'*Address Resolution Protocol*. Ce protocole permet de traduire des adresses IP à 32 bits en adresses Ethernet à 48 bits. Il est défini par la RFC 826.

```
dialup * <mot_de_passe> *
```

et vous pouvez entreprendre une connexion sortante avec la commande typique `pppd` :

```
pppd \
  connect "chat -S -s -v" 'AT&F1' OK ATDT<telephone> CONNECT ""
     /dev/<modem> 57600 debug crtscts modem lock nodetach
     hide-password defaultroute \
     user dialup \
     noauth
```

Vous devriez être très attentif à disposer d'une configuration DNS irréprochable pour les requêtes directes et inverses de vos adresses IP associées à `pppd`. Le but est qu'aucun service ne bloque en induisant de longs délais d'attente et aussi, pour que les autres machines de l'internet se comportent "amicalement" avec vos connexions d'utilisateur.

Notez que ce qui précède devrait fonctionner avec les télécopies, les connexions distantes, la voix et `uucp` (voir la section 35.3) sur le même modem puisque `mgetty` démarre seulement `pppd` au cas où il détecte une requêtes LCP (*Link Control Protocol* ; il s'agit d'une partie du protocole PPP).[9]

Si vous voulez seulement employer PPP, lisez le fichier de configuration dans `/etc/mgetty+sendfax` (ou `/etc/mgetty` sur Debian) pour désactiver les autres services.

42.6 Usage de `tcpdump`.

Si une connexion sortante se produit de manière inattendue, vous tirerez parti de `tcpdump` pour capter les paquets allant sur votre périphérique `ppp0`. Le résultat de la commande `tcpdump` vous montrea probablement l'origine de l'erreur. Ensuite, jettez un coup d'oeil au port TCP du service en cause et essayez de vous faire une idée du procesus qui a émis le paquet. La commande est :

```
tcpdump -n -N -f -i ppp0
```

La commande `tcpdump` est également discutée à la section 26.10.3.

42.7 RNIS (ISDN) au lieu des modems.

Ce paragraphe donne une brève description du RNIS à l'usage de ceux d'entre vous qui ne sont pas familiers avec ce système. RNIS est l'acronyme de Réseau Numérique à Intégration de Services (en anglais ISDN pour *Integrated Services Digital Network*). Les lignes RNIS se présentent comme des lignes usuelles de télephonie, sauf que RNIS fonctionne avec deux canaux analogiques et deux canaux digitaux. Les deux canaux analogiques sont des lignes téléphoniques

[9] NdT : Il s'agit d'un protocole permettant l'activation d'une ligne téléphonique, testant la ligne, négociant les options et désactivant proprement la ligne quand on n'en a plus besoin.

classiques en tout point –il suffit de brancher son téléphone pour pouvoir effectuer des appels. Les canaux digitaux supportent la transmission de données à raison de 64 Kbits/s. Seul l'équipement RNIS peut être branché sur ces canaux et la facturation est la même que pour les communications téléphoniques. Pour communiquer via les canaux digitaux, vous devez avoir un fournisseur d'accès internet (comme c'est le cas pour la ligne téléphonique classique). PPP fonctionne sur RNIS de la même manière qu'avec un modem. Il fût un temps où seuls de coûteux routeurs RNIS pouvaient fontionner avec le réseau numérique, mais les modems RNIS et les cartes RNIS ISA/PCI sont désormais bon marché afin de permettre à tout le monde d'utiliser le réseau numérique. Généralement, les compagnies de téléphonie installent une ligne RNIS aussi rapidement qu'une ligne téléphonique classique. Aussi, la question surgit de savoir ce que le RNIS apporte de plus ? On a probablement imaginé que le transfert de données simultanément à l'usage d'un téléphone normal pouvait être "la" réponse au service des communications. Il reste à déterminer, cependant, si la video-conférence sur des lignes à 64 Kb deviendra un outil usuel.

Le RNIS n'est pas couvert de manière plus approfondie dans ce livre. Un HOWTO lui est consacré. Soyez tout de même attentif au fait que le RNIS communique de manière *très rapide*, en effectuant des milliers de communications en quelques minutes, ce qui peut s'avérer plutôt coûteux.

Chapitre 43

Sources du noyau LINUX, modules et support matériel.

Ce chapitre explique comment configurer, appliquer un correctif (*patch*) et construire un noyau à partir des sources. La configuration des pilotes de périphériques et des modules est également discutée en détail.

43.1 Composition du noyau.

L'installation d'un noyau consiste à obtenir une image destinée à l'amorçage, les modules, le fichier `System.map`, les en-têtes (qui ne sont réellement utiles qu'aux développeurs) et divers démons. En général, ceux-ci sont déjà fournis par votre distribution. Cet ensemble constitue "Linux", comme on l'appelle communément. Il s'agit d'un code binaire résultant d'environ 50 Mo de code **C**, soit 1,5 millions de lignes de code.

- L'image du noyau LINUX est un fichier de 400 à 600 Ko qui se trouve dans `/boot/` (voir le chapitre 32). Si vous analysez ce répertoire, vous pourrez y voir plusieurs noyaux. Le choix se fait au moment de l'amorçage à l'aide de `lilo` ou de `grub`.
  Le noyau dans `/boot/` est compressé avec l'utilitaire `gzip` et il prend deux fois plus de place lorsqu'il est chargé en mémoire que sous sa forme compressée.
- Le noyau présente aussi des parties séparées, appelées *modules*. Ceux-ci siègent dans `/lib/modules/<version>/`. Ils sont répartis en sous-répertoires à l'intérieur de ce répertoire. Il peut y avoir jusqu'à ~400 modules, totalisant quelque 9 Mo.
  En réalité, les modules sont des fichiers-objets partagés, tels que les fichiers `.o` que nous avons crées à la section 24.1. Il ne s'agit pas des mêmes pilotes de périphériques que ceux de Windows, dans la mesure où il n'est pas possible d'utiliser un module avec un noyau autre que celui pour lequel ce module a été compilé. En conséquence, le terme "module" sera utilisé plutôt que le terme "pilote". Le but de la séparation entre modules et noyau consiste à économiser de la mémoire RAM. Les modules sont parfois compilés "en dur" dans le noyau de la même manière que notre programme `mon_test` a été lié statiquement dans l'exemple de la page 270. Si tel

est le cas, les modules seront absents dans `/lib/modules/<version>/` et, à strictement parler, ils ne devraient plus être désignés par le terme "modules". Dans ce chapitre-ci, nous montrons comment créer des versions de modules compilées en dur ou compilées indépendamment lors de la compilation d'un noyau.

- Ensuite, dans `/boot/`, nous trouvons le fichier `System.map`. Ce dernier est utilisé (i) par `klogd`, un démon qui lit les messages émis par le noyau (résolution de symboles), les reformate et les transmet au démon `syslogd` qui les enregistre dans divers journaux selon la configuration imposée par le fichier `syslog.conf`, et (ii) par `depmod` pour traiter les dépendances de modules ("quel module nécessite le chargement de tel ou tel autre module au préalable").
- Enfin, les en-têtes du noyau `/usr/src/linux/include/` sont utilisés quand certains paquets sont construits.
- Les divers démons devraient être en fonctionnement. Depuis la version `2.2`, ceux-ci ont été ramenés au seul `klogd`. Les autres démons du noyau qui semblent être en cours sont en réalité produits par le noyau lui-même.

43.2 Numéros de version du noyau.

Comme pour les autres paquets, le noyau se présente sous forme de versions : `linux-`*majeur.mineur.patch*. Les noyaux en développement sont caractérisés par des nombres mineurs impairs ; les noyaux stables par des numéros mineurs pairs. Au moment où la traduction a été entreprise, les noyaux-2.6 étaient parfaitement stables (mais pour les grappes de calculs Openmosix,[1] les noyaux 2.4 dominaient encore).

43.3 Modules, `insmod` et commandes associées.

Un module consiste usuellement en un pilote destiné à permettre la communication entre le noyau et un périphérique. Il est soit produit par la commande `mknod` ou existe déjà dans le répertoire `/dev/`. Par exemple, lorsqu'il est chargé, le pilote SCSI est défini avec pour numéro majeur de périphérique : `8` et pour mineur : `0`,`1`,... Le module "sound" a pour majeur : `14` et pour mineur : `3` (`/dev/dsp`). Très souvent, le module "people" est associé à SCSI, Ethernet et Sound. Il existe aussi beaucoup de modules qui supportent des commandes et non du matériel.

Les modules sont chargés à l'aide de la commande `insmod` et sont supprimés à l'aide de `rmmod`. Cela ressemble aux opérations effectuées par l'éditeur de liens (*linking*) dans les `Makefile` de la section 24.1. Si vous voulez afficher les modules chargés, utilisez la commande `lsmod`. Pour les noyaux `2.2`, essayez :

```
insmod /lib/modules/<version>/fs/fat.o
lsmod
insmod fat
lsmod
```

[1] NdT : Openmosix est une extension du noyau Linux. Le lecteur se référera au projet Openmosix à l'adresse : http ://openmosix.sourceforge.net/

Pour les noyaux 2.4, les commandes sont :

```
insmod /lib/modules/<version>/kernel/fs/fat/fat.o
```

Pour les noyaux 2.6, les commandes sont (la commande entre parenthèse permet la recherche des modules; remarquez l'extension .ko de ceux associés à 2.6) :

```
( find /lib/modules/‘uname -r‘/ -type f -iname ’*.ko’ )
modprobe /lib/modules/‘uname -r‘/kernel/fs/fat/fat
```

Par ailleurs, rmmod -a élimine tous les modules inutilisés.

Parfois, certains modules nécessitent le chargement préalable d'autres modules. Si vous tentez d'en charger un et que vous obtenez un message d'erreur comme <nom_module> : unresolved symbol <nom_symbole>, cela signifie qu'un autre module, au moins, devait être chargé au préalable. La commande modprobe charge un module ainsi que ceux dont il dépend. Essayez :

```
insmod /lib/modules/‘uname -r‘/fs/vfat.o
( insmod /lib/modules/<version>/kernel/fs/vfat/vfat.o )
modprobe vfat
```

Cependant, modprobe s'appuie sur une table de *dépendance* des modules. Celle-ci réside dans un fichier /lib/modules/<version>/modules.dep et est créée automatiquement par vos scripts de démarrage via la commande :

```
/sbin/depmod -a
```

quoique vous puissiez l'exécuter manuellement à tout instant. L'affichage produit par lsmod montre aussi les dépendances entre crochets :

```
Modules                 Size   Used  by
de4x5                  41396   1  (autoclean)
parport_probe           3204   0  (autoclean)
parport_pc              5832   1  (autoclean)
lp                      4648   0  (autoclean)
parport                 7320   1  (autoclean) [parport_probe parport_pc lp]
slip                    7932   2  (autoclean)
slhc                    4504   1  (autoclean) [slip]
sb                     33812   0
uart401                 6224   0  [sb]
sound                  57464   0  [sb uart401]
soundlow                 420   0  [sound]
soundcore               2596   6  [sb sound]
loop                    7872   2  (autoclean)
nls_iso8859-1           2272   1  (autoclean)
nls_cp437               3748   1  (autoclean)
vfat                    9372   1  (autoclean)
```

```
fat                30656   1   (autoclean) [vfat]
```

43.4 Interruptions, ports E/S et canaux DMA.

Un module chargé, qui pilote un matériel donné, utilise souvent des ports d'E/S, des IRQs et peut-être un canal DMA (voir le chapitre 4). Vous obtiendrez une liste des ressources utilisées en consultant le répertoire `/proc` :

```
[root@cericon]# cat /proc/ioports

0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
0220-022f : soundblaster
02f8-022f : serial (auto)
0330-0333 : MPU-401 UART
0376-0376 : ide1
0378-037a : parport0
0388-038b : OPL3/OPL2
03c0-03df : vga+
03c0-03f5 : floppy
03f6-03f6 : ide0
03f7-03f7 : floppy DIR
03f8-e47f : serial (auto)
e400-e47f : DC21140 (eth0)
f000-f007 : ide0
f008-f00f : ide1

[root@cericon]# cat /proc/interrupts

              CPU0
 0 :        8409034        XT-PIC  timer
 1 :         157231        XT-PIC  keyboard
 2 :              0        XT-PIC  cascade
 3 :         104347        XT-PIC  serial
 5 :              2        XT-PIC  soundblaster
 6 :             82        XT-PIC  floppy
 7 :              2        XT-PIC  parport0
 8 :              1        XT-PIC  rtc
11 :              8        XT-PIC  DC21140 (eth0)
13 :              1        XT-PIC  fpu
14 :         237337        XT-PIC  ide0
15 :          16919        XT-PIC  ide1
NMI              0
```

```
[root@cericon]# cat /proc/dma

 1 : SoundBlaster8
 2 : floppy
 4 : cascade
 5 : SoundBlater16
```

La configuration qui précède est tout-à-fait typique. Notez que la deuxième colonne associée à l'IRQ indique le nombre de signaux d'interruption reçus depuis le périphérique. Si on écoute les IRQs après un léger déplacement de la souris, on obtient :

```
 3 :          104851        XT-PIC  serial
```

ce qui montre que plusieurs centaines d'interruptions ont été reçues. Une autre entrée utile consiste en `/proc/devices`, qui répertorie les numéros majeurs alloués et ceux effectivement utilisés. Ce fichier s'avère extrêmement utile pour prendre connaissance des périphériques en activité sur votre système.

43.5 Options des modules et configuration des périphériques.

Les modules de périphériques requièrent très souvent des informations à propos de la configuration matérielle. Par exemple, les pilotes de périphériques ISA doivent connaître les IRQ et ports d'E/S auxquels une carte ISA a accès. Cette information est passée au module en tant qu'*options affectées au module* et, ce dernier s'en sert pour sa propre initialisation. Notez que, pour la plupart, les périphériques ne nécessitent aucune information de ce type. Les cartes PCI sont auto-détectées ; ce sont les cartes ISA qui requièrent des options.

43.5.1 5 manières de passer des options à un module.

1. Si un module a été compilé dans le noyau, il sera initialisé au moment de l'amorçage. `lilo` passe des options du module au noyau, depuis la ligne de commande, derrière l'invite `LILO :`. Par exemple, vous pouvez saisir (voir la section 5.4) :

   ```
   linux aha1542=<portbase>[,<buson>,<busoff>[,<dmaspeed>]]
   ```

   afin d'initialiser le pilote SCSI Adaptec 1542. Que représentent ces options ? La réponse peut être obtenue en consultant le fichier `/usr/src/linux/‘uname -r‘/drivers/scsi/aha1542.c`. Les commentaires explicitant la signification de ces options se trouvent au début du fichier.

2. Si vous utilisez `LOADLIN.EXE`, ou d'autres chargeurs de noyau DOS ou MS-Windows, vous pouvez aussi passer des options. Toutefois, nous n'irons pas plus loin pour ce cas.

3. `/etc/lilo.conf` peut prendre une option `append =`, tel que cela a été discuté à la page 359. Cette méthode permet de passer des options au noyau

comme si vous les saisissiez derrière l'invite `LILO :`. La ligne `lilo.conf` équivalente est :

```
append = aha1542=<portbase>[,<buson>,<busoff>[,<dmaspeed>]]
```

Ceci constitue la méthode la plus courante pour passer des options au noyau.

4. Les commandes `insmod` et `modprobe` peuvent aussi prendre des options qui sont passées au(x) module(s). La méthode est assez différente par rapport à `append =`. Par exemple, vous pouvez passer des options à un module Ethernet (compilé en dur) avec les commandes suivantes :

```
append = ether=9,0x300,0xd0000,0xd4000,eth0
append = ether=0,0,eth1
```

et ce, depuis le fichier `/etc/lilo.conf`. Cependant, si vous utilisez `modprobe` dans le cas des mêmes modules compilés en-dehors du noyau, les options devront être spécifiées de la manière qui suit :

```
modprobe wd irq=9 io=0x300 mem=0xd0000 mem_end=0xd4000
modprobe de4x5
```

Notez que les paramètres `0xd0000`, `0xd4000` sont seulement applicables à quelques cartes Ethernet et qu'ils sont usuellement omis. Sachez aussi que les `0` dans l'expression `ether=0,0,eth1` signifient que l'autodétection doit être appliquée. Pour déterminer les options qu'un module peut prendre, vous utiliserez la commande `modinfo` qui montre que le pilote `wd` est un des quelques pilotes Ethernet pouvant déterminer leur propre usage RAM [ce point ne sera pas discuté plus avant, mais les cartes peuvent parfois utiliser des zones de la mémoire de manière autonome] :

```
[root@cericon]# modinfo -p /lib/modules/<version>/net/wd.o
( [root@cericon]# modinfo -p /lib/modules/<version>/kernel/drivers/net/wd.o )
io int array (min = 1, max = 4)
irq int array (min = 1, max = 4)
mem int array (min = 1, max = 4)
mem_end int array (min = 1, max = 4)
```

5. Le fichier `/etc/modules.conf` [parfois appelé `/etc/conf.modules`, une dénomination désormais obsolète] contient les options, par défaut, relatives à `modprobe`, au lieu que celles-ci soient passées en ligne de commande derrière `modpro_be`. Il s'agit de la méthode préférentielle et la plus populaire pour passer des options de modules. Concernant notre exemple Ethernet, nous aurions :

```
alias eth0 wd
alias eth1 de4x5
options wd irq=9 io=0x300 mem=0dx0000 mem_end=0xd4000
```

Le fait d'avoir construit un fichier `/etc/modules.conf` permet le *charge-*

ment dynamique des modules. Ceci signifie que le noyau charge automatiquement les modules nécessaires au cas où le pilote est requis (de la même manière que lorsqu'`ifconfig` est utilisé pour les périphériques Ethernet). Le noyau essaie simplement `/sbin/modprobe eth0`, et la ligne `alias` informe `modprobe` d'exécuter `/sbin/modprobe wd`. Par ailleurs, la ligne `options` signifie qu'il faut exécuter `/sbin/modprobe wd irq=9 io=0x300 mem=0xd0000 mem_end=0xd4000`. De cette manière, `/etc/modules.conf` consiste en une table de correspondances "périphériques – pilotes".

43.5.2 Sources de documentation des modules.

A l'heure actuelle, il n'existe pas de résumé complet de toutes les options de modules avec des exemples d'utilisation propres aux cinq méthodes que nous venons de passer en revue. Ceci est principalement dû au fait que les utilisateurs sont toujours intéressés par la configuration d'un périphérique particulier et que la documentation spécialisée permet de résoudre le problème. Par ailleurs, certains modules spéciaux sont presque toujours compilés en dehors du noyau alors que d'autres s'arrangent mieux d'une compilation en dur.

Si vous avez à faire à un périphérique rare ou ancien, il vaut mieux lire le HOWTO approprié : `BootPrompt-HOWTO`, `Ethernet-HOWTO` et `Sound-HOWTO`. Un tel périphérique peut aussi être documenté dans `/usr/src/linux/<version>/-Documentation/` ou sous un des répertoires `/sound/` et `/networking/`. Il s'agit de la documentation écrites par les développeurs du module. Le fichier `/usr/-src/linux/Documentation/networking/net-modules.txt` est d'un grand intérêt, car, même si elle est périmée, cette documentation donne une liste complète des modules de cartes réseau et des options qu'ils prennent. Une autre source de documentation consiste en le code **C** lui-même, comme par exemple `aha1542.c` (nous avons parlé de ce module ci-dessus). Cette documentation explique les options qui doivent être utilisées dans `/etc/lilo.conf` et `/etc/modules.conf`. Souvent, néanmoins, cette documentation est un peu difficile d'accès. Par ailleurs, un pilote est souvent écrit avec l'intention d'être compilé en dur ou non. En principe, il devrait supporter les deux modes de compilation. Le choix de son mode de compilation est indiqué dans la documentation ou dans le code source **C**.

43.6 Configuration de divers périphériques.

D'autres exemples permettant de configurer des périphériques vont à présent être vus, mais nous nous concentrerons sur quelques cas, seulement. Voici comment on procède *généralement*.

43.6.1 Son et `pnpdump`.

Les cartes son ISA (PnP ou Plug-and-Play) (cas des cartes SoundBlaster) sont parmi les plus populaires parmi les utilisateurs de LINUX. En l'occurence, nous utilisons une telle carte pour montrer comment configurer une carte PnP ISA en quelques minutes. Donc, *ce dont nous parlons s'applique à d'autres cartes que les seules cartes son.*

L'utilitaire `isapnp` prend un argument (le fichier `/etc/isapnp.conf`) et permet de configurer tous les périphériques PnP ISA sur les IRQ et ports E/S spécifiés dans ce fichier. Le fichier `/etc/isapnp.conf` est un peu complexe mais il peut être créé à l'aide de la commande `pnpdump`. `pnpdump` émet un fichier d'exemple sur stdout, qui contient les valeurs d'IRQ et de ports E/S associées à nos périphériques. Vous devez le modifier pour y supprimer les valeurs non-utilisées. Autrement, vous pouvez employer `pnpdump --config` pour obtenir un fichier `/etc/isapnp.conf` contenant les valeurs correctes d'IRQ, de ports E/S et de canaux DMA obtenus par analyse des entrées `/proc`. On pratique ainsi :

```
[root@cericon]# pnpdump --config | grep -v '^\(#.*\|\)$' > /etc/isapnp.conf
[root@cericon]# isapnp /etc/isapnp.conf

Board 1 has Identify c9 00 00 ab fa 29 00 8c 0e : CTL0029 Serial No 44026 [checksum c9]
CTL0029/44026[0] {Audio       } : Ports 0x220 0x330 0x388 ; IRQ5 DMA1 DMA5 --- Enabled OK
CTL0029/44026[1] {IDE         } : Ports 0x168 0x36E ; IRQ10 --- Enables OK
CTL0029/44026[2] {Game        } : Ports 0x200 ; --- Enabled OK
```

ce qui permet de n'utiliser que deux commandes pour configurer toutes les cartes PnP ISA. Notez que le fichier `/etc/isapnp.gone` peut être employé pour faire en sorte que `pnpdump` évite d'utiliser certains ports E/S et IRQ. Voici un exemple de ce fichier :

```
IO 0x378,2
IRQ 7
```

qui permet d'éviter les conflits avec le port parallèle. La commande `isapnp /etc/isapnp.conf` doit être exécutée à chaque amorçage et devrait normalement être prête pour une utilisation par les scripts de démarrage.

A présent que votre carte ISA est identifiée, vous allez installer les modules nécessaires. Lisez le fichier `/etc/isapnp.conf` et la sortie d'`isapnp` ci-dessus de manière à référencer les ports d'E/S comme options :

```
alias sound-slot-0 sb
alias sound-service-0-0 sb
alias sound-service-0-1 sb
alias sound-service-0-2 sb
alias sound-service-0-3 sb
alias sound-service-0-4 sb
alias sound-synth-0 sb
post-install sb /sbin/modprobe "-k" "adlib_card"
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
options adlib_card io=0x388       # FM synthesizer
```

A présent, lancez `tail -f /var/log/messages /var/log/syslog`, et sur un autre terminal :

```
depmod -a
modprobe sb
```

Si vous ne recevez pas d'erreur provenant du noyau (ou tout autre type d'erreur), vos périphériques sont désormais fonctionnels.

Maintenant, il est souhaitable que le chargement du module se fasse de manière dynamique. Enlevez tous les modules son avec `rmmod -a` (éventuellement, faites cette opération manuellement) et, ensuite, essayez :

```
aumix
```

Dans le journal du noyau, vous devriez voir un message similaire à :

```
Sep 24 00 :45 :19 cericon kernel : SoundBlaster audio driver
 Copyright (C) by Hannu Savolainen 1993-1996
Sep 24 00 :45 :19 cericon kernel : SB 4.13 detected OK (240)
```

A ce stade, lancez la commande :

```
playmidi <un_fichier>.mid
```

Vous devriez voir dans le journal du noyau un message comme celui-ci :

```
Sep 24 00 :51 :34 cericon kernel : Soundblaster audio driver
 Copyright (C) by Hannu Savolainen 1993-1996
Sep 24 00 :51 :34 cericon kernel : SB 4.13 detected OK (240)
Sep 24 00 :51 :35 cericon kernel : YM3812 and OPL-3 driver
 Copyright (C) by Hannu Savolainen, Rob Hooft 1993-1996
```

Si vous devez décommenter les lignes `alias`, un message tel que `modprobe : Can't locate module sound-slot-0 :` devrait être émis. Ceci indique que le noyau essaye la commande `/sbin/modprobe sound-slot-0`, une procédure destinée à insérer une ligne `alias`. En réalité, `sound-service-0-0`,`1`,`2`,`3`,`4` représente les périphériques `/dev/mixer`,`sequencer`,`midi`,`dsp`,`audio`, respectivement. Le terme `sound-slot-0` désigne une carte qui devrait fournir tous ces services. L'option `post-install` indique qu'il faut réaliser une commande supplémentaire après l'installation du module, ceci pour prendre en compte le pilote du séquenceur Adlib. [...]

43.6.2 Port parallèle.

Le module associé au port parallèle est moins complexe à mettre en oeuvre :

```
alias parport_lowlevel parport_pc
options parport_lowlevel io=0x378 irq=7
```

Asurez-vous seulement que l'IRQ et le port E/S concernés correspondent bien aux caractéristiques de votre CMOS (voir la section 4.3) et qu'ils ne sont pas en conflit avec d'autres périphériques.

43.6.3 NIC - Ethernet, PCI et l'ancien ISA.

Ici, nous utilisons des cartes ISA non-PnP et des cartes PCI, en s'appuyant sur des périphériques Ethernet (NIC est un acronyme pour *Network Interface Card*, c'est-à-dire une carte 10 ou 100 Mo).

Concernant les anciennes cartes ISA (à cavaliers), vous devez vérifier les fichiers contenus dans `/proc/` pour déterminer les IRQ et les ports E/S non-utilisés et déplacer les cavaliers de manière appropriée. Alors, vous pouvez exécuter un `modprobe` comme d'ordinaire. Par exemple :

```
modinfo -p ne
modprobe ne io=0x300 irq=9
```

Naturellement, en cas de chargement dynamique, votre fichier `/etc/modules.conf` doit contenir les lignes :

```
alias eth0 ne
options ne io=0x300 irq=9
```

Il arrivera peut-être que vous disposiez d'une carte (type PnP) configurable de manière logicielle mais seulement à l'aide d'un utilitaire DOS. Dans ce cas, la compilation du module dans le noyau entraînera son auto-détection lors du démarrage *sans qu'il soit nécessaire d'en faire la configuration.*

Le pire scénario est celui d'une carte dont la fabrication est inconnue et dont vous ne connaissez pas les IRQ et E/S. Le numéro de la puce peut parfois livrer une indication (faites un `grep` sur les sources du noyau avec ce numéro) mais ce n'est pas garanti. Pour qu'une telle carte puisse fonctionner, compilez plusieurs modules qui vous semblent correspondre à la carte. L'expérience permet de s'améliorer rapidement à ce petit jeu. Si vos déductions sont correctes, votre carte sera très certainement découverte lors du réamorçage. Vous pouvez trouver les valeurs d'IRQ et E/S dans `/proc/` ou alors, vous pouvez lancer la commande `dmesg` pour déterminer la ou les ligne(s) se rapportant à votre périphérique. S'il s'agit d'une carte Ethernet, une des lignes doit débuter avec `eth0 :...` et le résultat de `dmesg` devrait fournir de l'information à propos du pilote.

Les périphériques PCI ne requièrent presque jamais que les IRQ ou les ports E/S soient fournis comme options. Tant que votre module est correct, une simple commande :

```
modprobe <module>
```

fonctionnera toujours. Cependant, la détermination du module correct peut encore s'avérer un problème parce que les fournisseurs donnent divers noms commerciaux à une carte donnée. L'utilitaire `scanpci` (qui est une fonctionnalité de **X**) vérifie vos *slots* PCI. L'exécution de `scanpci` peut engendrer une série de messages comme ceux-ci :

```
. . .
pci bus 0x0 cardnum 0x09 function 0x0000 : vendor 0x1011 device 0x0009
 Digital DC21140 10/100 Mb/s Ethernet
```

```
pci bus 0x0 cardnum 0x0b function 0x0000 : vendor 0x8086 device 0x1229
 Intel 82557/8/9 10/100MBit network controller

pci bus 0x0 cardnum 0x0c function 0x0000 : vendor 0x1274 device 0x1371
 Ensoniq es1371
```

`lspci` est un autre utilitaire provenant du paquet `pciutils`. Il donne une information complète là où `scanpci` n'en fournit parfois aucune. Ainsi un petit script (donné entre parenthèses pour les noyaux 2.4) tel que :

```
for i in /lib/modules/<version>/net/* ; do strings $i \
                              | grep -q -i 21140 && echo $i ; done

( for i in /lib/modules/<version>/kernel/drivers/net* \
            ; do strings $i | grep -q -i 21140 && echo $i ; done )

for i in /lib/modules/<version>/net/* ; do strings $i \
                               | grep -q -i 8255 && echo $i ; done

( for i in /lib/modules/<version>/drivers/net/* \
             ; do strings $i | grep -q -i 8255 && echo $i ; done )
```

révèlera la présence de trois modules `de4x5.o`, `eepro100.o` et `tulip.o` dont deux sont parfaitement corrects. Sur un autre système, `lscpi` donne :

```
.
.
.
00 :08 :0 Ethernet controller : Macronix, Inc. [MXIC] MX987x5 (rev 20)
00 :0a :0 Ethernet controller : Accton Technology Corporation SMC2-1211TX (rev
10)
```

et le même script `for ... grep...Accton` indique `rtl8139.o` et `tulip.o` (le premier est correct) ; alors que `for ... grep ... Macronix` (ou même `987`) donne `tulip.o`, ce qui paralyse la machine. Il nous reste encore à faire fonctionner la carte, une fois son module identifié. [...].

43.6.4 ID du vendeur PCI et ID de périphérique.

La norme PCI supporte le principe selon lequel chaque vendeur et chaque périphérique possède des identificateurs (ID) hex uniques. Par exemple, Intel a choisi de représenter ses périphériques avec le numéro 0x8086. Les cartes PCI fournissent ces données lorsqu'elles sont sondées. Vous obtiendrez ces numéros en utilisant `lspci`, `scanpci` ou `cat /proc/pci`. Le fichier `/usr/share/pci.ids` (sur Debian, `/usr/share/misc/pci.ids`) provenant du paquet `pciutils` contient une table complète de correspondance de tous les IDs et des noms de périphériques.

Le paquet `kudzu` possède aussi une table `/usr/share/kudzu/pcitable` contenant l'information que nous recherchons *réellement* : une table de correspondance des modules du noyau. Ceci vous permet d'utiliser la méthode logique

utilisée pour localiser les modules PCI corrects à partir des données /proc/pci du noyau. Le format du fichier est facile à comprendre. A titre d'exercice, vous devriez essayer d'écrire un script de shell pour réaliser une recherche de manière automatique.

43.6.5 PCI et son.

Le résultat produit par scanpci (ci-dessus) indique une carte Ensoniq, également populaire, parfois intégrée à la carte-mère. L'ajout de la ligne suivante :

```
alias sound es1371
```

à votre fichier /etc/modules.conf activera cette carte. Il est relativement facile de trouver le type de carte en y regardant de plus près puisque les cartes Ensoniq présentent une marque "es1371" gravée sur la puce.

43.6.6 Pilotes de cartes son commerciaux.

Si votre carte n'est pas reprise sous /usr/src/<version>/Documentation/-sound/, il vous sera sans doute possible d'obtenir un pilote auprès d'*Open Sound* http ://www.opensound.com. Si vous n'y arrivez pas, envoyez un courriel au fabricant pour vous plaindre [...].

43.6.7 Le projet ALSA pour le son.

Le projet ALSA (*Advanced Linux Sound Architecture* http ://www.alsa-project.org/) est destiné à fournir le meilleur support de son pour le noyau LINUX. Si votre carte n'est pas supportée par le noyau standard ou que vous ne trouvez pas de modules indépendants, visitez le site web d'ALSA.

43.6.8 Cartes Ethernet multiples.

Si vous possédez plus d'une carte Ethernet, vous pouvez les mentionner dans votre fichier /etc/modules.conf, comme stipulé dans la section 43.5. Par défaut, les modules compilés (en dur) dans le noyau ne répondent qu'à une seule carte (eth0). L'ajout de la ligne :

```
append = "ether=0,0,eth1 ether=0,0,eth2 ether=0,0,eth3"
```

amène les cartes eth0, eth1 et eth2 à être sondées. Par ailleurs, le remplacement des 0 par les valeurs réelles peut forcer certaines interfaces à être liées aux cartes physiques. Si toutes les cartes sont de type PCI, vous devrez absolument obtenir leur ordre d'attribution par tâtonnements.

Si vous avez deux cartes identiques, votre noyau pourrait émettre un avertissement lors du double chargement du même module. L'option -o d'insmod permet d'utiliser une astuce de chargement faisant "croire" au noyau que le premier module n'est pas encore chargé :

```
alias eth0 3c509
alias eth1 3c509
```

```
options eth0 -o 3c509-0 io=0x280 irq=5
options eth1 -o 3c509-1 io=0x300 irq=7
```

Néanmoins, avec les deux cartes PCI suivantes, il n'y aura pas de problèmes :

```
alias eth0 rtl8139
alias eth0 rtl8139
```

43.6.9 Disques SCSI.

L'acronyme SCSI (prononcez *scuzzy*) provient de *Small Computer System Interface*. SCSI désigne tout à la fois, une nappe, une norme et un protocole de communication entre périphériques et ordinateurs. Les nappes SCSI ne peuvent être connectées qu'à des disques durs SCSI. Elles existent sous différentes versions destinées à améliorer la vitesse des transferts, les derniers modèles (Ultra-Wide SCSI) étant constitués de nappes fines et garnies d'un grand nombre de connexions-aiguilles. Contrairement à IDE, le protocole SCSI permet de connecter aussi des systèmes de bandes, des scanners, et bien d'autres périphériques matériels. En théorie, le système SCSI autorise de nombreux ordinateurs à partager le même périphérique, bien que cette pratique soit assez peu courante. Du fait que de nombreuses plate-formes UNIX ne supportent que du SCSI, ce dernier est devenu une partie intégrante des systèmes d'exploitation UNIX.

Le protocole SCSI a introduit les notions de *LUN* (*Logical Unit Number*), de *bus* et d'*ID*. Il s'agit d'un triplet de nombres attribué à chaque périphérique associé aux cartes SCSI utilisées (lorsqu'il y a en plus d'une), auxs câbles SCSI posés sur ces cartes et au périphérique connecté à ces cartes (la norme SCSI prévoit de supporter de nombreuses cartes). Le noyau attribue des fichiers de périphérique : `/dev/sda`, `/dev/sdb`, etc.

De nombreux éléments pourraient être décrits mais l'essentiel tient dans la commande `insmod <pci-scsi-driver>`. Vous pourrez accéder au disque via `/dev/sd?`, aux lecteurs de bandes via `/dev/scd?` ou aux scanners par `/dev/sg?` [les programmes utilisateur pour les scanners contiennent la documentation relatives au périphérique à utiliser]. Les périphériques SCSI sont souvent fournis avec leur propre BIOS auquel on peut accéder au moment du démarrage (comme c'est le cas pour votre CMOS). Ceci permet d'effectuer des réglages. Dans certains cas, lorsque votre distribution permet de compiler les modules en dehors du noyau, vous serez peut-être amené à charger un des modules suivants (dans l'ordre) : `sd_mod.o`, `sr_mod.o` ou `sg.o`. Le noyau réclamera sans doute le module central `scsi_mod.o` et les fichiers de périphériques dans `/dev/` devront peut-être être créés. Une méthode sûre consiste à opérer ainsi :

```
cd /dev
./MAKEDEV -v sd
./MAKEDEV -v st0 st1 st2 st3
./MAKEDEV -v scd0 scd1 scd2 scd3
./MAKEDEV -v sg
```

Il est recommandé que vous compiliez en dur le support des cartes SCSI (aussi appelé *SCSI Host Adapter*) ainsi que le support pour les lecteurs de bandes, les

CD-ROMs, etc. Lorsque votre système sera réamorcé, chacun des périphériques sera autodétecté. Voici un exemple de message produit au réamorçage et qui concerne un disque et un lecteur de bandes SCSI :

```
(scsi0) <Adaptec AIC-7895 Ultra SCSI host adapter> found at PCI 0/12/0
(scsi0) Wide Channel A, SCSI ID=7, 32/255 SCBs
(scsi0) Cables present (Int-50 YES, Int-68 YES, Ext-68 YES)
(scsi0) Illegal cable configuration!! Only two
(scsi0) connectors on the SCSI Controller may be in use at a time!
(scsi0) Downloading sequencer code... 384 instructions dowloaded
(scsi1) <Adaptec AIC-7895 Ultra SCSI host adapter> found at PCI 0/12/1
(scsi1) Wide Channel B, SCSI ID=7, 32/255 SCBs
(scsi1) Downloading sequencer code... 384 instructions dowloaded
scsi0 : Adaptec AHA274x/284x/294x (EISA/VLB/PCI-Fast SCSI) 5.1.28/3.2.4
       <Adaptec AIC-7895 Ultra SCSI host adapter>
scsi1 : Adaptec AHA274x/284x/294x (EISA/VLB/PCI-Fast SCSI) 5.1.28/3.2.4
       <Adaptec AIC-7895 Ultra SCSI host adapter>
scsi : 2 hosts.
(scsi0 :0 :0 :0) Synchronous at 40.0 Mbyte/sec, offset 8.
 Vendor : FUJITSU   Model : MAE3091LP         Rev : 0112
 Type : Direct-Access                        ANSI SCSI revision : 02
Detected scsi disk sda at scsi0, channel 0, id 0, lun 0
(scsi0 :0 :0 :0) Synchronous at 10.0 Mbyte/sec, offset 15.
 Vendor : HP        Model : C1533A            Rev : A708
 Type : Sequential-Access                    ANSI SCSI revision : 02
Detected scsi tape st0 at scsi0, channel 0, id 3, lun 0
scsi : detected 1 SCSI tape 1 SCSI disk total.
SCSI device sda : hdwr sector= 512 bytes. Sectors= 17826240 [8704 MB] [8.7 GB]
 .
 .
 .
Partition check :
 sda : sda1
 hda : hda1 hda2 hda3 hda4
 hdb : hdb1
```

Vous devriez consulter la section 32.5 pour déterminer comment amorcer sur un disque SCSI quand le module requis se trouve dans un système de fichiers qui est sur le disque SCSI qui, à son tour, nécessite le module.

Pour utiliser un lecteur de bandes, revoyez la discussion à la page 174.

43.6.10 Terminaisons SCSI et refroidissement.

Voici une section de grande importance relative aux périphériques SCSI. Vous avez très probablement l'habitude des nappes IDE qu'il suffit de brancher pour qu'elles fonctionnent. Les nappes SCSI s'en différencient singulièrement car elles nécessitent une adaptation d'impédance et une terminaison. Il s'agit de termes utilisés en électricité/électronique. Fondamentalement, cela implique que vous devez utiliser des nappes SCSI de haute-qualité et *fermer* le circuit SCSI. Les nappes SCSI permettent à de nombreux disques et lecteurs de bandes

SCSI d'être connectés à une seule d'entre elles. *Fermer le circuit* signifie qu'il faut placer soit des cavaliers, soit un commutateur sur le dernier périphérique de la nappe. Cela implique aussi de brancher le connecteur du dernier câble dans une prise adéquate. La documentation de votre adaptateur et de votre disque devrait vous expliquer comment procéder techniquement. Si votre circuit n'est pas électriquement fermé, votre système pourra fonctionner mais avec des erreurs de disque(s) durant la vie de la machine. Notez que les périphériques SCSI les plus récents possèdent d'office une terminaison électrique.

Le refroidissement est un autre problème à prendre en considération. Quand la documentation relative à un périphérique disque recommande d'utiliser de l'air pulsé, *respectez cette consigne*. Les périphériques SCSI montent considérablement en température et peuvent se dégrader s'ils ne sont pas correctement refroidis. Le fait de pulser de l'air signifiera le plus souvent se procurer un petit ventilateur et le disposer de manière correcte. N'hésitez pas à vous munir d'un boîtier suffisamment large avec quelques centimètres de séparation entre les périphériques. Quiconque a déjà ouvert un serveur de haute de gamme se rend compte de l'attention portée au refroidissement.

43.6.11 Graveurs de CD.

Durant l'amorçage, un système avec un graveur CD (ATAPI-IDE) et un lecteur CD-ROM ordinaire devrait afficher un message similaire à celui-ci :

```
hda : FUJITSU MPE3084AE, ATA DISK drive
hdb : CD-ROM 50X L, ATAPI CDROM drive
hdd : Hewlett-Packard CD-Writer Plus 9300, ATAPI CDROM drive
```

Ces périphériques devraient fournir ces messages avant que n'apparaisse l'invite `LILO :` *et ce, pour indiquer qu'ils sont fonctionnels.*

Les lignes de `/etc/modules.conf` permettant l'auto-détection sont :

```
alias   scd0 sr_mod                # Load sr_mod upon access of /dev/scd0
alias   scsi_hostadaptor ide-scsi  # SCSI hostadaptor emulation
options ide-cd="hda hdc ddd" # Our normal IDE CD is on /dev/hdb
```

La ligne `alias scd0` doit être enlevée si `sr_mod` est compilé en dur dans le noyau. Effectuez une recherche dans le répertoire `/lib/modules/<version>/`. Notez que le noyau ne supporte pas directement les graveurs CD-ROMs ATAPI. Le module `ide-scsi` émule l'adapteur SCSI au nom du CD-ROM ATAPI.[2] Le logiciel de gravure de CD s'attend à communiquer avec `/dev/scd?`, et le module `ide-scsi` permet à ce périphérique d'être vu comme un graveur CD SCSI. [Les véritables graveurs CD SCSI sont plus chers.] Un avertissement, toutefois : votre pilote de CD-ROM IDE ordinaire, `ide-cd`, cherchera aussi à communiquer avec votre graveur CD comme s'il était un lecteur CD-ROM normal. L'option `ignore` permet au module `ide-cd` de passer au-dessus de tout périphérique qui ne devrait pas être sondé sur votre système. Il peut s'agir de disques durs, de graveurs CD, ou d'un second maître non-existant. *Cependant*, il n'existe pas de méthode pour donner une option `ignore` à un module `ide-cd` compilé en dur

[2] NdT : depuis le noyau `2.6`, il en va autrement.

dans le noyau (ce qui est le cas pour de nombreuses distributions).

Une alternative consiste à compiler en dur le support pour `ide-scsi` et à abandonner complétement le support `ide-cd`. Votre CD-ROM classique fonctionnera parfaitement comme un CR-ROM en lecture-seule sous émulation SCSI [y compris avec de la musique]. Voici la configuration du noyau associée :

```
<*> Enhanced IDE/MFM/RLL disk /cdrom/tape/floppy support
< >     Include IDE/ATAPI CDROM support
<*>     SCSI Emulation support

<*> SCSI support
<*> SCSI CD-ROM support
[*]     Enable vendor-specific extensions (for SCSI CDROM)
<*> SCSI generic support
```

Il n'y a pas de complément de configuration à réaliser et, au redémarrage de la machine, vous devriez obtenir des messages comme :

```
scsi0 : SCSI host adapter emulation for IDE ATAPI devices
scsi : 1 host.
 Vendor : E-IDE     Model : CD-ROM 50X L     Rev : 12
 Type :   CD-ROM                             ANSI SCSI revision : 02
Detected scsi CD-ROM sr0 at scsi0, channel 0, id 0, lun 0
 Vendor : HP        Model : CD-writer+ 9300  Rev : 1.0b
 Type : CD-ROM                               ANSI SCSI revision : 02
Detected scsi CD-ROM sr1 at scsi0, channel 0, id 1, lun 0
scsi : detected 2 SCSI generics 2 SCSI cdroms total.
sr0 : scsi3-mmc drive : 4x/50x cd/rw xa/form2 cdda tray
Uniform CD-ROM driver Revision : 3.10
srl : scsi3-mmc drive : 32x/32x writer cd/rw xa/form2 cdda tray
```

Si vous possédez un véritable graveur SCSI, la compilation du support de votre carte SCSI permettra sa détection. Ainsi, par exemple, le périphérique sur lequel vous monterez votre CD-ROM sera `/dev/scd0` et pour votre graveur CD, il s'agira de `/dev/scd1`.

Pour enregistrer un disque CD, la commande `cdrecord` est connue pour sa robustesse bien qu'il existe de nombreux frontaux graphiques. Pour localiser l'ID du CD, exécutez :

```
cdrecord --scanbus
```

de manière à obtenir le triplet numérique 'LUN,bus,ID'. Vous utiliserez alors ce triplet comme argument associé à `dev` suite à la commande `cdrecord`. Par exemple, il est possible d'exécuter :

```
mkisofs -a -A 'Paul Sheer' -J -L -r -P PaulSheer \
        -p www.icon.co.za/~psheer/ -o mon_iso /mon/repertoire
cdrecord dev=0,1,0 -v speed=10 -isosize -eject mon_iso
```

de manière à créer un CD-ROM ISO9660 d'un fichier qui se trouve dans le

répertoire `/mon/repertoire`. Ceci s'avère particulièrement utile pour les sauvegardes. (L'option `-a` devrait pouvoir être omise dans les nouvelles versions de la commande.) Soyez attentif à ne pas dépasser la vitesse limite de votre graveur.

43.6.12 Périphériques série.

Il n'y a pas de modules à charger pour permettre au clavier et à la souris d'être détectés. Les périphériques série normaux (de COM1 à COM4 sous DOS/Windows) sont auto-détectés à l'amorçage et sont disponibles en tant que fichiers périphériques `/dev/ttyS0` à `/dev/ttyS3`. Un message au démarrage tel que celui-ci :

```
Serial driver version 4.27 with MANY_PORTS MULTIPORT SHARE_IRQ enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
```

atteste d'une détection correcte.

En revanche, les cartes série à ports multiples peuvent s'avérer difficile à configurer. Ces périphériques forment une catégorie spéciale de matériel. Pour la plupart, ces périphériques utilisent une puce *16550A UART* (*Universal Asynchronous Receiver Transmitter*) qui est analogue à celle du port série intégré dans votre carte-mère. Le code série générique du noyau les supporte, et vous ne devriez pas utiliser un module séparé. La puce UART *consitue* réellement le port série et est caractérisée par des sous-types `8250`, `16450`, `16550`, `16550A`, `16650`, `16650V2` et `16750`.

Pour faire fonctionner ces cartes, il vous faudra utiliser la commande `setserial`. Elle consite à configurer le pilote série intégré au noyau. Un cas typique : la carte ISA non-PnP à 8 ports et à cavaliers (IRQ `5` ; ports E/S `0x180-0x1BF`). Notez que, contrairement à la plupart des périphériques, beaucoup de périphériques série peuvent partager la même IRQ [La raison en est que les périphériques série utilisent un port E/S pour indiquer le périphérique qui envoie une interruption. La CPU vérifie chaque périphérique série chaque fois qu'une interruption a lieu]. La carte est configurée avec ce script :

```
cd /dev
./MAKEDEV -v ttyS4
./MAKEDEV -v ttyS5
./MAKEDEV -v ttyS6
./MAKEDEV -v ttyS7
./MAKEDEV -v ttyS8
./MAKEDEV -v ttyS9
./MAKEDEV -v ttyS10
./MAKEDEV -v ttyS11
/bin/setserial -v /dev/ttyS4 irq 5 port 0x0180 uart 16550A skip_test
/bin/setserial -v /dev/ttyS5 irq 5 port 0x0188 uart 16550A skip_test
/bin/setserial -v /dev/ttyS6 irq 5 port 0x0190 uart 16550A skip_test
/bin/setserial -v /dev/ttyS7 irq 5 port 0x0198 uart 16550A skip_test
/bin/setserial -v /dev/ttyS8 irq 5 port 0x01A0 uart 16550A skip_test
/bin/setserial -v /dev/ttyS9 irq 5 port 0x01A8 uart 16550A skip_test
```

```
/bin/setserial -v /dev/ttyS10 irq 5 port 0x01B0 uart 16550A skip_test
/bin/setserial -v /dev/ttyS11 irq 5 port 0x01B1 uart 16550A skip_test
```

Vous devriez être en mesure d'utiliser des périphériques en tant que ports normaux. Notez que vous devriez vous attendre à voir les interruptions en cours sous `/proc/interrupts`. Concernant les périphériques série, ceci n'est vrai qu'après que les données aient commencés à fluer. Néanmoins, vous pouvez vérifier le fichier `/proc/tty/driver/serial` pour obtenir davantage d'information sur l'état des périphériques. La page de man de `setserial` nous en dit plus long sur les différents UARTs et les problèmes de compatibilité. Elle décrit aussi la technique d'auto-détection de l'UART, des IRQ et des ports E/S (ceci dit il est préférable de toujours d'être sûr des propriétés de votre carte et de ne pas utiliser l'auto-détection).

Les périphériques série engendrent d'innombrables problèmes. Le `Serial-HOWTO` est d'ailleurs très long. Il vous aidera très certainement à résoudre d'éventuels problèmes car il contient de nombreuses données techniques et discute en profondeur du support spécial du noyau dans le cas de nombreuses cartes non-standards.

43.7 Cartes modem.

Dans cet ouvrage, nous avons traité des modems externes qui sont connectés au port série auxiliaire de votre machine. Cependant, les cartes modem ISA sont bon marché et possèdent leur propre port série interne. Ce type de carte peut être traité comme nous venons de le voir, c'est-à-dire à l'instar des cartes ISA multiports (mais, en l'occurrence, avec un seul port). Il suffit d'ajuster les cavaliers du port E/S et de l'IRQ, puis de lancer la commande `setserial /dev/ttyS3...` .

Sachez encore qu'il existe des "winmodems". Il ne s'agit en fait que de cartes son. Votre système d'exploitation doit produire les signaux requis par le protocole de ces cartes, de la même manière que dans le cas des modems classiques. Voici trois ressources traitant de ces modems spéciaux : http ://linmodems.technion.ac.il/, http ://www.idir.net/~gromitkc/winmodems.html et http ://www.linmodems.org.

43.8 Davantage à propos de `LILO` : options.

Le `BootPrompt-HOWTO` contient une liste exhaustive d'options qui peuvent être passées à l'invite de démarrage pour réaliser des opérations spéciales comme, par exemple, monter une racine NFS. Si vous avez une idée de ce que LINUX supporte, ce document est très important à lire.

43.9 Construction du noyau.

Pour les noyaux inférieurs à 2.6, le sommaire des commandes consiste en ceci :

```
cd /usr/src/linux
make mrproper
```

```
make menuconfig
make dep
make clean
make bzImage
make modules
make modules_install
cp /usr/src/linux/arch/i386/boot/bzImange /boot/vmlinuz-<version>
cp /usr/src/linux/System.map /boot/System.map-<version>
```

Enfin, modifiez `/etc/lilo.conf` et exécutez la commande `lilo`. Les détails de ces opérations sont repris ci-dessous.

Pour les noyaux `2.6`, les opérations `make dep ... make modules_install` se réduisent à deux commandes :

```
make
make modules_install
```

43.9.1 Nettoyer et mettre-à-jour.

Le noyau LINUX peut être obtenu à partir de diverses adresses web mais la référence est la page des mainteneurs (LINUX *kernel's home*) sur ftp ://ftp.kernel.-org/pub/linux/kernel.

Le noyau peut être décompressé très simplement (remplacez les `?` par les numéros majeur, mineur et de correctif adéquats) :

```
cd /usr/src
mv linux linux-OLD
tar -xzf linux-2.?.?-test?.tar.gz
mv linux linux-2.?.?-test?
ln -s linux-2.?.?-test? linux
cd linux
```

Eventuellement, appliquez un correctif (*patch*) (voir la section 21.7.3). Dans l'exemple qui suit, pour être aussi explicite que possible, nous avons retenu la numérotation relative à un noyau-`2.4.0`) :

```
bzip2 -cd ../patch-2.4.0-test7.bz2 | patch -s -pl
cd ..
mv linux-2.4.0-test6 linux-2.4.0-test7
ln -sf linux-2.4.0-test7 linux
cd linux
make mrproper
```

L'arborescence des sources du noyau `2.4.0-test6` est à present une arborescence `2.4.0-test7`. Il arrivera régulièrement que vous appliquiez des correctifs au noyau `vanilla` publié par l'équipe de Linus, ne fût-ce que pour y inclure des correctifs de sécurité (voire même des pilotes commerciaux).[3]

Il est très important que les répertoires `include`, cités dans l'encadré, pointent

[3] NdT : la plupart des distributions fournissent des noyaux modifiés.

vers les bons répertoires de l'arborescence du noyau :

```
[root@cericon]# ls -al /usr/include/{linux,asm} /usr/src/linux/include/asm
lrwxrwxrwx  1  root root   24 Sep 4 13 :45 /usr/include/asm ->
                                                ../src/linux/include/asm
lrwxrwxrwx  1  root root   26 Sep 4 13 :44 /usr/include/linux -> /linux/include/linux
lrwxrwxrwx  1  root root    8 Sep 4 13 :45 /usr/src/include/asm -> asm-i386
```

Avant de poursuivre, vous devriez lire le fichier `Changes` (`/usr/src/linux/Documentation/`) pour obtenir les élements requis pour construire un noyau. Si votre arborescence est fournie par votre distribution, la mise-à-jour devrait être complète.

43.9.2 Configurer.

Une arborescence qui a déjà subi de nombreuses modifications pourrait requérir un nettoyage :

```
make mrproper
```

avant de poursuivre. Ceci permet de retrouver un système tel que vous l'obtiendriez juste après la décompression d'un nouveau noyau.

Il existe trois interfaces de configuration (mais l'interface traditionnelle en mode texte `y/n` est particulièrement fastidieuse à employer). La commande :

```
make menuconfig
```

vous affiche une interface semi-graphique. Sous **X** :

```
make xconfig
```

vous permet d'accéder à une interface de configuration graphique. Dans la discussion, nous supposerons que vous avez utiliser `make menuconfig`.

Le programme de configuration vous permet de spécifier une grande quantité de caractéristiques. Il est pertinent de commencer par faire un tour général de manière à avoir une idée de l'organisation de l'arborescence. La plupart des options sont indiquées à l'aide de l'expression `[*]` ou `<*>` pour compiler en dur, `<M>` pour compiler en module ou `[ ]` pour ignorer une option. Vous pouvez aussi désactiver le support des modules depuis l'en-tête `Loadable module support --->`. Il existe aussi une section d'aide pour chaque option ; pour cela sélectionnez `<Help>` en vous déplaçant avec les touches fléchées après vous être rendu sur une option donnée. Le fichier d'aide brut que lit le programme de configuration s'appelle `/usr/src/linux/Documentation/Configure.help` (il représente environ 700 Ko) et mérite d'être consulté.

Quand vous êtes satisfait de votre sélection, choisissez `<Exit>` et `save your kernel configuration`.

La configuration de votre noyau sera sauvegardée dans le fichier `/usr/src/linux/.config`. Lorsque vous exécuterez encore `make menuconfig`, vous repartirez sur la configuration contenue dans `.config`. Le fichier `/usr/src/linux/-`

`arch/i386/defconfig` contient les options par défaut en l'absence du fichier `.config`. Notez que la commande `make mrproper` supprime le fichier `.config`.

43.10 Utilisation des sources empaquetées du noyau.

Votre distribution vous a très vraisemblablement fourni un paquet source prêt à l'emploi. Il vaut mieux employer ce paquet que de télécharger par vous-même les sources d'un noyau car, en effet, le paquet contient diverses options par défaut. Par exemple, RedHat 7.0 comprend le fichier `/usr/src/linux-2.2.16/configd/kernel-2.2.16-i586-smp.config` qui peut être copié dans `/usr/src/linux/.config` pour construire un noyau optimisé pour le SMP (*Symmetric Multiprocessor Support*). Toutes les options par défaut de RedHat s'y trouvent. Il y a ausi un fichier `defconfig` pour construire des noyaux identiques à ceux de RedHat. Enfin, RedHat a appliqué de nombreux correctifs pour inclure des caractéristiques qui, autrement, vous auraient pris énormément de temps pour arriver au même résultat. Il en va de même pour Debian.

Un noyau comme celui de RedHat contient un maximum d'options de manière à supporter de nombreux matériels. La recompilation d'un tel noyau prend du temps. En revanche, un noyau minimaliste est compilé plus rapidement.

43.11 Construction, installation.

Pour construire votre noyau, lancez les commandes suivantes. Selon les options que vous aurez retenues, le processus de compilation prendra de quelques minutes à quelques heures. Lorsque chaque commande est terminée, vérifiez bien les messages retournés à l'écran pour détecter d'éventuelles erreurs (vérifiez aussi le code de retour, `$?`). N'agissez jamais de manière aveugle.

```
make dep && \
make clean && \
make bzImage && \
make modules && \
make modules_install
```

La commande `make modules_install` installera tous les modules dans `/lib/modules/<version>` [il vous arrivera bien de vouloir effacer le contenu de ce répertoire pour relancer la commande `make modules_install` car d'anciens modules provoquent parfois des problèmes avec `depmod -a`].

L'image elle-même du noyau, `/usr/src/linux/arch/i386/boot/bzImage`, et le fichier `/usr/src/linux/System.map` sont deux fichiers produits par la compilation. Ils doivent être copiés dans `/boot/`, éventuellement avec les liens suivants :

```
cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-<version>
cp /usr/src/linux/System.map/ boot/System.map-<version>
ln -sf System.map-<version> /boot/System.map
ln -sf /boot/vmlinuz-<version> vmlinuz
```

Enfin, `lilo.conf` devra être modifié, comme décrit au chapitre 32. A ce stade, beaucoup oublie d'exécuter `lilo`, ce qui rend le système non-amorçable. Exécutez `lilo`, et assurez-vous que vous y avez laissé votre noyau précédent en tant qu'option, au cas où vous devriez y revenir. N'oubliez pas de réaliser une disquette de démarrage, comme cela a été décrit à la section 32.4.

Chapitre 44

Le système X-Window.

Avant l'apparition du *Système X-Window* (*The X Window System*, appelé **X**), UNIX était un système d'exploitation basé sur le mode "console" et ne disposait pas d'un environnement graphique couramment appelé en anglais GUI pour *Graphical User Interface*. **X** a été conçu pour combler ce manque et pour permettre d'utiliser en mode graphique toute la puissance des ordinateurs mis en réseau.

Le développement d'**X** a commencé en 1985 au Massachussetts Institute of Technology par le consortium X et il est maintenant la propriété de l'Open Software Foundation. Le système X-Window représente plus de deux millions de lignes de code **C** et fonctionne sur toutes les variantes d'UNIX.

Vous pourriez imaginer que la mise en oeuvre d'un système graphique sur un écran ne consiste essentiellement qu'en la création d'une bibliothèque utilisateur pouvant réaliser des fonctions telles que le tracé de lignes, de polices de caractères, etc. Pour comprendre pourquoi **X** est bien plus que cela, nous allons d'abord considérer l'exemple des applications fonctionnant sur des terminaux classiques : il s'agit de programmes exécutés sur une machine *distante* alors qu'ils effectuent simultanément un affichage sur un terminal caractère et reçoivent des retours de frappe clavier (ou *feedback*) de ce terminal caractère. Il y a donc deux entités à l'oeuvre : l'application elle-même et l'affichage à l'écran. Or, ces deux applications sont en interaction via une connexion série ou via un réseau. A présent, que se passerait-il si le terminal caractère pouvait aussi afficher des fenêtres et des graphiques en plus du texte, alors que s'effectue un retour vers une application faisant appel à une souris (ou un clavier) ? C'est précisément là qu'intervient **X**.

44.1 Le protocole X.

X est un protocole de commandes échangées entre une application et un terminal graphique spécial appelé serveur **X** (désormais, nous l'appelerons le serveur). [En l'occurence, le terme "serveur" est ambigü, parce qu'il y a de nombreux serveurs X sur chaque machine cliente, et que l'utilisateur se trouve du côté serveur. Classiquement, c'est l'inverse.] Concernant la manière avec laquelle

le serveur effectue des tracés graphiques sur une unité matérielle,[1] l'application doit juste savoir que, si elle envoye une séquence particulière d'octets par TCP/IP, le serveur interprétera ces octets de sorte qu'une ligne, un cercle, une boîte, une police de caractères ou toute autre objet graphique soit affiché à l'écran. En retour, l'application doit savoir qu'aux séquences particulières d'octets correspond l'enfoncement d'une touche d'un clavier ou le déplacement d'une souris. Cette communication TCP est appelée *protocole X*.

Quand vous utilisez **X**, nous ne réalisez pas forcément que l'intéraction décrite ci-dessus a lieu. Le serveur et l'application peuvent se trouver sur la même machine. La puissance réelle d'**X** ressort lorsque l'application et le serveur ne fonctionnent pas sur la même machine. Supposons, par exemple, que 20 utilisateurs soient connectés à un même ordinateur et qu'ils employent des programmes différents dont l'affichage est réalisé sur 20 serveurs **X**. Tout se passe comme si une machine unique permettait l'usage d'écrans et de claviers multiples. C'est pour cette raison qu'**X** est défini comme un *système de fenêtrage transparent en mode réseau*.

Les développeurs d'applications graphiques sont donc dispensés de connaître les propriétés graphiques du matériel (comparons cela avec les applications DOS où chacun doit construire le support de nombreuses cartes graphiques) ; ils ne doivent pas connaître le type de machines sur lesquelles l'application graphique sera affichée.

Le programme qui réalise cette opération miraculeuse se trouve dans `/usr/X11/bin/X`. La séquence typique des événements qui permettent à un programme graphique d'être exécuté est donnée ci-dessous. Il s'agit d'une description simplifiée car, en pratique, de nombreux utilitaires sont mis à contribution.

1. Le programme `/usr/X11R6/bin/X` est démarré et exécuté en tâche de fond. En lisant les fichiers de configuration (`/etc/XF86Config` ou `/etc/X11/XF86Config` sur LINUX), **X** détecte le matériel graphique disponible (par exemple, le type de carte graphique). Il initialise ce matériel en mode graphique.
2. **X** ouvre une connexion *socket* pour écouter les appels entrants sur un port spécifique (généralement, le port TCP 6000) et s'apprête à interpréter toute connexion comme un flux de commandes graphiques.
3. Une application est lancée sur la machine locale ou sur une machine distante. *Tous* les programmes **X** possèdent une option de configuration via laquelle vous pouvez indiquer (grâce à une adresse IP ou un nom d'hôte) où vous souhaitez que le programme se connecte, c'est-à-dire le serveur où vous aimeriez que le résultat soit affiché.
4. L'application ouvre une connexion *socket* à un serveur donné sur le réseau. C'est lors de cette étape que la plupart des erreurs ont lieu. Il peut y avoir échec si le serveur n'est pas en cours d'exécution, s'il a été indiqué incorrectement ou si le serveur refuse une connexion à un hôte qui n'est pas de confiance.
5. L'application commence à envoyer des requêtes de protocole **X**, en attendant qu'elles soient traitées. Après réception, les réponses du protocole **X** sont traitées à leur tour. Aux yeux de l'utilisateur, l'application semble fonctionner sur l'écran du serveur.

[1] Les développeurs d'applications graphiques ne doivent pas se préoccuper de la manière dont le serveur **X** interprète les graphiques.

La communication entre l'application et le serveur est tout de même plus compliquée que le simple tracé de lignes, de rectangle et que la traduction des déplacements de la souris ou de l'enfoncement des touches du clavier. Le serveur doit être capable de traiter de multiples applications qui se sont connectées à partir de nombreuses machines. Ces applications peuvent intéragir (songez aux opérations dites de "copier-coller" entre des applications qui fonctionnent sur différentes machines). Voici quelques exemples de requêtes du protocole **X** de base qu'une application peut entreprendre :

Créer une fenêtre : ouvre une fenêtre (rectangle logique sur l'écran, propriété d'une application) dans laquelle des tracés graphiques peuvent être réalisés.

Afficher des polices : affiche une liste des polices de caractères mises à disposition d'une application.

Allouer de la couleur : permet de définir une couleur par un nom ou une valeur RGB (*Red Green Blue* – Rouge Vert Bleu).

Créer un contexte graphique : un contexte graphique est une définition de la manière dont les graphiques doivent être tracés à l'intérieur d'une fenêtre. Par exemple, la couleur de fond par défaut, le style de ligne, les coupures, les polices de caractères.

Trouver le propriétaire d'une sélection : permet de trouver la fenêtre associée à une sélection (par exemple, une copie de texte). La fenêtre peut appartenir à une autre application.

En retour, le serveur répond en renvoyant des *événements* à l'application. L'application doit constamment écouter le serveur pour capter ces événements. A côté des événements associés au déplacement de la souris ou des entrées au clavier, il y en a d'autres comme, par exemple, l'indication qu'une fenêtre est passée en premier plan (c'est-à-dire par dessus une autre fenêtre. L'application devrait alors envoyer les commandes idoines pour retracer les graphes dans la fenêtre en premier plan). Un autre exemple concerne l'avis de requête d'un collage depuis une autre application lors d'un "copier-coller". Le fichier `/usr/include/X11/Xproto.h` contient les listes complètes des requêtes du protocole **X** et des événements.

Les programmeurs d'applications **X** ne sont pas forcément concernés par ces requêtes. Une bibliothèque de haut-niveau traitent les tâches relatives à l'interaction avec le serveur. Cette bibliothèque est appelée la bibilothèque **X**, `/usr/X11R6/lib/libX11.so.6`.

Une des limitations du protocole **X** tient en ce que les développeurs ne peuvent utiliser que le jeu de commandes qui a été définis. **X** contourne ce problème en rendant le protocole "extensible" depuis le début, c'est-à-dire en permettant que soient ajoutées des extensions ou des améliorations sans compliquer ou affecter la compatibilité. A l'heure actuelle, il existe des extensions à X pour permettre l'affichage 3D sur le serveur, l'interprétation des commandes PostScript, et bien d'autres possibilités qui améliorent l'esthétique et les performances. A chaque extension sont associés des requêtes et des événements du protocole **X** ainsi que des interfaces de bibliothèques de programmation.

Voici un exemple de programme **X** réel. Il s'agit d'un programme très simple qui affiche un fichier image XPM dans une fenêtre et attend qu'une touche soit

enfoncée ou qu'un clic de souris ait lieu pour se terminer. Vous pouvez le compiler avec `gcc -o splash splash.c -lX11 -L/usr/X11R6/lib`. Vous verrez de suite pourquoi il y a peu d'applications écrites directement avec **X**. Notez que toutes les fonctions des bibliothèques **X** sont préfixées avec un X.

```
/* splash.c - display an image */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <X11/Xlib.h>

/* XPM */
static char *graham_splash[] = {
"28 32 16 1"
"  c #34262e", ". c #4c3236", "X c #673a39", "o c #543b44",
"O c #724e4e", "+ c #6a5459", "@ c #6c463c", "# c #92706c",
"$ c #92685f", "% c #987e84", "& c #aa857b", "n c #b2938f",
"= c #bca39b", "- c #a89391", "; c #c4a49e", " : c #c4a8a4",
/* pixels */
"--%#%%nnnn#-nnnnnn=====;;=; :", "--------n-nnnnnn=n==;==;= :; :",
"----n--n--n-n-n-nn=== : : : : : : :", "-----&------nn-n=n==== : : : : : :",
"----------------n===;= : : : : : :", "----%&-%--%##%---n=== : : : : : : :",
"------%#%+++o+++----= : : : : : : :", "--#-%%#+++oo. oo+#--= : : : : : : :",
"-%%%%++++o..    .++&-== : : : : :", "---%#+#+++o.      oo+&n= : : : :",
"--%####+$+++Oo.      o+#- := : :", "-&%##########++Oo       @$-== :",
"####$$$+####$++OX      .O+&==", "&###$O+OXo+++$#+Oo.    ..O&&-",
"&##+OX..... .oOO@@...  o@+&&", "&####$Oo.o++     ..oX@oo@O$&-",
"n####$$$$O$o ...X.. .XXX@$$$&", "nnn##$$#$OO. .XX+@ .XXX@$$#&",
"nnn&&####$OX.X$$@.   XX$$$$&", "nnnnn&&###$$$OX$$X..XXX@O$&n",
"nnnnnn&&%###$$$$@XXXXX@O$&&n", ";n=;nnnn&&&#$$$$$@@@@@@O$&n;",
";n;=nn;nnnn#&$$$@X@O$@@$$&n;", "=n=;;;n;;nn&&&$$$$OO$$$$$&;;",
"n;=n;;=nn&n&&&&&$$$$$###&n;", "n;=;;;;;;;;;&&&n&&&&&&&&&n=;",
";n;n;;=n;&;&;&n&&&&&&&nn;;;", "n;=;;;;;;;;;n;&&n&&&n&n&nnnn;",
"n=;;  :;;=;;nn;&n&nnnnnnn=;", "nn;;;;;;;;;;;;;;;;n&nnnnnn===;",
"=nn;;  :;n;;;;&&&&n&&nnnnnn;=;", "n====;;;;&&&&&&&nnnnnnnnnn;;"
};

int main (int argc, char **argv)
{
   int i, j, x, y, width, height, n_colors;
   XSetWindowAttributes xswa;
   XGCValues gcv;
   Display *display;
   char *display_name = 0;
   int depth = 0;
   Visual *visual;
   Window window;
```

```
   Pixmap pixmap;
   XImage *image;
   Colormap colormap;
   GC gc;
   int bytes_per_pixel;
   unsigned long colors[256];
   unsigned char **p, *q;
   for (i=1; i < argc -1; i++)
      if (argv[i])
         if ( !strcmp (argv[i], "-display"))
            display_name = argv[i+1];
   display = XOpenDisplay (display_name);
   if ( !display) {
     printf ("splash : cannot open display\n");
     exit(1);
   }
   depth = DefaultDepth (display, DefaultScreen (display));
   visual = DefaultVisual (display, DefaultScreen (display));
   p = (unsigned char **) graham_splash;
   q = p[0];
   width = atoi ((const char *) q);
   q = (unsigned char *) strchr (q, ' ');
   height = atoi ((const char *) ++q);
   q = (unsigned char *) strchr (q, ' ');
   n_colors = atoi ((const char *) ++q);

   colormap = DefaultColormap (display, DefaultScreen (display));
   pixmap = XCreatePixmap (display, DefaultRootWindow (display), width, height, depth);
   gc = XCreateGC (display, pixmap, 0, &gcv);
   image = XCreateImage (display, visual, depth, ZPixmap, 0, 0, width, height, 8, 0);
   image->data = (char *) malloc (image->bytes_per_line * height + 16);

/* create color palette */
   for (p = p + 1, i = 0; i < n_colors; p++, i++) {
      XColor c, c1;
      unsigned char *x;
      x = *p + 4;
      if (*x == '#') {
         unsigned char *h = (unsigned char *) "0123456789abcdef";
         x++;
         c.red =
                ((unsigned long) strchr (h, *x++) -
                 (unsigned long) h) << 12;
         c.red |=
                ((unsigned long) strchr (h, *x++) -
                 ((unsigned long) h) <<8;
         c.green =
                ((unsigned long) strchr (h, *x++) -
                 (unsigned long) h) << 12;
```

```
         c.green |=
                ((unsigned long) strchr (h, *x++) -
                 ((unsigned long) h) << 8 ;
         c.blue =
                ((unsigned long) strchr (h, *x++) -
                 (unsigned long) h) << 12 ;
         c.blue |=
                ((unsigned long) strchr (h, *x++) -
                 ((unsigned long) h) << 8 ;
         if ( !XAllocColor (display, colormap, &c))
            printf ("splash : could not allocate color cell\n") ;
      } else {
         if ( !XAllocNamedColor (display, colormap, (char *) x, &c, &c1))
            printf ("splash : could not allocate color cell\n")
      }
      colors[(*p)[0]] = c.pixel ;
}

   bytes_per_pixel = image->bytes_per_line / width ;

/* cope with servers having different byte ordering and depths */
   for (j = 0 ; j < height ; j++, p++) {
      unsigned char *r ;
      unsigned long c ;
      q = image->data + image->bytes_per_line *j ;
      r = *p ;
      if (image->byte_order == MSBFirst) {
         switch (bytes_per_pixel) {
         case 4 :
            for (i = 0 ; i < width ; i++) {
               c = colors[*r++] ;
               *q++ = c >> 24 ;
               *q++ = c >> 16 ;
               *q++ = c >> 8 ;
               *q++ = c ;
            }
            break ;
         case 3 :
            for (i = 0 ; i < width ; i++) {
               c = colors[*r++] ;
               *q++ = c >> 16 ;
               *q++ = c >> 8 ;
               *q++ = c ;
            }
            break ;
         case 2 :
            for (i = 0 ; i < width ; i++) {
               c = colors[*r++] ;
               *q++ = c >> 8 ;
               *q++ = c ;
            }
```

```
            break;
         case 1 :
            for (i = 0; i < width; i++)
               *q++ = colors[*r++];
            break;
         }
      } else {
         switch (bytes_per_pixel) {
         case 4 :
            for (i = 0; i < width; i++) {
               c = colors[*r++];
               *q++ = c;
               *q++ = c >> 8;
               *q++ = c >> 16;
               *q++ = c >> 24;
            }
            break;
         case 3 :
            for (i = 0; i < width; i++) {
               c = colors[*r++];
               *q++ = c;
               *q++ = c >> 8;
               *q++ = c >> 16;
            }
            break;
         case 2 :
            for (i = 0; i < width; i++) {
               c = colors[*r++];
               *q++ = c;
               *q++ = c >> 8;
            }
            break;
         case 1 :
            for (i = 0; i < width; i++)
               *q++ = colors[*r++];
            break;
         }
       }
   }

   XPutImage (display, pixmap, gc, image, 0, 0, 0, 0, width, height);

   x = (DisplayWidth (display, DefaultScreen (display)) - width) / 2;
   y = (DisplayHeight (display, DefaultScreen (display)) - heigth) / 2;

   xswa.colormap = colormap;
   xswa.background_pixmap = pixmap;
   window =
      XCreateWindow (display, DefaultRootWindow (display), x, y, width,
                     height, 0, depth, InputOutput, visual,
```

```
                    CWColormap | CWBackPixmap, &xswa);
   XSelectInput (display, window, KeyPressMask | ButtonPressMask);

   XMapRaised (display, window);

   while (1) {
      XEvent event;
      XNextEvent (display, &event);
      if (event.xany.type == KeyPress || event.xany.type == ButtonPressMask)
         break;
   }
   XUnmapWindow (display, window),
   XCloseDisplay (display);
   return 0;
}
```

Il est possible d'apprendre à programmer avec **X** en utilisant la documentation du *Système X Window* . Voyez ci-dessous. On qualifie le programme que nous venons de voir de "programme écrit directement avec X-lib" parce qu'il est seulement lié à la bibilothèque **X** de plus bas-niveau, `libX11.so`. Les avantages d'un développement selon cette méthode viennent du fait que votre programme fonctionne sur les différentes variantes d'UNIX sans aucune adaptation. Notez que le programme n'a rien à voir avec la résolution (largeur × hauteur ou, nombre de pixels par cm^2), la couleur ou l'architecture matérielle.

44.2 Bibliothèques de *widgets* et bureaux.

La programmation **X** est très ardue. C'est pourquoi, pour la plupart, les développeurs utilisent une bibliothèque de *widgets* de plus haut-niveau.[2]

Les utilisateurs sont bien habitués aux widgets : boutons, menus, boîte de saisie de texte, etc. Les programmeurs **X** doivent construire ces éléments manuellement. Si les widgets ne sont pas incorporés au protocole **X**, c'est pour que plusieurs interfaces utilisateur puissent être développées au-dessus d'**X**. Cette souplesse fait d'**X** une technologie solide.

44.2.1 Aspect fondamentaux.

X Toolkit (`libXt.so`) est une bibliothèque de widgets qui a toujours été fournie librement avec **X**. Elle a une apparence brute par rapport aux normes actuelles. Elle ne prend pas en compte les widgets en 3D (les ombres) [La très bonne application `xfig`, une application basée sur X Toolkit, a été utilisée pour réalisé la majorité des diagrammes de ce livre]. *Motif* (`libM.so`) est une bibliothèque de widgets bien fournie et moderne, qui provient de l'industrie. Cependant, Motif est très volumineuse, lente et elle dépend de **X** Toolkit. Elle a toujours été une bibliothèque propriétaire coûteuse. *Tk* (prononcez "tee-kay", `libtk.so`) est une bibliothèque initialement destinée au langage de scripts *Tcl*.

[2] NdT : en anglais, le terme *widget* désigne un "bidule" (et ce, depuis le début du XXème siècle). En informatique, le terme est basé sur une contraction de *Window* gad*get*. Il s'agit donc d'un élément d'une interface graphique (bouton, menu, ascenseur, etc).

Il s'agit très probablement de la première bibliothèque indépendante des plate-formes (donc, fonctionnant pour UNIX, MS-Windows et Apple Mac). Elle est cependant assez lente et limitée dans ses possibilités (quoique ceci est en train de changer). Ni Tcl, ni Motif n'ont une présentation élégante.

En 1996, approximativement, nous avons vu l'émergence de nombreuses bibliothèques de widgets avec différentes licences. V, xformset graphix viennent à l'esprit immédiatement (c'était à l'époque où l'auteur original commençait à écrire les *coolwidgets*, sa propre bibliothèque de widgets). Il n'y avait pas de bibliothèque de widgets qui, à cette époque, fût libre, à présentation élégante, universelle, efficace et destinée à UNIX. Cette situation a retardé les développement de la Free Software Foundation.

44.2.2 Qt.

A cette époque, une nouvelle bibliothèque GUI (*Graphic User Interface*) a été élaborée. Nommée *Qt*, elle était (et est toujours) developpée par *Troll Tech*. Initialement, non-libre, elle constituait une prouesse technique remarquable du fait qu'elle fonctionnait efficacement et proprement sur diverses plate-formes. Elle a été ignorée par quelques groupes de la communauté du Libre car elle était écrite en C++. Ce langage n'est pas considéré comme la norme en matière de développement par la Free Software Foundation du fait qu'il n'est pas totalement portable (peut-être, existe-t-il d'autres raisons). En réalité, cette bibliothèque n'était libre que pour des applications non-commerciales.

Néanmoins, les partisans de Qt ont commencé à développer le remarquable projet KDE (*K Desktop Environment*) qui comprend un jeu de bibliothèques de développement de plus haut niveau, un gestionnaire de fenêtrage, et beaucoup d'autres applications fondamentales qui constituent le bureau KDE. Les problèmes de licences associés à Qt se sont dissipés et, actuellement, cette bibliothèque est sous licence GPL et sous license propriétaire.

44.2.3 Gtk.

Avant que KDE ne soit suffisamment développé, les programmeurs opposés à Qt avançaient qu'étant donné qu'il y avait plus de lignes de codes de Qt que de KDE, il serait pertinent de développer une bibliothèque *ex nihilo*. La bibliothèque de widgets GTK, écrite spécialement pour le *Gimp* (*GNU Image Manupilation Program*) est sous GPL, entièrement en **C** avec des appels de bas-niveau **X** (donc, sans X Toolkit), orientée-objet, rapide, propre, extensible et proposant une grande gamme de propriétés. Elle incorpore (i) *Glib*, une bibliothèque destinée à étendre le **C** standard, à fournir des fonctions de haut niveau usuellement apparentées aux langages de scripts (comme les tables associatives ou *hash tables* et, les listes), (ii) *Gdk*, une interface de la bibliothèque brute de **X** pour donner des conventions d'attribution de nom GNU à **X** et aussi pour fournir une interface de plus haut niveau à **X** et (iii) *Gtk*, la bibliothèque elle-même.

Le projet *Gnome* (GNU Network Object Model Environment) –basé sur Gtk– s'est développé similairement à KDE, mais en étant entièrement écrit en **C**.

44.2.4 GNUStep.

OpenStep (basé sur NeXTStep) est une spécification d'interface utilisateur graphique publiée en 1994 par Sun Microsystems et NeXT Computers, destinée à construire des applications graphiques. Elle utilise le langage *Objective-C*, extension orientée-objet du **C**. Cela en fait un langage plus adapté à ce type de développements que le C++.

Open Step requiert un moteur d'affichage PostScript semblable au protocole X, mais considéré comme supérieur à X parce que tous les graphiques sont indépendants de la résolution d'écran. En d'autres termes, les écrans à haute résolution permet d'améliorer la qualité des images sans que les graphiques ne rapetissent.

Le projet GNUStep présente un moteur d'affichage PostScript opérationnel et, en tant que logiciel libre, il est destinée à remplacer OpenStep.

44.3 XFree86.

X a été développé par le Consortium X en tant que norme et comme implémentation de cette norme. Il y a des portages pour chaque plate-forme qui supporte le mode graphique. La version courante de cette norme est 11, sous-version 6 (d'où le répertoire `/usr/X11R6/`). Il n'y aura probablement pas d'autre version.

XFree86 sur http ://www.xfree86.org/ est un portage libre d'**X** qui comprend les machines Intel LINUX parmi les matériels supportés. **X** possède certaines caractéristiques dignes d'intérêt pour l'utilisateur Windows, XFree86 en reprend certaines. XFree86 possède son propre système de versions en plus de la notation "X11R6" comme expliqué ci-dessous.

Note de traduction : *au cours de 2004, une modification de la licence sous laquelle XFree86 était publié, a entraîné son abandon par la plupart des distributions GNU/*LINUX. *Le paquet XFree86 est remplacé par le logiciel libre Xorg-x11. La X.Org Foundation fournit désormais une mise en oeuvre libre du* Système X-Window. *Au moment de la traduction, le dernier paquet stable d'`xorg-x11` était la version `6.8.0`. D'un point de vue pratique, le fichier `/etc/X11/XF86Config` (ou `/etc/X11/XF86Config-4`) est remplacé par `/etc/X11/xorg.conf`. La même structure de fichier est conservée.*

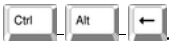
44.3.1 Exécution d'X et conventions pour les touches.

(Voir la section 44.6 pour la configuration d'**X**).

A l'invite du terminal, vous pouvez saisir :

```
X
```

pour démarrer **X**, pourvu qu'il ne fonctionne pas déjà. Si vous avez une configuration adéquate (et que notamment `/usr/X11R6/bin` est dans votre `PATH`), la commande qui précède initiera le matériel graphique et une grille fine en noir et blanc apparaîtra avec un **X** comme curseur central. Contrairement à ce que suggère l'intuition, cela signifie qu'**X** est opérationnel.

– pour tuer **X**, utilisez la séquence de touche 

.

- pour passer vers une autre console, pressez : Ctrl-Alt-F1 ... Ctrl-Alt-F6.
- pour passer vers la console **X**, pressez : Alt-F7. Les sept consoles virtuelles usuelles de LINUX sont des terminaux textes (de 1 à 6) et 7 est un terminal **X** (voir la section 3.7).
- dans une session X, l'agrandissement et la réduction sont respectivement obtenus avec Ctrl-Alt-+ et Ctrl-Alt--.

44.3.2 Exécution des utilitaires d'X.

Le répertoire `/usr/X11R6/bin/` contient un grand nombre d'utilitaires **X** sur lesquels la plupart des autres systèmes d'exploitation ont basé les leurs. Pour la plupart, les noms de ces utilitaires commencent avec un x. Les programmes de base d'XFree86 sont :

Superprobe	iceauth	rstartd	xcmdb	xhost	xmessage
X	ico	scanpci	xconsole	xieperf	xmodmap
XFree86	lbxproxy	sessreg	xcutsel	xinit	xon
Xmark	listres	setxkbmap	xditview	xkbbell	xprop
Xprt	lndir	showfront	xdm	xkbcomp	xrbd
Xwrapper	makepsres	showrgb	xdpiinfo	xkbevd	xrefresh
appres	makestrs	smproxy	xedit	xkbprint	xset
atobm	mergelib	startx	xev	xkbvleds	xsetmode
bdftopcf	mccfm	twm	xeyes	xkbwatch	xsetpointer
beforelight	mkdirhier	viewres	xf86config	xkill	xsetroot
bitmap	mkfontdir	X11perf	xfd	xload	xsm
bmtoa	oclock	x11perfcomp	xfindproxy	xlogo	xstdcmap
dga	pcitweak	xauth	xfontsel	xlsatoms	xterm
editres	proxymngr	xbiff	xfs	xlsclients	xvidtune
fsinfo	resize	xcalc	xfwp	xlsfonts	xwd
fslsfonts	revpath	xclipboard	xgamma	xmag	xwininfo
fstobdf	start	xclock	xgc	xman	xwud

Pour exécuter un programme **X**, vous devez indiquer le serveur distant auquel il faut se connecter. En majorité, les programmes prennent l'option `-display` pour indiquer le serveur **X**. Avec **X** fonctionnant dans la septième console virtuelle, tapez dans votre première console virtuelle :

```
xterm -display localhost :0.0
```

`localhost` définit la machine sur laquelle le serveur **X** est en cours de fonctionnement ; dans ce cas, il s'agit de la vôtre. Le premier `0` correspond à l'écran où nous voulons faire l'affichage (**X** supporte de multiples écrans physiques). Le second `0` représente la fenêtre principale (ou *root window*) sur laquelle nous voulons réaliser l'affichage. Considérons un affichage *multi-sorties* [par exemple, deux moniteurs adjacents pour un seul écran continu] : nous voudrons indiquer sur quel moniteur l'application doit effectuer ses affichages.

Alors qu'`xterm` est en cours d'éxécution, le basculement vers votre session

X montre un terminal caractère où vous saisissez les commandes.

Une meilleure méthode pour spécifier l'écran consiste à utiliser la variable d'environnement `DISPLAY` :

```
DISPLAY=localhost :0.0
export DISPLAY
```

Ceci amène toute application **X** subséquente à faire un affichage vers `localhost:0.0`, bien qu'un `-display` en ligne de commande soit prioritaire.

Les utilitaires **X** affichées ci-dessus sont particulièrement inesthétiques et non-intuitifs. Essayez par exemple, `xclock`, `xcalc`, et `xedit`. Essayez aussi `xbill`. Ensuite, faites :

```
rmp -qa | grep '^x'
```

44.3.3 Exécution de 2 sessions.

Vous pouvez démarrer un second serveur X sur votre machine. La commande :

```
/usr/X11R6/bin/X :1
```

lance une deuxième session **X** dans la console virtuelle 8. Vous pouvez vous y rendre en pressant : 

ou 

.

Il est également possible de démarrer un second serveur **X** dans votre **X** courant :

```
/usr/X11R6/bin/Xnest :1 &
```

Un serveur sera lancé; il sera de plus petite taille et utilisera une sous-fenêtre comme afficheur. Naturellement, il est possible de poursuivre avec un troisième serveur **X** et ainsi de suite.

Pour obtenir une application qui s'affiche sur ce second serveur, utilisez comme précédemment :

```
DISPLAY=localhost :1.0
export DISPLAY
xterm
```

ou

```
xterm -display localhost :1.0
```

44.3.4 Exécution d'un gestionnaire de fenêtres.

Le démarrage manuel d'**X** pour ensuite exécuter une application n'est pas la méthode habituelle d'utilisation d'**X**. Nous souhaitons disposer d'un gestionnaire de fenêtre qui exécute les application proprement. Un bon gestionnaire de

fenêtres est `icewm`, disponible sur *icewm.cjb.net* à l'adresse htpp ://icewm.cjb.net/. Les gestionnaires de fenêtres englobent chaque application dans une fenêtre de taille ajustable avec des boutons tels que  , et  . Par ailleurs, cette fenêtre comporte une barre de tâches, un bouton Start, ce avec quoi vous avez l'habitude de travailler. Un gestionnaire de fenêtre est en fait une autre application **X** qui doit gérer la position des applications **X** de base sur votre bureau. Les exécutables des gestionnaires de fenêtres présentent un suffixe `wm`. Si vous n'avez pas `icewm`, le gestionnaire `twm` (minimaliste et très peu esthétique) sera toujours installé par défaut.

– *Cliquer sur l'arrière-plan* constitue une convention des interfaces utilisateur d'**X**. Les différents boutons de la souris affiche une liste d'actions possibles. Cela est très proche de ce qu'il y a dans le menu du bouton Start.

Une très grande attention est prêtée au gestionnaire de fenêtres. Il doit y avoir une vingtaine de choix à ce jour. Rappelez-vous que des graphiques exagérement chargés provoqueront une certaine irritation au bout de quelques centaines d'heures d'utilisation. Il est à supposer que vous n'aimerez pas que des systèmes graphiques trop gourmands en ressources consomment trop de mémoire.

44.3.5 Contrôle de l'accès à X et affichage à distance.

La manière dont nous avons décrit **X** peut laisser penser que quelqu'un sur l'internet pourrait démarrer une application sur votre écran. Par défaut, **X** interdit l'accès depuis toute les autres machines sauf la vôtre. La commande `xhost` permet l'accès depuis des machines particulières. Par exemple, vous pouvez lancer `xhost +192.168.5.7` pour permettre que l'hôte `192.168.5.7` affiche sur votre machine. La commande `xhost +` désactive totalement l'accès. Une méthode typique consiste à lancer une application sur une machine distante depuis une machine locale. Voici un exemple de session :

```
[psheer@divinian]# xhost +192.168.3.2
192.168.3.2 being added to access control list
[psheer#divinian]# ifconfig | grep inet
        inet addr :192.168.3.1 Bcast :192.168.3.255 Mask :255.255.255.0
        inet addr :127.0.0.1 Mask :255.0.0.0
[psheer@divinian]# telnet 192.168.3.2
Trying 192.168.3.2...
Connected to 192.168.3.2.
Escape character is '^]'.
Debian GNU/Linux 2.2 cericon
cericon login : psheer
Passwd :
Last login : Fri Jul 13 18 :46 :43 2001 from divinian on pts/1
[psheer@cericon]# export DISPLAY=192.168.3.1 :0.0
[psheer@cericon]# nohup rxvt &
[1] 32573
nohup : appending output to 'nohup.out'
[psheer@cericon]# exit
Connection closed by foreign host.
```

44.3.6 Sélections d'X, copier et coller.

Démarrez un `xterm` pour démontrer les opérations que fait la souris.

L'opération de "copier-coller" d'**X** est plus ancienne que celles mises en oeuvre par Windows et Mac. **X** requiert une souris à trois boutons, bien que l'action consistant à presser les deux boutons externes simultanément revient à appuyer sur le bouton central de la souris. [Il faut qu'**X** ait été configuré pour cela –voir l'option `Emulate3Buttons` dans le fichier de configuration donné à titre d'exemple, ci-dessous]. Pratiquons de la manière suivante :

- Pour sélectionner du texte, pressez le bouton gauche de la souris et –en ne le relâchant pas– déplacez la souris sur le texte à sélectionner. Le texte est mis en surbrillance et passe dans un tampon, appelé parfois le *clipboard*.
- L'élargissement ou la réduction de la sélection se fait avec le bouton droit de la souris.
- Un clic sur le bouton central copie le texte à l'endroit où se trouve le curseur.

Les applications modernes de Gtk et Qt ont maintenus la compatibilité avec ces conventions propres à **X**.

44.4 La distribution d'X.

La distribution officielle d'**X** est un paquet de très grande taille disponible au format `tgz` sur http ://www.xfree86.org [puisque `xorg-x11` remplace XFree86 et est sous GPL, signalons que le paquet `xorg-x11` est disponible sur http ://wiki.x.org/wiki. Le paquet `xorg-x11` (nommément `X11R6.8.x`) existe sous forme de paquet `tar.Z`, `.tar.gz` et `.tar.bz2`.]

Le téléchargement et l'installation d'**X** est une entreprise importante, mais vous devriez vous y atteler si vous êtes intéressé par le développement d'**X**.

Toutes les distributions UNIX sont fournies avec une installation compilée (et pratiquement installée) d'**X**. Donc, la version officielle d'**X** n'intéressera généralement que les programmeurs.

44.5 Documentation d'X.

La documentation associée au *Système X Window* représente environ 10 Mo.

44.5.1 Programmation.

Tous les livres qui s'attachent à décrire la programmation des interfaces de programmation d'applications (API – *Application Programming Interface*) sont incluses dans la distribution d'**X**. Pour la plupart, elles sont très spécialisées et ne devraient pas être incorporées par défaut dans votre distribution LINUX ou UNIX. Elles sont accessibles par le téléchargement des sources d'**X**. Pour apprendre à programmer avec X, consultez `xc/doc/specs` (en particulier, `xc/doc/specs/X11`).

Debian contient aussi le paquet `xbooks`, et RedHat, les paquets `XFree86doc`.

44.5.2 Documentation de configuration.

Les répertoires /usr/X11R6/lib/X11/doc/ou /usr/share/xserver-common/ (ou encore /usr/share/doc/xorg-x11-6.8.y) sont très importants pour la configuration d'**X**. Ils contiennent typiquement :

AccelCards.gz	README.MAch64.gz	README.ark.gz	README.neo.gz
Devices.gz	README.NVIDIA.gz	README.ati.gz	README.r128.gz
Monitors.gz	README.Oak.gz	README.chips.gz	README.rendition.gz
QuickStart.doc.gz	README.P9000.gz	README.cirrus.gz	README.trident.gz
README.3DLabs.gz	README.S3.gz	README.clkprog.gz	README.tseng.gz
README.Config.gz	README.S3V.gz	README.cyrix.gz	RELNOTES.gz
README.DGA.gz	README.SiS.gz	README.epson.gz	changelog.Debian.gz
README.Debian	README.Video7.gz	README.fbdev.gz	copyright
README.I128.gz	README.W32.gz	README.gz	examples
README.Linux.gz	README.WstDig.gz	README.i740.gz	xinput.gz
README.MGA.gz	README.agx.gz	README.i810.gz	
README.Mach32.gz	README.apm.gz	README.mouse.gz	

Comme vous pouvez le constater, il existe de la documentation pour de nombreuses cartes graphiques. Le fait de lire Quickstart et de vérifier les caractéristiques de votre carte vous apprendra à configurer X aisément.

44.5.3 Site web d'XFree86 – site web d'Xorg-x11.

Toute documentation qui viendrait à manquer peut être récupérée sur *XFree-86* http ://www.xfree86.org/ ou sur *X.org Foundation* http ://wiki.x.org/wiki/.

Des cartes graphiques sont commercialisées très régulièrement. *XFree86* http-://www.xfree86.org/ ou sur *X.org Foundation* http ://wiki.x.org/wiki/ relatent des FAQ à propos de ces périphériques et fournissent des pilotes au cas où vous ne pourriez réaliser votre configuration avec les informations se trouvant ci-dessous. Avant de rapporter un problème quelconque, vérifiez bien l'information mise à disposition sur ces sites.

44.6 Configuration d'X.

Pour configurer d'**X**, vous devez modifier le fichier /etc/X11/XF86Config (/etc/X11/XFConfig-4 ou /etc/X11/xorg.conf). Ce fichier a peut-être été créer lors de l'installtion, mais sans doute pas de manière correcte. Il est souvent nécessaire de le retoucher à la main pour tirer au mieux parti de votre matériel.

Notez qu'XFree86 (ou xorg-x11) a un fichier de configuration légèrement différent en ce qui concerne les versions 4. Ces différences sont expliquées ci-dessous.

44.6.1 Un simple serveur X à 16 couleurs.

La documentation que nous avons mentionnée ci-dessous est très volumineuse. La manière la plus simple de configurer **X** est de déterminer le type

de souris et de créer un fichier /etc/X11/XF86Config (/etc/XF86Config-4 ou /etc/X11/xorg.conf) (en sauvegardant votre original), de la manière suivante.[3]

Modifiez la section "Pointer" pour les paramètres Device et Protocol. Si vous exécutez la version 3.3 (qui, de nos jours est dépassée), vous devriez aussi commenter la ligne Driver "vga". Vous serez peut-être amené à modifier la ligne contenant 25.275 en 28.32 sur certains portables.

```
Section "Files"
    RgbPath         "/usr/X11R6/lib/rgb"
    FontPath        "/usr/X11R6/lib/X11/fonts/misc"
EndSection
Section "ServerFlags"
EndSection
Section "Keyboard"
    Protocol        "Standard"
    Autorepeat      500 5
    XkbDisable
    XkbKeymap       "xfree86(us)"
EndSection
Section "Pointer"
#   Protocol        "Busmouse"
#   Protocol        "Intellimouse"
#   Protocol        "Logitech"
   Protocol        "Microsoft"
#   Protocol        "MMHitTab"
#   Protocol        "MMSeries"
#   Protocol        "MouseMan"
#   Protocol        "MouseSystems"
#   Protocol        "PS/2"
    Device          "/dev/ttyS0"
#   Device          "/dev/psaux"
    Emulate3Buttons
    Emulate3Timeout 150
EndSection
Section "Monitor"
    Identifier      "My Monitor"
    VendorName      "Unknown"
    ModelName       "Unknown"
    HorizSync       31.5-57.0
    VertRefresh     50-90
#   Modeline        "640x480"  28.32 640 664 760 800 480 491 493 525
```

[3] NdT : avec Xorg, il est possible de créer un fichier de configuration de base selon trois méthodes. (1) Manuellement, en se servant du fichier de référence /etc/X11/xorg.conf.exemple. (2) A l'aide de la commande Xorg -configure. En supposant que le fichier de base ait bien été créé, Xorg devrait afficher qu'il a créé un fichier de configuration /root/xorg.conf.new. Pour tester si la configuration est correcte, il faut exécuter : X -config /root/xorg.conf.new ce qui permet de démarrer le gestionnaire de fenêtres minimaliste twm. (3) Xorg contient un outil de configuration semi-automatique : xorgconfig qui pose des questions sur le matériel et crée un fichier de base /etc/X11/xorg.conf. Dans la dernière étape, les modifications du fichier xorg.conf doivent être réalisées à la main pour améliorer les paramètres.

```
    Modeline       "640x480"  25.175 640 664 760 800 480 491 493 525
EndSection
Section "Device"
    Identifier      "Generic VGA"
    VendorName      "Unknown"
    Boardname       "Unknown"
    Chipset         "generic"
#    Driver          "vga"
    Driver          "vga"
EndSection
Section "Screen"
    Driver         "vga16"
    Device         "Generic VGA"
    Monitor        "My monitor"
    SubSection  "Display"
        Depth          4
        Modes          "640x480"
        Virtual        640 480
    EndSubSection
EndSection
```

Vous pouvez alors démarrer **X.** Pour ce qui concerne la version XFree86 3.3, lancez :

```
/usr/X11R6/bin/XF86_VGA16 -cc 0
```

et pour la version 4, lancez :

```
/usr/X11R6/bin/XFree86 -cc 0
```

Ces deux commandes affichent une ligne d'état contenant `clocks : ...` qui confirme que votre choix de la valeur `25.175` est le bon. Il s'agit de la fréquence en MHz à laquelle l'information relative aux pixels transite depuis votre carte graphique. C'est la seule variable pour configurer un écran à 16 couleurs.
A présent, vous devriez avoir une écran à niveau de gris qui est quasiment utilisable. Il présente l'avantage de *toujours* fonctionner.

44.6.2 Opération de Plug-and-Play.

La version 4 d'XFree possède un support "Plug-and-Play". Effectuez :

```
/usr/X11R6/bin/Xfree86 -configure
```

de manière à produire un fichier `XF86Config` fonctionnel. Vous pouvez copier ce fichier dans `/etc/X11/XF86Config` et lancer le serveur **X**. Néanmoins, ce fichier peut très bien ne pas être optimal. Lisez-le pour l'améliorer.

44.6.3 Configuration d'X.

Une manière simple et sûre de configurer un serveur **X** fonctionnel consiste à suivre les étapes suivantes (si le lancement d'**X** échoue, vous devrez lire la documentation mentionnée ci-dessus). Il y a aussi un outil appelé `Xconfigurator` qui fournit une interface graphique de configuration.

1. Sauvegardez votre fichier `/etc/X11/XF86Config` (ou `/etc/X11/xorg.conf`) vers `/etc/X11/XF86Config.ORIG` (ou `/etc/X11/xorg.conf.ORIG`) .
2. Exécutez `SuperProbe` dans une console caractère. Il va nettoyer votre écran et indquer la carte que vous essayez de configurer. Laissez cette information à l'écran et passez sur un autre terminal. Si `SuperProbe` échoue dans l'identification de votre carte, il en sera de même pour `XFree86`.
3. Exécutez `xf86config`. C'est le script officiel pour la configuration d'**X**. Assurez-vous de toutes les options, sans jouer aux devinettes. Si vous ne disposez pas d'une information plus déterminante, mettez la fréquence horizontale à `31.5`, `35.15`, `35.5`, `Super VGA`...La synchronisation verticale sera fixée à 50-90. Sélectionnez votre carte dans la base de données (vérifiez l'affichage de `SuperProbe`) et vérifiez le serveur **X** que le programme recommande. Il devrait s'agir d'`XF86_SVGA`, `XF86_S3`, `XF86_S3V`, etc. Cela n'a aucune importance d'établir ou non "le lien symbolique", ou de "modifier /etc/X11/Xserver". Notez qu'avec les cartes graphiques PCI modernes, vous n'avez pas besoin du paramètres "RAM DAC". De même concernant "Clockship setting".
4. *Ne* démarrez *pas encore* **X**.
5. La commande `xf86config` devrait vous avoir préparé un fichier `/etc/X11-/XF86Config`. Vous ne devriez plus la ré-exécuter. Vous allez parcourir le fichier et observez qu'il est constitué de sections telles que :

```
Section "<nom_section>"
    <ligne_configuration>
    <ligne_configuration>
    <ligne_configuration>
EndSection
```

Cherchez la section "`Monitor`". Un peu plus bas, vous verrez ces lignes :

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"  25.175 640 664 760 800 480 491 493 525
# 800x600 @ 56 Hz, 35.15 kHz hsync
Modeline "800x600"  36     800 824 894 1024 600 601 603 625
# 1024x768 @ 87 Hz interlaced, 35.5 kHz hsync
Modeline "1024x768"  44.9 1024 1048 1208 1264 768 776 784 817 Interlace
```

Il y a différents paramètres de fréquence pour différents écrans physiques et différentes résolutions. Naguère le choix d'une fréquence trop élevée pouvait conduire à la destruction du moniteur (mais depuis un mécanisme de sécurité prévient ce type d'inconvénient majeur). L'élimination de certaines lignes se fait en les commentant à l'aide du symbole `#` (si vous le souhaitez, n'hésitez pas à sauvegarder votre fichier). Vous pourriez aussi

essayer de laisser les lignes non-commentées de manière à ce qu'X détermine de lui-même les caractéristiques du moniteur mais cela ne fonctionne pas toujours. Il est de loin préférable de choisir explicitement le paramètre `Modelines`.
Si vous ne trouvez pas de "modelines" dans `XF86Config` (ou `xorg.conf`), vous pouvez vous baser sur ceci :

```
Section "Monitor"
   Identifier   "My Monitor"
   VendorName   "Unknown"
   ModelName    "Unknown"
   HorizSynx    "30-40"
   VertRefresh  "50-90"
   Modeline     "320x200"  12.588 320  336  384  400  200 204 205  255 Doublescan
   Modeline     "400x300"  18     400  416  448  512  300 301 302  312 Doublescan
   Modeline     "512x384"  20.160 512  528  592  640  384 385 388  404 -HSync -VSync
   Modeline     "640x480"  25.175 640  664  760  800  480 491 493  525
   Modeline     "800x600"  36     800  824  896  1024 600 601 603  625
   Modeline     "1024x768" 44.9   1024 1048 1208  1264 768 776 784  817 Interlace
EndSection
```

6. Modifiez votre section "`Device`". Vous pouvez faire cela comme suit pour la version XFree86 3.3. Il ne devrait y avoir qu'une seule section "`Device`" :

```
Section "Device"
   Identifier   "My Video Card"
   VendorName   "Unknown"
   BoardName    "Unknown"
   VideoRam     4096
EndSection
```

Concernant la version 4, vous devez ajouter le module du pilote de la carte graphique. Sur un portable avec une carte ati (par exemple), vous aurez :

```
Section "Device"
   Identifier   "My Video Card"
   Driver       "ati"
   VendorName   "Unknown"
   BoardName    "Unknown"
   VideoRam     4096
EndSection
```

Plusieurs options pourraient être ajoutées à la section "Device" pour l'affinage des paramètres de la carte graphique. Voici trois lignes éventuelles :

```
   Option       "no_accel"
   Option       "sw_cursor"
   Option       "no_pixmap cache"
```

qui désactive respectivement l'accélération matérielle, le support matériel du curseur et la mise en tampon "pixmap" de la mémoire video.[4] Ce dernier point se rapporte à l'utilisation de la mémoire non-employée de la carte pour des opérations intermédiaires. Si vous observez des artéfacts ou des problèmes avec votre écran, vous devriez tester ces options.

7. Votre section "`Screen`" devrait contenir les modes indiqués dans la section "`Monitor`". Elle devrait utiliser votre unique section "`Device`" et votre unique section "`Monitor`" : c'est-à-dire, dans l'ordre, "`My video Card`" et "`My Monitor`". Notez que la version 3.3 d'XFree86 ne prend pas en d'option `DefaultDepth`.

```
Section "Screen"
   Identifier   "My Screen"
   Device       "My Video Card"
   Monitor      "My monitor"

   DefaultDepth  16

   SubSection "Display"
      ViewPort     0   0
      Virtual   1024 768
      Depth     16
      Modes     "1024x768" "800x600" "640x480" "512x384" "400x300" "320x240"
   EndSubSection
   SubSection "Display"
      ViewPort     0   0
      Virtual   1024 768
      Depth     24
      Modes     "1024x768" "800x600" "640x480" "512x384" "400x300" "320x240"
   EndSubSection
   SubSection "Display"
      ViewPort     0   0
      Virtual   1024 768
      Depth     8
      Modes     "1024x768" "800x600" "640x480" "512x384" "400x300" "320x240"
   EndSubSection
EndSection
```

8. A ce stade, vous pouvez lancer **X**. En ce qui concerne la version 3.3, il y aura un paquet séparé pour chaque carte vidéo, aussi bien qu'un binaire séparé avec le code du pilote approprié lié statiquement à ce binaire. Le nom de ces binaires est de la forme `/usr/X11R6/bin/XF86_`*`nom-de-carte`*. Les paquets adéquats seront retrouvés avec la commande `dpkg -l 'xserver-*'` dans le cas de Debian et `rpm -qa | grep XFree86` dans le cas de RedHat (ou `RedHat/RMPS/Xfree86-*` sur votre CD-ROM). Vous pourrez alors exécuter :

[4] NdT : `pixmap` est une définition d'image selon un tableau 3D (2 dimensions pour l'image et 1 dimension pour la profondeur de chaque pixel).

```
/usr/X11R6/bin/XFree86-<carte> -bpp 16
```

Cette commande fixe la *profondeur* d'écran à 16, c'est-à-dire le nombre de bits par pixel (ceci peut être traduit en nombre de couleurs).
Concernant la version 4, le support de carte est compilé en tant que module séparé (`/usr/X11R6/lib/modules/drivers/nom-de-carte_drv.o`). Un exécutable unique `/usr/X11R6/bin/XFree86` charge le module approprié basé sur le `Driver` "*nom-de-carte*" dans la section "`Device`". Une fois ce chargement réalisé, vous pouvez exécuter :

```
/usr/X11R6/bin/XFree86
```

La profondeur d'écran est déterminée par la ligne `DefaultDepth 16` dans la section "`Screen`". Vous trouverez le pilote à utiliser en effectuant un `grep` sur les modules et ce, avec le nom de votre carte graphique. Ceci est tout-à-fait similaire à ce que nous avons fait avec les modules du noyau (voir la section 43.6.3).

9. Il est à présent astucieux de créer un script `/etc/X11/X.sh` contenant votre option `-bpp` avec le serveur que vous utilisez. Par exemple :

```
#!/bin/sh
exec /usr/X11R6/bin/<serveur> -bpp 16
```

10. Vous pouvez établir un lien symbolique `/usr/X11R6/bin/X` vers votre script. Il peut s'avérer intéressant de faire un lien symbolique `/etc/X11/X` vers ce script car certaines configurations le rechercheront. Vérifiez à nouveau le fonctionnement d'**X**, en exécutant la commande `X` en ligne de commande.

44.7 Visuels.

44.7.1 RGB et XCMS.

Le texte original a été modifié sur base du document acessible à l'adresse http ://www.liafa.jussieu.fr/~carton/Enseignement/InterfacesGraphiques/Licence-Info/Cours/X/couleurs.html .

Le Système *X Window* utilise deux modes pour représenter les couleurs. Le premier est le format RGB, non portable dans la mesure où la même couleur peut être rendue de manière assez différente par deux écrans. Pour pallier à cet inconvénient, le système XCMS (*X Colors Management System*) a été introduit depuis la version X11R5. Ce dernier système utilise plusieurs modèles de couleurs indépendants des écrans.

Dans le format RGB, une couleur est représentée par ses trois composantes en rouge (*Red*), vert (*Green*) et bleu (*Blue*). Une couleur peut être utilisée par ses trois composantes ou en donnant un nom qui lui est attribué dans le fichier `rgb.txt`. Ce dernier se trouve généralement dans le répertoire `/usr/X11R6/lib-/X11/`. Ce fichier contient une correspondance entre des noms de couleurs et des triplets de composantes RGB en décimal.

Le système XCMS est nettement plus compliqué. Une couleur peut être exprimée dans plusieurs modèles (par exemple : TeKHVC, CIEXYZ, CIEuvY) et dans le modèle RGB. Une couleur est alors écrite en mettant le modèle en préfixe, le caractère " : " puis les valeurs séparées par le caractère "/" (par exemple : CIEuvY :0.152/0.405/0.443, TeKHVC :223.93/72.45/29.67 ou RGB :6a/bb/d8).

44.7.2 Visuels.

La notion de visuel (*visual* en anglais) a été introduite dans **X**. Dans la mémoire du serveur **X**, la couleur de chaque pixel de l'écran est représentée par un nombre entier. Le nombre de bits alloués à cet entier représente la profondeur. Celle-ci est souvent exprimée en nombre de plans et elle détermine le nombre de couleurs possibles à l'écran. L'entier associé à un pixel ne représente pas la couleur directement. En fait, il est utilisé comme une (ou plusieurs) entrée(s) dans une table des couleurs (*colormap*). Cette indirection est utilisée pour augmenter le nombre de couleurs tout en occupant une place raisonnable en mémoire, nécessaire pour tous les pixels de l'écran. Un même serveur **X** présente plusieurs méthodes pour interpréter l'entier associé à une couleur. Chacune de ces méthodes est un "visuel", une méthode (matérielle) qui décrit la manière dont les pixels représentent les couleurs. Les visuels disponibles sur un serveur X peuvent être affichés grâce à l'utilitaire `xdpyinfo`. Chacun des visuels est caractérisé par sa profondeur et sa classe. Pour les écrans "noir et blanc", les classes possibles sont **StaticGray** et **GrayScale**. Pour les écrans "couleur", les classes possibles sont **StaticColor**, **TrueColor**, **PseudoColor** et **DirectColor**. Ces différentes classes se distinguent par le fait d'être en lecture seulement ou en lecture/écriture et par le fait que la table des couleurs soit décomposée ou non.

	Noir et Blanc		Couleur	
	Lecture	Lecture-écriture	Lecture	Lecture-écriture
décomposée			TrueColor	DirectColor
non-decomposée	StaticGray	GrayScale	StaticColor	PseudoColor

A chaque pixel de l'écran est associé une zone mémoire où est enregistré le niveau de gris du pixel ou la couleur. Le nombre de plans donne le nombre de bits consacrés à chaque pixel. Il détermine aussi le nombre maximal de couleurs qui peuvent être présentes simultanément à l'écran. Si le nombre de plan est 1, la couleur d'un pixel ne peut prendre que deux valeurs. C'est le cas d'un écran noir et blanc sans niveau de gris. Au contraire, si le nombre de plans est 24, le nombre de couleurs possibles pour un pixel est $2^{24} = 16.777.216$.

Ce n'est pas directement la couleur qui est mémorisée dans la zone mémoire associée à chaque pixel. En fait, il s'agit d'un point d'entrée dans une table des couleurs où sont vraiment mémorisées les coordonnées (ou valeurs RGB) de la couleur. Nécessaire pour gagner de l'espace mémoire, cette indirection est faite de façon différente selon que la table est décomposée ou non.

Lorsque la table des couleurs n'est pas décomposée, l'entier en mémoire écran donne directement le numéro de l'entrée dans la table des couleurs. Cette entrée est appelée une "cellule". La cellule donne alors les valeurs des trois composantes RGB de la couleur (figure 44.1).

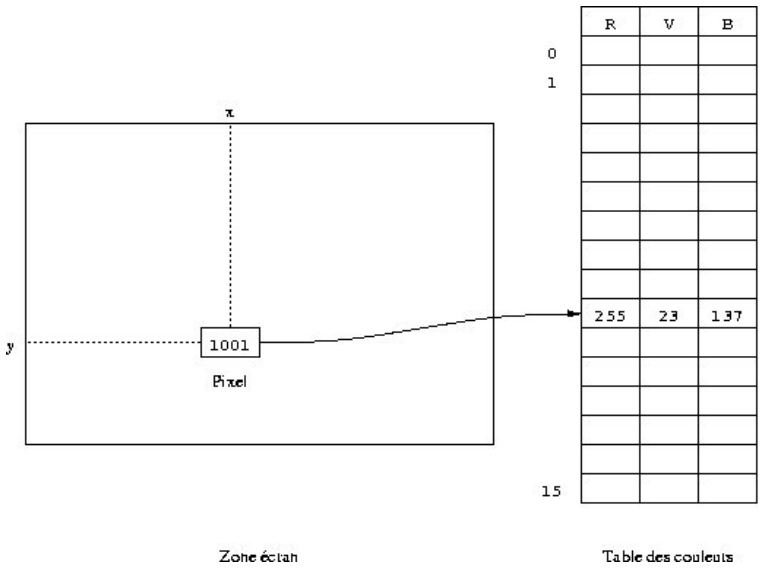


FIG. 44.1 – Table non-décomposée avec 4 plans.

Lorsque la table des couleurs est décomposée, l'entier en mémoire écran se décompose en trois entiers définis par trois champs de bits. Chacun de ces trois entiers donne le numéro d'une entrée dans une des tables des trois composantes (figure 44.2).

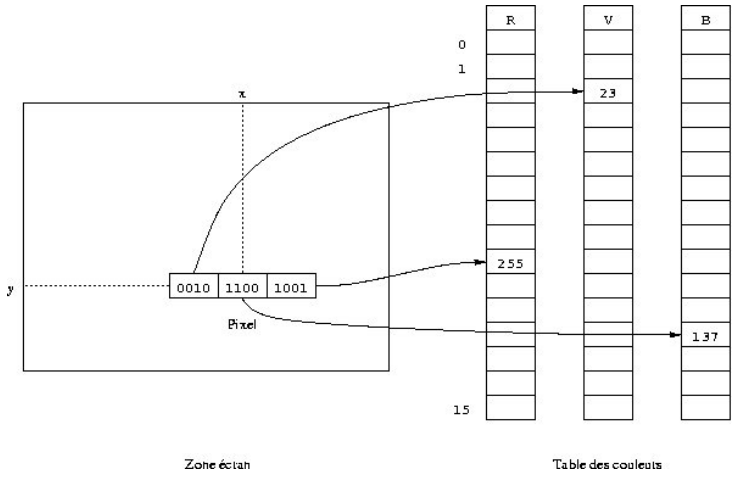


FIG. 44.2 – Table décomposée avec 12 plans.

Qu'un table soit décomposée ou non, les coordonnées des couleurs sont mémorisées avec un nombre de bits plus importants, généralement 12 ou 16. Lorsque la table est décomposée et que les coordonnées sont mémorisées avec 12 bits, on obtient $2^{36} = 68.719.476.736$ couleurs possibles même si toutes ne peuvent pas être affichées simultanément à l'écran.

A présent, vérifiez les visuels que votre écran supporte avec la commande `xdpyinfo`. Vous noterez qu'il y a en plus d'un, car **X** peut effectivement supporter un visuel simple *StaticColor* avec *PseudoColor*, ou un visuel *DirectColor* avec *TrueColor*. Le visuel par défaut est affiché en premier lieu et peut être installé avec l'option `-cc` telle que nous l'avons employée ci-dessus dans le cas d'un serveur à 16 couleurs. L'argument de l'option `-cc` est le numéro de code entre parenthèses.

Notez que les bonnes applications **X** vérifient la liste des visuels disponibles et y sélectionnent un visuel approprié. Il existe aussi des applications qui nécessitent un visuel particulier, et d'autres qui prennent l'option `-visual` en ligne de commandes.

44.8 Les commandes `startx` et `xinit`.

Le démarrage d'un serveur **X** et d'un gestionnaire de fenêtres devrait être automatique. La méthode classique pour démarrer **X** consiste à exécuter la commande `xinit`. Sur LINUX, cettre commande est remplacée par :

```
startx
```

qui est un script exécutant `xinit` après avoir fixé diverses variables d'environnement. Cette commande appelle indirectement plusieurs scripts de configuration dans `/etc/X11/xinit/` et dans le répertoire de l'utilisateur, où ce dernier peut spécifier son gestionnaire de fenêtres et les applications de démarrage. Voir `xinit`(1) et `startx`(1).

44.9 Ecran de connexion.

`init` exécute `mgetty`, qui affiche l'invite de connexion `login :` sur chaque terminal caractère. `init` peut aussi exécuter `xdm` qui affiche un écran de connexion graphique sur chaque serveur **X**. Usuellement, il n'y a qu'un serveur **X**, celui de votre machine.

Les lignes importantes de votre fichier inittab sont :

```
id :5 :initdefault :
```

et

```
x :5 :respawn :/usr/X11R6/bin/xdm -nodaemon
```

qui établissent que le niveau d'exécution (*runlevel*) par défaut est `5` et qu'`xdm` doit être démarré au niveau d'exécution `5`.[5] Ceci ne devrait être tenté que lorsque vous êtes sûr qu'**X** est fonctionnel (en exécutant `X` en ligne de commande). La

[5] NdT : sur la distribution Gentoo-Linux, le démarrage en mode graphique est obtenu (1) en modifiant le fichier `/etc/rc.conf` de manière à ajuster les variables `DISPLAYMANAGER="xdm"` (ou `DISPLAYMANAGER="kdm"` ou encore `DISPLAYMANAGER="gdm"`) et `XSESSION="<gestionnaire>"` (où `<gestionnaire>` est par exemple : Gnome, kde-<version> ou Xsession) et (2) en exécutant la commande `rc-update add xdm default`. Voir `rc-update`(8).

commande `X` tentera de démarrer **X** en désactivant la console. Sur les systèmes basés sur RedHat ou Debian, le mode graphique peut être activé au niveau d'exécution `2` ou `3` (`5` étant réservé à d'autres choses). Dans tous les cas, il devrait y avoir des commentaires dans `/etc/inittab` qui explicite les conventions de la distribution.

44.10 Conventions pour nommer les polices d'X.

Pour la plupart, les applications **X** prennent une option `-fn` ou `-font` pour spécifier la police de caractères. Dans cette section, nous donnons un guide partiel pour l'attribution de nom des polices de caractères.

Un nom de police est une liste de mots et de nombres séparés par des traits d'union. Voici un exemple typique : `-adobe-courier-medium-r-normal--12-120-75-75-m-60-iso8859-1`. Employez la commande `xlsfonts` de manière à obtenir une liste complète des polices de caractères.

Les champs constituant le nom d'une police ont la signification suivante :

adobe le nom d'un fabricant de polices de caractères. Les autres sont :

abisource	b&h	daewoo	gnu	macromedia	monotype	sotfware	urw
adobe	bitstream	dec	isas	microsoft	mutt	sony	xfree86
arabic	cronyx	dtp	jis	misc	schumacher	sun	

courier la famille des polices. Il s'agit du nom réel de la police de caractères. En voici d'autres :

arial	dingbats	lucidux serif	startbats
arial black	fangsong ti	marlett	starmath
arioso	fixed	mincho	symbol
avantgarde	goth	new century schoolbook	tahoma
bitstream charter	gothic	newspaper	tera special
bookman	helmet	nil	terminal
century schoolbook	helmetcondensed	nimbus mono	times
charter	helvetic	nimbus roman	times new roman
chevara	helvetica	nimbus sans	timmons
chevaraoutline	impact	nimbus sans condensed	unifont
clean	lucida	open look cursor	utopia
comic sans ms	lucida console	open look glyph	verdana
conga	lucidabright	palatino	webdings
courier	lucidatypewriter	palladio	wingdings
courier new	lucidux mono	song ti	zapf chancery
cursor	lucidux sans	standard symbols	zapf dingbats

medium L'épaisseur de la police de caractères. Il peut s'agir de `bold`, `demibold`, `regular`.

r indique que la police est de type `r`oman ; `i` vaut pour italique et `o` pour oblique.

normal Il s'agit de la largeur des caractères et de l'espace entre caractères.[6] Il pourrait aussi s'agir de `condensed`, `semi condensed`, `narrow` ou `double`.

[6] NdT : notez qu'en typographie, espace est de genre féminin.

12 représente la taille des pixels. Un zéro indique que la taille d'une police de caratères peut être choisie à toute valeur. La plus grande taille est de 40 points.

120 représente la taille (en dizième) des points d'une imprimante. Usuellement, c'est 10 fois la taille d'un pixel.

75-75 représente la résolution horizontale et verticale avec laquelle la police de caractères a été élaborée. Pour la plupart, les moniteurs actuel sont en 75 pixels par pouce. Les autres valeurs possibles sont `72-72` et `100-100`.

m détermine l'espace entre caractères. Les valeurs sont `m` pour monoespace, `p` pour proportionel ou `c` pour condensé.

60 il s'agit de la largeur moyenne de tous les caractères de la fonte en dizièmes de pixel.

iso8859-1 désigne le jeu de caractères ISO. Dans ce cas, `1` indique qu'il s'agit d'**ISO Latin 1**, un méta-jeu de caractères ASCII. Le dernier bit représente la caractéristique locale.

A titre d'exemple, démarrez `cooledit` avec :

```
cooledit -font '-*-times-medium-r-*--20-*-*-*-p-*-iso8859-1'
cooledit -font '-*-times-medium-r-*--20-*-*-*-p-*'
cooledit -font '-*-helvetica-bold-r-*--14-*-*-*-p-*-iso8859-1'
cooledit -font '-*-helvetica-bold-r-*--14-*-*-*-p-*'
```

Ces commandes invoquent, respectivement, une police utilisée dans la rédaction de quotidiens et une police aisée à lire. Une `*` indique au serveur **X** d'utiliser des valeurs par défaut. De cette manière, vous ne devez pas indiquer une police de caractères particulière.

La commande `xfontsel` est l'utilitaire traditionnel d'**X** pour afficher des polices. La commande `showfont` copie des caractères sous forme de texte ASCII.

44.11 Configuration des polices de caractères.

Les polices de caractères utilisées par **X** sont conventionellement stockées sous `/usr/X11R6/lib/X11/fonts/`. Chaque répertoire contient un fichier `fonts.alias` qui contient une correspondance des noms complets de toutes les polices et des noms simplifiés. De plus, ce fichier contient une liste de polices contenues dans ce répertoire. Pour créer ces fichiers, vous devez descendre (`cd`) dans chaque répertoire et exécuter `mkfontdir` de la manière suivante :

```
mkfontdir -e /usr/X11R6/X11/fonts/encodings -e /usr/X11R6/lib/X11/fonts/encoding/large
```

Vous pouvez exécuter à nouveau cette commande à tout moment pour faire bonne mesure.

Afin d'indiquer à **X** d'utiliser ces répertoires, ajoutez les lignes suivantes à la section "`Files`" de `/etc/X11/XF86Config`, `/etc/X11/XF86Config-4` ou `/etc/X11/xorg.conf`. Typiquement, cette section devrait ressembler à ceci :

```
Section "Files"
     RgbPath    "/usr/X11R6/lib/X11/rgb"
     FontPath   "/usr/X11R6/lib/X11/fonts/misc/ :unscaled"
     FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/ :unscaled"
     FontPath   "/usr/X11R6/lib/X11/fonts/Speedo/"
     FontPath   "/usr/X11R6/lib/X11/fonts/Type1/"
     FontPath   "/usr/X11R6/lib/X11/fonts/misc"
     FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/"
EndSection
```

Il arrivera régulièrement que vous souhaitiez ajouter un répertoire sans vouloir redémarrer **X**. La commande pour ajouter un répertoire au *chemin des polices de caratères* d'**X** est :

```
xset +fp /usr/X11R6/lib/X11/fonts/<nouveau_repertoire>
```

tandis que pour supprimer un répertoire, vous utiliserez :

```
xset -fp /usr/X11R6/lib/X11/fonts/<nouveau_repertoire>
```

Pour donner le chemin vers les polices, utilisez la commande :

```
xset fp= /usr/X11R6/lib/X11/fonts/misc,/usr/X11R6/lib/X11/fonts/75dpi
```

Revenez à la configuration par défaut avec :

```
xset fp default
```

Si vous changez quoique ce soit dans vos répertoires de polices de caractères, vous devriez exécuter :

```
xset -fp rehash
```

r

pour conduire **X** à relire les répertoires de polices.

La commande `chkfontpath` affiche les paramètres du chemin courant des polices. Notez aussi qu'XFree86, version 4, possède un moteur TrueType. Les polices TrueType (`.ttf`) sont fréquentes sous Windows. Il s'agit de polices de taille variable et de haute qualité, destinées aux écrans graphiques. Vous pouvez ajouter un répertoire TrueType à côté des autres répertoires cités ci-dessus et exécuter :

```
ttmkdir fonts.scale
mkfontdir -e /usr/X11R6/lib/X11/fonts/encodings -e /usr/X11R6/X11/lib/fonts/encodings/large
```

dans chacun des répertoires. Notez que `ttmkfdir` est requis pour cataloguer les polices de caractères en tant que polices variables.

44.12 Le serveur de polices de caractères.

Il est désavantageux de stocker sur chaque machine toutes les polices de caractères. Vous pourriez imaginer disposer d'une grande base de données de polices installées sur une machine de sorte que ces dernières soient utilisées par d'autres machines, sur le réseau et à la demande. Vous pourriez aussi utiliser un serveur **X** qui ne supporte pas une police donnée ; s'il pouvait lire des polices depuis le réseau, il ne sera pas nécessaire que ce serveur inclue par lui-même le support de cette police. Le démon `xfs` (**X** *font server*) est un service qui résoud ce problème.

`xfs` lit son fichier de configuration dans `/etc/X11/fs/config` ou `/etc/X11/xfs/config`. Ce fichier contient typiquement :

```
client-limit = 10
clone-self = on
catalogue = /usr/X11R6/lib/X11/fonts/misc :unscaled,
          /usr/X11R6/lib/X11/fonts/75dpi :unscaled,
          /usr/X11R6/lib/X11/fonts/ttf,
          /usr/X11R6/lib/X11/fonts/Speedo,
          /usr/X11R6/lib/X11/fonts/Type1,
          /usr/X11R6/lib/X11/fonts/misc,
          /usr/X11R6/lib/X11/fonts/75dpi
default-point-size = 120
default-resolutions = 75,75,100,100
deferglyphs = 16
use-syslog = on
no-listen = tcp
```

Vous démarrez le serveur en utilisant la commande :

```
/etc/init.d/xfs start
( /etc/rc.d/init.d/xfs start )
```

et vous changez votre chemin dans `/etc/X11/XF86Config` (`/etc/X11/XF86Config-4` ou `/etc/X11/xorg.conf`) pour inclure un jeu minimal de polices :

```
Section "Files"
     RgbPath    "/usr/X11R6/lib/X11/rgb"
     FontPath   "/usr/X11R6/lib/X11/fonts/misc/ :unscaled"
     FontPath   "unix/ :7100"
EndSection
```

Sinon, utilisez `xset` :

```
xset +fp unix/ :7100
```

Remarquez qu'aucune autre machine ne pourra utiliser votre propre serveur de polices en raison de la présence l'option `nolisten = tcp`. Le fait de supprimer cette ligne (et de redémarrer `xfs`) permettra d'utiliser :

```
FontPath "inet/127.0.0.1 :7100"
```

ce qui implique d'avoir une connexion TCP ouverte sur votre serveur de polices, avec les problèmes de sécurité que cela entraîne. Les machines distantes peuvent utiliser la même configuration après que l'adresse IP `127.0.0.1` ait été changée en votre adresse IP.

Enfin, notez que, concernant la version 3.3 d'XFree86 (qui n'a pas de support pour TrueType), le serveur `xfstt` est disponible sur *Fresh Meat* à l'adresse http ://freshmeat.net/.

Chapitre 45

UNIX et la sécurité.

Ce chapitre est sans doute le plus important de tout le livre.[1]

Linux a été décrit à la fois comme le système d'exploitation le plus sécurisé et le plus insécurisé. Ces deux points de vue sont vrais. Si vous ne prêtez aucune attention aux conseils de la communauté Linux, votre serveur sera l'objet d'une attaque. En revanche, si vous adoptez quelques précautions élémentaires, il sera sûr durant de nombreuses années sans maintenance assidue.

L'attitude de la plupart des administrateurs novices est souvent :"Etant donné que Unix est un système si vaste et si complexe et qu'il y a des millions de systèmes Unix sur l'internet, il est peu vraisemblable que *ma* propre machine soit l'objet d'une attaque". Bien sûr, vous n'aurez pratiquement jamais à faire à une personne *ciblant* l'entreprise pour laquelle vous travaillez. Il suffit d'un pirate ayant écrit un scanneur automatique qui essaie de "craquer" chaque ordinateur de votre ville. Il pourrait aussi s'agir de quelqu'un qui, tout en n'étant pas pirate informatique, a simplement téléchargé un utilitaire de piratage. De nombreux experts chevronnés écrivent de tels utilitaires et les distribuent au grand jour. De petits morveux, seulement capables d'exécuter des scripts écrits par d'autres (et qui se vantent d'être "hackers") utilisent ces programmes et font des dégâts considérables. [Il faut absolument distinguer un "hacker" d'un "cracker" ; les premiers sont de véritables programmeurs qui effectuent un travail de programmation ou d'administration de manière positive et avec enthousiasme. Les autres sont des personnes qui, pensant appartenir à la première catégorie, s'ingénient à accéder à des machines sur lesquelles ils n'ont aucun droit.]

Dans le présent chapitre, nous allons décrire une série de méthodes de piratage d'un système Unix. Ainsi, vous aurez une idée de la manière avec laquelle réagir au mieux pour minimiser les risques d'attaques.

45.1 Attaques communes.

Nous pouvons répartir les attaques en deux groupes :(i) les attaques tentées par un utilisateur du système et (ii) les attaques en provenance du réseau (c'est-à-dire externes au système). Si un serveur n'est utilisé que pour le courriel et le web, les connexions au shell peuvent être désactivées *totalement*. Donc, le

[1] NdT : le texte original anglais contient des remerciements à l'attention de Ryan Rubin pour avoir revu ce chapitre.

risque d'attaques du premier type peut être largement réduit. A partir de là, il y a différentes méthodes pour compromettre un système. Nous indiquerons des cas où les problèmes concernent un système multi-utilisateur.

45.1.1 Attaques par dépassement de tampon.

Le terme anglais est *buffer overflow attack(s)*. Considérons le programme **C** dans l'encadré suivant. Si vous ne comprenez pas très bien le **C**, ce n'est pas grave. Ce qui est important, c'est la logique du programme (si vous envisagez d'essayer le programme, déconnectez-vous du réseau) :

```
# include <stdio.h>

void do_echo ( void )
{
    char buf[ 256 ] ;
    gets ( buf ) ;
    printf ( "%s", buf ) ;
    fflush ( stdout ) ;
}

int main ( int argc, char *argv[] )
{
    for ( ; ; ) {
        do_echo () ;
}
```

Compilez ce programme avec la commande `gcc -o /usr/local/sbin/mon_echod mon_echod.c`. A présent, créez un service géré soit par `xinetd` soit par `inetd`. Dans le cas de `xinetd`, créez un fichier `/etc/xinetd.d/mon_echod` contenant :

```
service mon_echod
{
      flags          = REUSE
      socker_type    = stream
      wait           = no
      user           = root
      server         = root
      log_on_failure += USERID
}
```

Concernant `inetd`, vous ajouterez la ligne suivante dans `/etc/inetd.conf` :

```
mon_echod  stream  tcp    nowait   root     /usr/local/sbin/mon_echo
```

Bien sûr, le service `mon_echod` n'existe pas. Il faut donc ajouter la ligne suivante à votre fichier `/etc/services` :

```
mon_echod  400/tcp    # service demo temporaire
```

et redémarrer `xinetd` (ou `inetd`) comme d'habitude.

Maintenant, vous pouvez exécuter `netstat -na`. Vous devriez voir une ligne comme celle-ci en sortie :

```
tcp        0        0.0.0.0 :400        0.0.0.0 :*        LISTEN
```

Lancez la commande `telnet localhost 400` et saisissez quelques commentaires. Le démon `mon_echod` se charge d'afficher simplement les lignes.

Un habitué de la programmation **C** a déjà noté que la saisie de plus de 256 caractères s'effectue dans une zone non-définie de la mémoire. Comment utilisez cette propriété pour *faire fonctionner le programme hors du champ pour lequel il a été prévu* ? Le principe est simple. Il suffit d'écrire des instructions dans une zone de la mémoire pour qu'elles soient exécutées en différé ; ceci va conduire le démon à exécuter tout ce que vous voudrez. Or, le processus fonctionne avec les privilèges de `root`, si bien que quelques instructions envoyées au noyau pourraient conduire à la troncature du fichier `passwd` ou la suppression d'un superbloc. Cette technique qui se base sur l'exécution d'un programme particulier s'appelle, en anglais, un *exploit* (c'est une technique de piratage basée sur une astuce permettant d'exploiter une faille de sécurité dans un programme). En général, ce type d'attaque entre dans la catégorie des *attaques par dépassement de tampon* (ou *buffer overflow attacks*).

La prévention est simple, elle aussi : lorsque vous écrivez des programmes. assurez-vous que les données entrantes sont traitées comme potentiellement dangereuses. Dans le cas qui nous occupe, la fonction `fgets` devrait être employée (à la place de `gets`) parce qu'elle limite le nombre de caractères pouvant être écrits dans le tampon. Cependant, il existe de nombreuses fonctions qui ont un comportement potentiellement dangereux : même `strcpy` écrit jusqu'à un caractère nul (`'\0'`) qui peut ne jamais être encodé. `sprintf` écrit une chaîne qui peut être plus longue que le tampon et `getwd` est une autre fonction[2] qui ne fait jamais de vérification des limites de chaînes.

45.1.2 Programmes setuid.

Un programme tel que `su` doit être *setuid* (voir le chapitre 15). Ce programme doit fonctionner avec les privilèges de `root` afin de basculer les UIDs vers un autre utilisateur. La responsabilité incombre donc à `su` de refuser les privilèges à quiconque n'est pas de confiance. Ainsi, `su` demande un mot de passe et le vérifie le fichier `passwd` avant de procéder plus avant.

A nouveau, la logique du programme est de garantir la sécurité et de fournir une assurance contre les attaques par dépassement de tampon. Si `su` devait présenter une erreur dans la logique d'authentification, cela permettrait à n'importe qui de modifier l'UID pour lequel l'attaquant n'a pas de privilèges.

Les programmes *setuid* devraient donc être considérés d'office comme suspects. La plupart des programmes *setuid* sont de petite taille et simples afin de pouvoir en vérifier la logique de sécurité de manière aussi efficace que possible. Les vulnérabilités peuvent principalement provenir de programmes *setuid*

[2] NdT : `getwd` permet de déterminer le chemin absolu du répertoire courant. Voir man getwd.

volumineux et complexe.

(Cet aspect est d'une importance d'autant plus grande que le nombre d'utilisateurs du système est grand).

45.1.3 Programmes clients sur le réseau.

Considérons maintenant que votre client FTP se connecte à un site distant qui ne soit pas de confiance. Si le serveur du site retourne une réponse telle que "le client FTP ne peut manipuler une réponse trop longue" (exemple d'un dépassement de tampon), ce serveur pourrait permettre à un code malicieux d'être exécuté par votre client FTP au nom du serveur.

En conséquence, il est possible d'exploiter un trou de sécurité dà l'aide d'un programme client en attendant que celui-ci se connecte à un site.

(Ceci concerne tout particulièrement des systèmes auxquels des utilisateurs se connectent).

45.1.4 Vulnérabilité du répertoire `/tmp`.

Lorsqu'un programme crée un fichier temporaire dans votre répertoire `/tmp`, et que le nom de ce fichier est prévisible, il peut être possible de le créer par avance et de le modifier sans que le programme n'en ait connaissance. Les programmes qui créent des fichiers temporaires de manière prévisible ou ceux qui n'établissent pas de manière correcte les droits sur ces fichiers sont susceptibles d'être exploités du point de vue de la sécurité. Par exemple, si un programme fonctionnant au nom du superutilisateur tronque un fichier `/tmp/9260157.TMP` et qu'il est possible de connaître le nom de ce fichier par avance, un pirate peut créer un lien symbolique vers `/etc/passwd`. Il en résultera que le programme tronquera le fichier `passwd`.

(Il s'agit d'un souci dans les systèmes auxquels se connectent de nombreux utilisateurs qui ne sont pas de confiance.)

45.1.5 Problèmes des droits.

Il est facile de voir qu'un utilisateur `jsmith` –s'il n'est pas membre du groupe `admin`– ne peut accéder à un répertoire dont les droits sont fixés à `660` et ayant comme propriétaire `root :admin`. C'est moins évident lorsqu'il y a des milliers de répertoires et des centaines d'utilisateurs et de groupes. Les modalités d'accès sont complexes et il est nécessaire d'écrire des scripts qui établissent les droits de manière sûre. En effet, même un périphérique `/dev/tty*` peut devenir une cible vulnérable.

(Il s'agit d'un souci dans les systèmes auxquels se connectent de nombreux utilisateurs qui ne sont pas de confiance.)

45.1.6 Variables d'environnement.

Il y a de nombreuses façons de créer et de lire des variables d'environnement pour exploiter une vulnérabilité ou pour obtenir des informations susceptibles de compromettre la sécurité. Les variables d'environnement ne devraient jamais contenir d'information secrète comme des mots de passe.

En revanche, lorsqu'ils manipulent des variables d'environnement, les programmes devraient considérer les données qu'elles contiennent comme étant potentiellement malicieuses. Ils devraient donc effectuer un contrôle des limites de chaînes de caractères et une vérification du contenu des variables.

(Il s'agit d'un aspect qui concerne les systèmes auxquels se connectent de nombreux utilisateurs qui ne sont pas de confiance.)

45.1.7 Renifler un mot de passe.

Lorsque `telnet`, `ftp`, `rlogin` ou d'une manière générale, tout programme qui permet l'authentification sur le réseau sans chiffrement sont utilisés, le mot de passe est transmis en clair sur le réseau (*plain text*). Il est donc sous une forme lisible par tous. Il faut décourager l'utilisation de ces programmes et des utilitaires réseau employés par les vieilles versions d'UNIX. Le fait est que des outils élémentaires comme `tcpdump` permettent de lire les mots de passe transmis en clair par ces anciennes commandes réseau (voir la page 308). Aucun de ces services ne devraient donc plus être employés sur un réseau comme l'internet. Cependant, ils peuvent encore être utilisés sur des réseaux LAN, pourvu que ceux-ci soient protégés par un pare-feu et que les utilisateurs du LAN soient de confiance.

45.1.8 Craquer un mot de passe.

Ce point est discuté à la section 12.3.

45.1.9 Attaques par déni de service.

Le terme anglais est *Denial of Service Attack(s)* (ou encore *DoS*). Un déni de service est une attaque qui ne compromet pas le système dans son intégrité mais qui empêche d'autres utilisateurs d'employer légitimement le service bloqué. Cela peut consister à charger répétitivement un service au point que plus personne ne puisse y accéder. Dans chaque cas, les journaux ou les outils de surveillance du trafic permettent de déterminer l'origine du problème. Vous pouvez refuser l'accès grâce à une règle de pare-feu. Il existe beaucoup d'attaques de type DoS qu'il est difficile, voire impossible, de parer.

45.2 Autres types d'attaques.

Ce qui vient d'être vu à la section 45.1 ne constitue en rien une liste exhaustive des attaques communes possibles. Il est toujours étonnant de prendre connaissance de la découverte de nouveaux trous de sécurité dans la logique des programmes. Il n'est pas possible de classer toutes les attaques. Donc, la sécurité doit être en mouvement permanent.

45.3 Contre-mesures.

Une bonne sécurité commence par l'élimination des risques connus, puis des risques potentiels. Ensuite, il faut rendre la vie difficile aux pirates en cumu-

lant des barrières de sécurité UNIX connues. Finalement, il faut s'impliquer de manière active pour contrecarrer des tentatives d'attaques.

45.3.1 Eliminer les risques connus : enlever les paquets anciens.

Il est toujours particulièrement triste de voir des administrateurs naïfs installer des paquets bien connus pour leur(s) vulnérabilité(s) et pour lesquels des logiciels de piratage sont aisément accessibles sur l'internet.

Lorsqu'un trou de sécurité est découvert, le paquet doit être mis-à-jour par la distribution ou par l'auteur du paquet. La liste de diffusion `bugtraq` sur http ://www.securityfocus.com/forums/bugtraq/intro.html annonce les dernières failles. Elles s'adressent à des milliers de membres dans le monde entier. Vous devriez participer à cette liste de diffusion pour connaître les dernières nouvelles.

The Linux Weekly News sur http ://lwn.net est une alternative pour les annonces de sécurité si vous ne voulez avoir qu'un rapport par semaine. Vous pouvez ensuite télécharger et installer le nouveau binaire remplaçant le paquet vulnérable. Consultez les annonces de sécurtié est donc une tâche de la plus haute importance. [Note de Paul Sheer : "Je demande souvent à des administrateurs s'ils ont mis-à-jour le service *xxx* et j'obtiens souvent la réponse suivante : ils ne sont pas sûrs que cela les concerne, ne croyent pas que le paquet est vulnérable, ne savent pas si le service fonctionne, ne savent pas où trouver le paquet ou comment même installer la mise-à-jour. Tout ça comme si, l'ignorance pouvait les absoudre de leurs responsabilités. [...]].

Il en va de même pour les nouveaux systèmes que vous installez. N'y adjoignez jamais de paquets périmés. Certaines distributions livrent des mises-à-jours à leurs CDs ou DVD de base. Vous pouvez donc installer la version de base et ensuite effectuer les mises-à-jour, en fonction d'une liste de mises-à-jour, avant de mettre le système en service. Ainsi, vous pouvez très bien installer une version *x-y* RedHat vieille de six mois et la remettre à jour avec les paquets émis depuis la version *x-y*. Cela reviendra au même que d'installer la version la plus récente de RedHat. En revanche, il existe des distributions mineures qui ne sont pas mises à jour parce qu'il n'y a plus de support. Dans ce cas, veillez à compiler vous-même les paquets vulnérables.

Cependant, au-delà de tout ce qui vient d'être dit, rappelez-vous que les distributions sont parfois lentes à incorporer les alertes de sécurité. En conséquence, ne faites confiance qu'aux alertes de sécurité émises par la communauté et réagissez de vous-même.

45.3.2 Eliminer les risques connus : les paquets compromis.

Les paquets modifiés par un pirate peuvent lui donner accès à une porte dérobée de votre système. C'est ce qu'on appelle communément les *chevaux de Troie* (ou *Trojans* en anglais). Utilisez toujours les commandes de contrôle des paquets telles que discutées dans la sous-section 25.2.6 et vérifiez l'intégrité des paquets logiciels que vous installez.

45.3.3 Eliminer les risques connus : les droits.

Il est facile de localiser des fichiers vulnérables (notamment dans `/dev` et `/tmp`) :

```
find / -perm -2! -type l -ls
```

Les fichiers sans propriétaire donnent une indication d'une mauvaise gestion ou d'une compromission de votre système. Utilisez la commande `find` :

```
find / -nouser -o -nogroup -ls
```

45.3.4 Gestion des mots de passe.

Il va de soi qu'une grande diversité de mots de passe apporte davantage de sécurité. Il est donc toujours bon de ne pas laisser les utilisateurs novices choisir leur propre mot de passe. Créez un programme de production aléatoire de mots de passe incorporant 8 caractères. Vous pouvez utiliser aussi l'utilitaire `pwconv` qui provient du paquet `shadow-utils` pour créer des fichiers de mots de passe cryptés (voir la section 12.3). Consultez `pwconv`(8) pour plus d'information.

45.3.5 Empêcher les services non-sécurisés en soi.

Les services non-sécurisés en soi sont ceux qui permettent à un mot de passe d'être découvert sur l'internet ou qui ne fournissent pas de méthode d'authentification correcte lorsqu'on les appelle. Tout service qui ne chiffre pas le trafic ne devrait être utilisé sur l'internet. Il s'agit de `ftp`, `telnet`, `rlogin`, `uucp`, `imap`, `pop3` et de tout autre service n'utilisant pas de mot de passe chiffré.

Vous devriez plutôt utiliser `ssh` et `scp`. Il s'agit de versions sécurisées de POP et d'IMAP (SPOP3 et SIMAP). Toutefois, il se peut que vous ne trouviez pas un client approprié. Si vous *devez* réellement employer des services non-sécurisés, restreignez les réseaux autorisés à s'y connecter (voir les pages 334 et 338).

Depuis longtemps, les versions d'UNIX sont connues pour l'exportation de partages NFS depuis l'internet (`/etc/exports`), ces partages étant en lecture et en écriture. Le groupe de fonctions pour constituer la table des ports de Sun Microsystem –les paquets `nfs-utils` (`rcp`, ...) et `portmap`– ne procure pas un sentiment de confiance absolue tant s'en faut. Par conséquent, n'utilisez pas NFS sur l'internet.

45.3.6 Eliminer les risques potentiels : le réseau.

Installons `libsafe`. Il s'agit d'une bibliothèque qui interface toutes les fonctions **C** connues pour être vulnérables, ce qui permet de tester les dépassements de tampon lors de chacun de leurs appels. Ensuite, envoyez un courriel à l'administrateur lors des tentatives d'attaques. Rendez-vous à l'adresse http ://www.avyalabs.com/project/libsafe/index.html pour davantage d'information, ou envoyez un courriel à `libsafe@research.avayalabs.com`. La bibliothèque `libsafe` résoud environ 90% des problèmes de dépassement de tampon. Il y a une très légère perte de performances, cependant.

Désactivez tous les services en cours qui ne s'avèrent pas utiles. Ensuite, essayez d'évaluer si les services restants sont absolument nécessaires. Par exemple, avez-vous besoin d'IMAP ou POP3 suffit-il ? IMAP a eu bien davantage d'alertes de sécurtié que POP3, car il s'agit d'un programme plus complexe. Est-il utile de courir le risque de l'employer ?

`xinetd` (ou `inetd`) exécutent de nombreux services, dont seule une partie est réellement nécessaire. Vous devrez donc révisiter `/etc/xinetd.d` (ou `/etc/inetd.conf`) pour y laisser un minimum de services. Concernant `xinetd`, vous pouvez ajouter la ligne `disable = yes` dans le (ou les) fichier(s) adéquat(s). Il ne devrait rester que quelques fichiers fonctionnels. `/etc/inetd.conf` , quant à lui, ne devrait contenir qu'un minimum de lignes. Voici un exemple réel :

```
ftp     stream  tcp   nowait  root    /usr/sbin/tcpd in.ftpd -l -a
pop-3   stream  tcp   nowait  root    /usr/sbin/tcpd ipop3d
imap    stream  tcp   nowait  root    /usr/sbin/tcpd imapd
```

Ce conseil devrait être pris au pied de la lettre. Le principe de base est que si vous ne savez pas le rôle joué par un service, vous devez désactiver ce dernier. Voir aussi la section 30.6.

Dans le cas cité dans l'encadré (et qui est un cas réel), les services ont été restreints afin de n'autoriser que certains réseaux à se connecter (voir les pages 334 et 338).

`xinetd` (ou `inetd`) ne constitue pas le seul problème. Il y a de nombreux autres services, sources de problèmes potentiels. Si vous saisissez la commande `netstat -npl` vous obtiendrez quelque chose comme ceci :

```
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address    Foreign Adress  State PID/Program name
tcp      0      0  0.0.0.0 :25       0.0.0.0 :*      LISTEN 2043/exim
tcp      0      0  0.0.0.0 :400      0.0.0.0 :*      LISTEN 32582/xinetd
tcp      0      0  0.0.0.0 :21       0.0.0.0 :*      LISTEN 32582/xinetd
tcp      0      0  172.23.80.52 :53  0.0.0.0 :*      LISTEN 30604/named
tcp      0      0  127.0.0.1 :53     0.0.0.0 :*      LISTEN 30604/named
tcp      0      0  0.0.0.0 :6000     0.0.0.0 :*      LISTEN 583/X
tcp      0      0  0.0.0.0 :515      0.0.0.0 :*      LISTEN 446/
tcp      0      0  0.0.0.0 :22       0.0.0.0 :*      LISTEN 424/sshd
ucp      0      0  0.0.0.0 :1045     0.0.0.0 :*             30604/named
ucp      0      0  172.23.80.52 :53  0.0.0.0 :*             30604/named
ucp      0      0  127.0.0.1 :53     0.0.0.0 :*             30604/named
raw      0      0  0.0.0.0 :1        0.0.0.0 :*             -
raw      0      0  0.0.0.0 :6        0.0.0.0 :*             -
```

Cependant, cela ne vous indique pas que le PID `446` correspond à `lpd`. Pour obtenir cette information, tapez `ls -al /proc/446/`.

Du tableau, vous pouvez déduire que sont ouverts les ports `1`, `6`, `21`, `22`, `25`, `53`, `400`, `515`, `1045` et `6000`. Les ports de `1` à `6` sont des ports du noyau, `21` et `400` correspondent, respectivement, à FTP et à notre démon `echo` (voir

la sous-section 45.1.1). Le grand nombre de ports ainsi laissés actifs prêtent le flanc à une attaque.

A ce stade, vous devriez vérifier chaque service pour :

1. décider de sa nécessité,
2. vous assurez que vous disposez de la dernière version du paquet,
3. et enfin, consultez la documentation des paquets de manière à restreindre l'accès à vos ports pour certains réseaux qui seront autorisés à s'y connecter.

Il est intéressant de noter que beaucoup de personnes se livrent à des supputations du genre : "Ce service est si populaire qu'il est difficile d'imaginer qu'il soit vulnérable". C'est exactement l'inverse. Lorsqu'un paquet est ésotérique ou peu utilisé, il est bien plus probable que personne ne se soit attaché à en trouver une vulnérabilité. Dans le cas de `named` (c'est-à-dire `bind`), de nombreuses failles ont été rendues publiques notamment pour ce qui concerne les versions antérieurs à `9`. Donc, la mise-à-jour vers la dernière version (`9.2.2`, au moment de la traduction) est un acte de prudence essentiel à adopter pour toutes les machines que vous administrez.

45.3.7 Eliminer les risques potentiels : les programmes setuid.

La commande suivante permet de trouver les programmes setuid de votre système :

```
find / -type f -perm +6000 -ls
```

Leur désactivation est tout aussi simple :

```
chmod -s /bin/ping
```

Il n'y a rien d'anormal à priver un simple utilisateur de la commande `ping`. Par ailleurs, si vous n'autorisez aucune connexion à un shell sur votre système, vous pouvez éliminer les droits setuid de toutes les commandes du shell.

45.3.8 Rendre la vie difficile aux pirates.

De nombreuses actions peuvent être engagées, qui ne relève pas de la sécurité à strictement parler mais, qui gêneront considérablement un pirate pour le décourager d'entreprendre une attaque classique courante, et ce, même si votre système est vulnérable. Une tentative de piratage est toujours conçue pour un système configuré d'une certaine manière. En éloignant la configuration de votre système d'une configuration standard, vous allongez le temps de vie de votre système.

Partitions en lecture seule : il est permis de monter la partition `/usr` (et les répertoires de haut niveau critiques comme `/bin`) en lecture-seule puisque par définition, il s'agit de données statiques. Naturellement, toute personne qui a les droits de `root` pourra les remonter en lecture-écriture, mais un script d'attaque générique sera impuissant. Certains disques SCSI peuvent

être configurés pour être montés en lecture-seule car ils utilisent des micro-interrupteurs DIP ou DIP switches (*Dual In-line Pin Switch*).[3] La partition `/usr` peut être construite comme une partition ISO9660 (système de fichiers CD-ROM) qui, par définition, est en lecture-seule. Vous pouvez d'ailleurs monter vos CD-ROMs comme une partition `/usr`. L'accès sera ralenti, mais la partition sera absolument inmodifiable. Finalement, il vous est possible de modifier le code noyau pour faire échouer les tentatives de montage en écriture de `/usr`.

Attributs en lecture seule : LINUX possède des attributs spéciaux de fichiers pour rendre un fichier non-modifiable quels que soient les droits usuels. Ces attributs sont contrôlés par les commandes `lsattr` et `chattr`. Vous pouvez aussi créer un fichier de journalisation en mode `a`jout-seul (`a`ppend-only) avec `chatter +a /var/log/messages /var/log/syslog` ou rendre les fichiers immuables à l'aide de `chatter +i /bin/login`. Il s'agit là de deux bonnes idées à suivre. La commande :

```
chattr -R +i /bin /boot /lib /sbin /usr
```

est encore meilleure. Naturellement, un utilisateur qui s'emparerait des privilèges de superutilisateur pourra restaurer le système à sa guise.

Contrôle périodique du système : il est utile d'écrire vos propres scripts `crond` pour vérifier les fichiers qui sont modifiés sur votre système. Les scripts devraient contrôler les nouveaux programmes setuid, les droits et les modifications ayant eu lieu dans les fichiers binaires. Par ailleurs, n'hésitez pas à réattribuer des droits qui vous paraissent mieux adaptés. Gardez à l'esprit que les programmes `cron` peuvent être modifiés par quiconque pirate votre système. La simple commande :

```
find / -mtime 2 -o -ctime 2
```

recherche tous les fichiers qui ont été modifiés durant les deux derniers jours.

Paquets non-standards : lorsque vous notez beaucoup d'alertes de sécurité pour un paquet donné, passez à un paquet concurrent. Il y a des alternatives à `bind`, `wu-ftpd`, `sendmail` (toutes ont été vues au chapitre 31). Il en est de même pour chaque service, pratiquement. Passez complétement à FreeBSD, OpenBSD ou NetBSD réduit aussi les risques.

Messages non-standards : de nombreux services fournissent des messages d'information qui donnent la version de votre logiciel. Par exemple, les serveurs de courriel ont des réponses, par défaut, à `HELO` qu'ils utilisent pour eux-mêmes. Les connexions et les bannières FTP affichent souvent le type de système d'exploitation d'un serveur. N'hésitez pas à modifier ces messages pour qu'ils en disent le moins possible car un attaquant tire toujours des conclusions de ces données. Commencez par modifier `/etc/motd`.

[3] NdT : Il s'agt de micro-interrupteurs (ou micro-commutateurs) permettant de paramétrer la configuration d'un appareil : carte mère, carte d'extension (vidéo, modem, imprimante, etc), disques, etc. Ces dispositifs ne sont pratiquement plus utilisés de nos jours car la configuration des périphériques se fait par voie logicielle.

Noyaux minimalistes : il est aisé de compiler votre noyau sans support de modules, avec un jeu d'options minimales. Par le passé, le chargement de modules contenant des portes dérobées a été identifié comme une source de vulnérabilités. Des noyaux légers sont toujours plus sûrs.

Architectures non-Intel : les pirates doivent apprendre le langage *assembleur* pour exploiter les nombreuses vulnérabilités. L'*assembleur* le plus commun est celui des processeurs Intel 80?86. L'usage de processeurs non-Intel en déroutera plus d'un.

Projet OpenWall : Ce projet consiste à fournir un correctif du noyau qui rend la pile d'un processus non-exécutable (ceci diminue considérablement les tentatives de dépassement de tampon) et qui effectue d'autres traitements intéressants sur le répertoire `/tmp` et les processus d'E/S.

45.3.9 Modèles de sécurité usuels.

Les pirates ont des ressources limitées. L'investissement que requiert un piratage doit être comparé aux chances de succès de celui-ci. On pourrait représenter cela par un rapport "investissement du pirate / chance de succès". Si vous sentez que ce rapport est bas pour la machine que vous administrez, vous devrez vous investir davantage et envisager des méthodes comme celles décrites ci-dessous. En poussant à l'extrême, il est possible de si bien sécuriser un système LINUX que le budget de la Défense d'un gouvernement ne suffirait pas à le craquer.

Privilèges (*capabilities*) : il s'agit d'une approche sécuritaire qui donne un accès superutilisateur limité aux programmes qui, normallement, doivent agir comme véritables exécutables à setuid `root`. En effet, en majorité, les processus exécutés avec les privilèges de `root` (setuid) fonctionnent précisément avec ces privilèges parce qu'ils doivent accèder à une seule fonction privilégiée. Par exemple, le programme `ping` nécessite les droits de superutilisateur (lancez `ls -l /bin/ping` et vous constaterez la présence d'un bit setuid). Les "*capabilities*" constituent un jeu de privilèges réglables finement qui indiquent à un processus qu'il peut agir là où un utilisateur normal ne pourrait le faire, *sans* avoir un plein accès `root`. Dans le cas de la commande `ping`, le *privilège* serait un type d'accès au réseau que seul `root` est normalement en droit de pratiquer.

Listes de contrôle d'accès : ces listes étendent les droits simples "utilisateur/groupe/autres" des fichiers UNIX de manière à permettre à certains groupes d'utilisateurs d'accéder à certains fichiers. Ces listes n'agissent pas sur la sécurité du réseau mais elles sont utiles lorsqu'il y a de nombreux utilisateurs sur votre système et que vous voulez limiter leurs actions (une ACL[4] n'a pas vraiment sa place dans les listes dont nous venons de discuter).

DTE : Il s'agit de l'acronyme pour *Domain and Type Enforcement*. Quel en est le fonctionnement : quand un programme est exécuté, il entre dans une catégorie et il n'est autorisé à réaliser que certaines opérations, même lorsqu'il est exécuté au nom de `root`. Ces restrictions de comportement sont étendues aux processus-fils que ce programme peut engendrer. Il s'agit

[4] NdT : ACL est l'acronyme d'*Access Control List*. Il s'agit d'une liste des personnes ayant le droit d'accéder à une ressource (un fichier, un réseau ...).

d'une méthode de sécurité *réelle* et il existe des correctifs du noyau pour accomplir ces restrictions de comportement. La NSA (*National Security Agency* aux Etats-Unis) utilise une distribution LINUX construite avec le DTE.

meduse : il s'agit d'un système de sécurité qui amène le noyau à interroger un démon utilisateur avant qu'un processus quelconque n'entreprenne une action. Il est omniprésent car il est entièrement configurable au point que les restrictions peuvent porter sur tout ce que vous souhaitez.

VXE : Il s'agit de l'acronyme pour *Virtual eXecuting Environment*. Cet environnement impose à un programme d'être exécuté dans son propre espace (protégé) pendant que VXE exécute un programme Lisp pour vérifier qu'un appel système est autorisé. *VXE* est très semblable à *medusa*.

MAC : *Mandatory Access Controls*. Ce système de sécurité traite aussi de l'environnement virtuel des processus. MAC est une norme POSIX.

RSBAC et RBAC : *Rule-Set-Based Access Controls* et *Role-Based Access Controls*. Il s'agit principalement d'une combinaison des systèmes présentés ci-dessus.

LIDS : *Linux Intrusion Detection System* prend de simples mesures préventives pour empêcher le chargement de modules, les modifications de fichiers et l'affichage d'informations relatives aux processus.

Des correctifs existent aussi pour le noyau afin de mettre en oeuvre les procédures qui précèdent. La plupart de ces projets sont matûres désormais mais, pour autant, ils ne sont pas systématiquement retenus dans les noyaux, probablement parce que les développeurs ne sont pas certains de ce qui doit être l'approche idéale en termes de sécurité UNIX. Tous les projets ont un point commun : la double vérification des processus privilégiés, ce qui –en soi– est une très bonne chose.

45.3.10 Détections des intrusions.

Nous entendons par ce terme (i) la gestion des attaques et la réponse à y apporter ainsi que (ii) la gestion des intrusions (attaques réussies) et la réponse à y apporter. Les utilitaires qui permettent d'avoir un comportement réactif appartiennent à la classe des logiciels de *détection d'intrusions réseau*. L'idée de base est qu'il est possible de détecter une instrusion et de réagir... *mais bien avant cela*, il faut admettre, au-delà de tout caractère émotionnel, que si une attaque se prépare ou a eu lieu, le système est ou était insécurisé. Or, il faut réaliser impérativement qu'il est stupide d'installer des systèmes anti-intrusion tant que les plus élémentaires mesures de sécurité ne sont même pas mises en oeuvre. Par conséquent, il faut absolument appliquer les mesures discutées précédemment avant de penser aux contrôles anti-intrusion.

Pour se faire une image du contrôle d'intrusion la plus élémentaire, considérons les élements suivants. Pour pirater un système, il est nécessaire de déterminer les ports actifs. A cette fin, le pirate essaie de se connecter à chaque port pour déterminer ceux qui sont ouverts. C'est la technique de *balayage de ports* (ou *ports scan* en anglais). Il existe des outils très simples pour détecter un balayage des ports, à commencer par une règle de pare-feu qui refuse à l'attaquant tout accès ultérieur. Notez que ceci peut se retourner contre vous si

l'attaquant a réussi à usurper votre adresse IP (*IP address spoofing* ou usurpation d'adresse IP). Cependant, ce qui est important, c'est que les outils de détection de balayage de ports rapportent les adresses IP qui servent à préparer l'attaque. Donc, une requête inverse vous donnera le nom de domaine et une requête `whois` sur le site d'enregistrement DNS authoritaire adéquat vous révèlera l'adresse physique et le numéro de téléphone du serveur du domaine.

Le contrôle de balayage de ports est la forme la plus élémentaire de surveillance et de réactivité que vous puissiez mettre en oeuvre. A partir d'ici, vous trouverez toute sorte d'outils plus ou moins curieux ou astucieux qui vous rapporteront l'activité du réseau et des processus. Nous vous laissons à votre propre recherche sur ce point, quoique vous pourriez commencer avec *Snort traffic scanner* htpp ://www.snort.org, le *Tripwire intrusion detection system* http ://www.tripwiresecurity.com et *IDSA* http ://jade.cs.uct.ac.za/idsa.

La surveillance des ports est bien sûr un élément dissuasif pour les pirates. Un réseau devrait être capable de trouver l'origine des attaques et traquer à leur tour les attaquants. La menace d'être découvert fait du piratage un passe-temps bien moins attirant. Il ne vous reste plus alors qu'à connaître les recours légaux que vous pouvez engager à l'endroit de ceux qui tentent de compromettre votre système.

45.4 Lectures importantes.

Ce qui précède constitue, en fait, un guide pratique. Cependant, la sécurité est un espace d'un très grand intérêt, en soi. Un point de départ pour approfondir le sujet peut être la FAQ de `comp.os.linux.security`. Cette FAQ fournit les références de sécurité UNIX les plus importantes disponibles sur l'internet. Vous pouvez la télécharger depuis http ://www.memeticcandiru.com/colsfaq.html, http-://www.linuxsecurity.com/docs/colsfaq.html ou http ://www.geocities.com/swan-_daniel/colsfaq.html. La page web *Linux Security* http ://www.linuxsecurity.com/ présente un sommaire de référence relatif à la sécurité, qui synthétise en deux pages les points essentiels que vous devez connaître.

45.5 Jeu-concours rapide pour tester votre gestion de la sécurité.

- *Combien de rapport de sécurité avez-vous déjà reçu ?*
- *Combien de paquet avez-vous mis-à-jour en raison de vulnérabilités ?*
- *Combien de services avez-vous désactivez parce que vous n'êtes pas sûr de leur état de sécurité ?*
- *Combien de règles d'accès avez-vous dans vos services `hosts.*/xinetd` ?*

Si, en cumulant vos réponses, vous arrivez à moins que 5, vous n'êtes pas très consentieux en terme de sécurité.

45.6 Vérifications de la sécurité.

Ce chapitre concerne presqu'exclusivement la sécurité de votre serveur LINUX. Cependant, si vous administrez un grand réseau, le contrôle de la sécurité

(*security auditing*) va devenir une entreprise beaucoup plus coûteuse. Le contrôle de la sécurité devient une tâche vraiment complexe lorsque différents administrateurs gèrent des plate-formes différentes sur un grand réseau. Des entreprises sont spécialisées dans ce travail. Si une personne aguerrie aux problèmes de sécurité n'est pas affectée à un réseau, il faudra consacrer un budget important pour louer les services d'une telle entreprise.

Le contrôle de votre réseau impliquera :

- la réalisation de tests de pénétration des pare-feux,
- le balayage de ports,
- l'installation des outils de détection d'intrusions,
- l'analyse des voies d'attaques possibles par l'internet,
- l'évaluation de l'accès aux services depuis votre réseau LAN local,
- l'enregistrement des activités de gestion des adminsitrateurs,
- les tentatives pour craquer les mots de passe de tous les services d'authentification,
- le contrôle de l'activité des comptes utilisateurs légitimes.

Les attaques effectuées par le réseau coûtent des milliards de dollars par an en durée d'arrêt et en réparations. Oublier la sécurité est une fausse économie.

Chapitre 46

Programme de cours.

Nous reproduisons ici la méthode de cours utilisée par Paul Sheer en Afrique du Sud.

Les sections constituant cette annexe décrivent un programme de cours de 36 heures sous forme de 12 séances de 3 heures. A raison de 2 séances par semaines, le cours s'étend sur 6 semaines. Les cours sont donnés en suivant le texte de ce livre de très près, mais certains chapitres font l'objet d'un travail personnel à domicile.

46.1 Exigences matérielles.

Il est nécessaire que les personnes qui suivent le cours disposent d'un système LINUX pour leurs travaux à domicile. [...]

La classe accueille de 4 à 10 personnes. Il est impératif que les étudiants disposent d'une machine pour pouvoir saisir des commandes et interpréter leur résultat. Il s'agit donc d'un cours *interactif* par opposition à un cours *ex cathedra*. L'enseignant ne doit pas nécessairement disposer d'une machine sauf s'il envisage d'utiliser un projecteur de document Impress (OpenOffice.org). Un rétro-projecteur et un tableau suffisent. Les ordinateurs devraient être mis en réseau local, et connectés sur l'internet. Une installation Linux complète sera préférée dans la mesure où tous les logiciels mentionnés dans le livre doivent être installés (tous les services, les différents types bureaux graphiques, les sources du noyau et les paquets associés au langage **C**).

Les CDs d'une distribution LINUX devraient être disponibles afin que les étudiants puissent installer LINUX sur leur machine à domicile.

Chaque étudiant devrait disposer du livre.

46.2 Sélection des candidats.

Le programme des cours est destiné à des administrateurs du système MS-Windows, à des personnes ayant des notions de programmations, ou des personnes ayant une expérience dans l'installation de matériel informatique et dans l'installation de systèmes d'exploitation. Les utilisateurs n'ayant pas de notions de ce qu'est un shell, ne sachant pas monter un PC ou établir une connexion

réseau suivront un cours moins intensif et ne devraient pas être confrontés à l'abstraction du shell.

Bien sûr, des personnes alertes suivront le cours avec facilité même si elles n'ont pas d'expérience en informatique. Par ailleurs, l'homogénéité d'une classe facilite l'apprentissage. [...].

Un programme de cours moins poussé consiste à ne couvrir qu'une moitié du livre environ, tout en personnalisant davantage l'apprentissage.

46.3 Style de cours.

Les leçons durent 3 heures et il est particulièrement important de les espacer de quelques jours pour permettre l'assimilation des notions et pour permettre que les étudiants pratiquent d'eux-mêmes durant l'intervalle.

Le cours se passe avec des instructions du type : "à présent, tapez ceci ...". Le cours est émaillé de nombreux exemples et il devrait suivre l'ordre des chapitres du livre. Parfois, certaines répétitions sont évitées. Des variantes au exercices du livre seront introduites.

Le *leitmotiv* du cours est de *laisser les étudiants saisir leurs commandes et analyser ce que leur retourne le système d'exploitation.*

L'enseignant doit conserver à l'esprit que les étudiants peuvent "coincer". Il faut donc vérifier l'écran de ceux chez qui cela arrive et éventuellement corriger leur commande pour ne pas laisser la classe se dissiper.

46.4 Leçon 1.

Un historique d'UNIX et de LINUX est donné de manière à décrire les personnes et les organismes faisant autorité. Les diverses formes de licences sont expliquées avec une attention tout particulière à la GPL.

Le chapitre 5 (*Les commandes de base*) occupe le reste des trois premières heures.

Travail à domicile : Les chapitres 49 (*FAQ-Linux*) et 50 (*La licence publique GPL*) seront lus. Les étudiants installeront leur distribution LINUX. Le chapitre 7 devrait être vu pour apprendre les bases de `vi`.

46.5 Leçon 2.

Le chapitre 6 (*Les expressions rationnelles*) occupe la première heure, le chapitre 8 (*Les scripts du shell*) le reste de la leçon. L'enseignant devrait s'apensentir sur l'importance qu'il y a à bien comprendre les expressions rationnelles et leur usage dans le monde UNIX.

Travail à domicile : rechercher différentes configurations du bureau et des applications pour l'utilisateur. Les étudiants devraient devenir coutumiers des différents types de bureaux et des applications majeures qu'ils offrent.

46.6 Leçon 3.

La première heure est consacrée au chapitre 9 (*Flux et sed comme éditeur de flux*). La deuxième traite des chapitres 10 (*Processus et variables d'environnement*) et 11 (*Courriel*). Le chapitre 12 (*Comptes d'utilisateurs et droits*) est vu dans la troisième heure.

Travail à domicile : rechercher LINUX sur internet. Toutes les ressources mentionnées aux chapitres 14 (*Ressources* LINUX) et 17 (*Documentation pré-installée*) devraient être parcourues.

46.7 Leçon 4.

Les deux premières heures traitent des chapitres 13 (*Utilisation des services internet*), 14 (*Ressources* LINUX), 15 (*Droits et types de temps*) et 16 (*Liens symboliques et physiques*). Durant la troisième heure, les chapitres 17 (*Documentation pré-installée*) et 18 (*Survol de la structure des répertoires* UNIX) sont vus.

Travail à domicile : Les chapitres 19 à 22 (c'est-dire, dans l'ordre : *Les périphériques d'*UNIX ; *Partitions, systèmes de fichiers, formatage et montage* ; *Script de shell avancés* ; *Services du système et lpd*) seront lus. Les étudiants ne seront pas à même de modifier leur partition sur la machine sur laquelle les cours sont suivis et les imprimantes ne seront pas disponbiles, donc ces aspects seront vus à domicile. Le chapitre 21 (*Scripts de shell avancés*) n'est pas considéré comme étant essentiel. Les étudiants essayeront de configurer leur(s) imprimate(s) personnelles et discuteront au cours des problèmes rencontrés.

46.8 Leçon 5.

Durant la première heure, le chapitre 23 (*Eléments de programmation en* ***C***) sera vu. La deuxième heure sera consacrée au chapitre 25 (*Paquets sources et binaires*). Durant la troisième heure, les étudiants se consacreront à la lecture des chapitres 26 et 27 (soit : *Introduction à IP* ; *TCP et UDP*). Ils poseront des question sur les points qui ne leur semblent pas clairs.

Travail à domicile : le chapitre 24 (*Bibliothèques partagées*) est optionnel. Nous conseillons la relecture des chapitres 26 et 27 (*Introduction à IP* ; *TCP et UDP*).

46.9 Leçon 6.

La leçon 6 couvre les chapitres 26 et 27 (*Introduction à IP* ; *TCP et UDP*). On y démontre un exemple de détection de mots de passe utilisés par `telnet` grâce à l'utilisation de `tcpdump`. La même opération est réalisée mais `telnet` est remplacé par `ssh`.

Travail à domicile : lecture des chapitres 28 à 30 (c'est-à-dire : *DNS et résolution de noms*, *Systèmes de fichiers en réseau NFS* et *Les services exécutés sous inetd*) en préparation à la leçon 7.

46.10 Leçon 7.

Les chapitres 28 à 30 (c'est-à-dire : *DNS et résolution de noms*, *Systèmes de fichiers en réseau NFS* et *Les services exécutés sous inetd*) sont vus durant les deux premières heures. Un serveur DNS aura été installé afin que les étudiants puissent l'utiliser. Durant la dernière heure, on explique comment le courriel est distribué sur l'internet (aspects théoriques) et quelle est la structure du fichier de configuration d'`exim`.

Travail à domicile : lecture du chapitre 31 (*exim et sendmail*) afin de préparer la leçon 8.

46.11 Leçon 8.

Les deux premières heures sont consacrées au chapitre 31 (*exim et sendmail*). Les étudiants configurent leur propre serveur de courriel. Un serveur DNS aura été activé de manière à effectuer des enregistrements `MX` pour leur domaine. La dernière heure est consacrée aux chapitres 32 et 33 (*Démarrage, lilo et initrd* ; *init, ?getty et niveaux d'exécution* UNIX) sauf ce qui concerne les modems.

Travail à domicile : exécuter les commandes du chapitre 33 (*init, ?getty et niveaux d'exécution* UNIX). Le chapitre 35 (*uucp et uux*) ne doit pas être vu. Le chapitre 36 (*Le système de fichiers* LINUX) doit être étudié en profondeur. Les étudiants installeront et configureront un serveur web à partir des instructions du chapitre 37 (*httpd : serveur web Apache*) et rapporteront les difficultés rencontrées. Apache –en soi– ne sera pas vu durant les leçons.

46.12 Leçon 9.

Durant la première heure, le chapitre 38 (*crond et atd*) est traité. Les deux autres heures permettent de voir la chapitre 41 (*named –serveur de noms de domaine*). Les étudiants configurent leur propre serveur de noms pour effectuer des requêtes directes et inverses. Notez que Samba n'est pas vu s'il n'y a pas de machines ou d'imprimantes Windows correctement configurées pour faire une démonstration. Une possibilité pourrait être de monter un partage d'imprimante(s) et de fichiers en utilisant `smbmount`.

Travail à domicile : Les étudiants verront le chapitre 42 (*Protocole point-à-point – Réseau "dialup"*) de manière à configurer un réseau à connexion intermittente, et ce d'eux-mêmes. Lecture du chapitre 43 (*Sources du noyau Linux, modules et support matériel*) pour la préparation de la leçon 10.

46.13 Leçon 10.

Les deux premières heures concernent le chapitre 43 (*Sources du noyau Linux, modules et support matériel*). Les étudiants doivent pouvoir configurer une carte réseau si le pilote de périphérique n'est pas disponible. Ils doivent construire un noyau avec des options de personnalisation. La troisième heure est consacrée au système X-Window (aspects théoriques). Les étudiants doivent

utiliser la variable d'environnement `DISPLAY` afin de pouvoir utiliser une application **X** sur d'autres terminaux que le leur.

Travail à domicile : étudier le chapitre 28 (*DNS et résolution de noms*).

46.14 Leçon 11.

La première leçon se rapporte à la configuration de NFS, en respectant la nécessité d'avoir un nom de serveur qui permette d'effectuer des requêtes directes et inverses. Les deuxième et troisième heures sont consacrées au chapitre 39 (*Serveur postgreSQL*).

Trvail à domicile : télécharger et lire le tutoriel relatif à Python. Consultez les rapports de sécurité hebdomadaire en ligne. Etudier le chapitre 45 (*Unix et la sécurité*).

46.15 Leçon 12.

Les première et deuxième heures sont consacrées à la sécurité (chapitre 45 : *Unix et la sécurité*) et à une introduction au langage de programmation Python. La dernière heure consiste en une évalutation. Cette dernière leçon pourrait être consacrée à un examen si le cours est préparatoire à une certification.

Chapitre 47

Certification LPI – Références croisées.

Les pré-requis pour la réussite de l'habilitation LPI ont été extraits intégralement du site web LPI francophone : http ://www.fr.lpi/ (actualisation 2005). Pour chaque objectif, le chapitre ou la section pertinents du livre est indiqué entre parenthèse (addition au texte LPI original). Parfois certaines références externes sont communiquées.

Chaque objectif se voit attribuer une pondération allant de 1 à 8, qui permet de déterminer l'importance relative de la matière. Plus la pondération est élevée, plus le nombre de questions posées à l'examen est grand.

47.1 Détails de l'examen pour 101.

47.2 Linux général (Partie I).

Cet examen est nécessaire à l'obtention du certificat de niveau I. Il couvre les activités fondamentales de l'administration système qui sont communes aux différentes versions de GNU/Linux.

47.2.1 Sujet 1.3 : Commandes Unix et GNU.

Obj 1 : Travaillez efficacement avec la ligne de commande Unix

Poids de l'objectif : 4

Interagir avec les shells et les commandes depuis la ligne de commande (chapitre 5). Ceci comprend : taper des commandes et des séquences de commandes valides (chapitre 5) ; déclarer, définir et exporter des variables d'environnement (chapitre 10) ; utiliser l'historique des commandes et les facilités d'édition (section 3.6) ; invoquer des commandes dans le chemin et hors du chemin (section 5.6) ; utiliser des substitutions de commandes ; appliquer des commandes récursivement sur une arborescence (section 21.7.5).

Obj 2 : Traiter des flux de textes en utilisant les outils de filtres de textes *Poids de l'objectif : 7*

Envoyer des fichiers textes et des flux de sorties vers des utilitaires de filtrage de textes pour modifier la sortie de manière utile (chapitre 9). Ceci inclut l'utilisation des utilitaires UNIX standards qu'on trouve dans l'archive GNU textutils (sed, sort, cut, expand, fmt, head, join, nl, od, paste, pr, plit, tac, tail, tr, wc) (voir les pages de man de ces commandes en plus chapitre 9).

Obj 3 : Effectuer les opérations basiques de gestion de fichiers *Poids de l'objectif : 2*

Utiliser les commandes de base UNIX pour copier et déplacer des fichiers et répertoires (chapitre 5). Effectuer des opérations avancées de gestion de fichiers comme la copie récursive de fichiers et le déplacement de fichiers qui correspondent à des motifs (chapitre 5). Utiliser des motifs simples et avancés pour désigner des fichiers (section 5.3).

Obj 4 : Utilisation des flux UNIX, des tubes, et des redirections *Poids de l'objectif : 3*

Passer des fichiers à des commandes et des commandes à d'autres commandes pour traiter efficacement des données en mode texte. Ceci comprend les redirections d'entrée, de sortie et d'erreur standards ainsi que l'utilisation des tubes pour utiliser la sortie d'une commande en tant qu'entrée ou argument (avec xargs) d'une autre commande, et l'envoi d'une sortie vers la sortie standard et dans un fichier (avec tee) (chapitre 10).

Obj 5 : Créer, surveiller et tuer des processus *Poids de l'objectif : 5*

Lancer des tâches en avant-plan ou en arrière-plan (chapitre 10), déplacer une tâche de l'avant-plan à l'arrière-plan et vice versa, surveiller les processus actifs, et tuer des processus. Ceci présuppose notamment l'utilisation des commandes ps, top, kill, bg, fg and jobs (chapitre 10).

Obj 6 : Modification de la priorité des processus *Poids de l'objectif : 2*

Lancer un programme avec un niveau de priorité plus ou moins élevé, déterminer la priorité d'un processus, changer la priorité d'un processus existant (section 10.7). Ceci présuppose notamment l'utilisation de la commande nice et des commandes associées (section 10.7).

Obj 7 : Effectuer des recherches dans des fichiers textes en utilisant les expressions rationnelles.

Poids de l'objectif : 3 Créer des expressions rationnelles simples et utiliser des outils comme grep ou sed pour effectuer les recherches (chapitres 5 et 8).

47.2.2 Sujet 2.4 : périphériques et système de fichiers LINUX, hiérarchie standard du système de fichiers.

Obj 1 : Création de partitions et de systèmes de fichiers *Poids de l'objectif : 3*

Créer des partitions en utilisant fdisk, créer le système de fichiers d'un disque dur ou d'un autre support en utilisant mkfs (chapitre 20).

Obj 2 : Maintenir l'intégrité du système de fichiers *Poids de l'objectif : 5*

Vérifier l'intégrité du système de fichiers, surveiller l'espace libre et les i-noeuds (*inodes*), régler les problèmes des systèmes de fichiers simples. Ceci suppose la connaissance des commandes fsck, du, df (chapitre 20).

Obj 3 : Contrôle des systèmes de fichiers, montage et démontage *Poids de l'objectif : 3*

Monter et démonter manuellement des systèmes, configurer le montage des systèmes de fichiers lors de l'amorçage, configurer le montage par les utilisateurs des médias amovibles. Ceci suppose la connaissance de la syntaxe du fichier /etc/fstab (chapitre 20).

Obj 4 : Mettre en place et voir les quota disques *Poids de l'objectif : 1*

Mettre en place des quotas pour un système de fichiers, modifier les quota disque pour un système de fichiers, vérifier les quotas pour des utilisateurs, produire un rapport des quotas utilisateurs. Commandes à connaître : quota, edquota, repquota, quotaon (Les quotas n'ont pas été vu mais vous pouvez aisément consulter le Mini-HOWTO Quota).

Obj 5 : Utiliser les droits des fichiers pour en contrôler l'accès *Poids de l'objectif : 3*

Changer les droits des fichiers, répertoires et fichiers spéciaux, utiliser les modes de droits spéciaux comme les bits suid et sticky, utiliser le champ groupe pour permettre l'accès au fichier à un groupe de travail, changer les droits par défaut lors des créations de fichiers. Ceci suppose la connaissance des commandes chmod et umask et nécessite la compréhension des modes de droits symboliques et numériques (chapitre 15).

Obj 6 : Gérer les appartenances de fichiers *Poids de l'objectif : 2*

Changer le propriétaire ou le groupe d'un fichier, contrôler à quel groupe sont attribués les nouveaux fichiers dans un répertoire. Ceci suppose la connaissance des commandes chmod et chgrp (chapitre 12).

Obj 7 : Créer et changer des liens physiques ou symboliques *Poids de l'objectif : 2*

Créer des liens physiques ou symboliques, trouver les liens physiques vers un fichier, copier des fichiers en suivant ou ne suivant pas les liens symboliques, utiliser les liens physiques et symboliques dans le cadre d'une administration efficace (chapitre 15).

Obj 8 : Trouver les fichiers système et placer les fichiers au bon endroit *Poids de l'objectif : 2*

Comprendre la hiérarchie standard du système de fichiers, connaître l'emplacement des fichiers standards, connaître l'utilité des différents répertoires systèmes, trouver des commandes et des fichiers. Ceci met en jeu les commandes : find, locate, which, updatedb et la modification de /etc/updatedb.conf (section 5.14 ainsi que les chapitres 18 et 36).

47.2.3 Sujet 2.6 : Démarrage, initialisation, arrêt et niveaux d'exécution.

Obj 1 : Amorcer le système *Poids de l'objectif : 3*

Guider le système pendant le processus d'amorçage, passer des options au noyau au moment de l'amorçage, et vérifier le déroulement dans les fichiers de journalisation. Ceci met en jeu la commande dmesg (lilo) et l'étude des fichiers /var/log/messages, /etc/lilo/conf, /etc/conf.modules (ou /etc/modules.conf) (sous-sections 22.4.8 et 43.5.1 ainsi que les chapitres 32 et 33).

Obj 2 : Changer le niveau d'exécution, extinction et redémarrage du système *Poids de l'objectif : 3*

Changer de manière sûre le niveau d'exécution du système, spécialement en mode mono-utilisateur, arrêter ou ré-amorcer. S'assurer que les utilisateurs actifs soient alertés et que les processus se terminent naturellement. Ceci met en jeu les commandes shutdown et init (chapitre 33).

47.2.4 Sujet 1.8 : Documentation.

Obj 1 : Utiliser et gérer la documentation locale au système *Poids de l'objectif : 5*

Utiliser et administrer les pages man et les données dans /usr/doc. Ceci comprend la recherche des pages nécessaires, la recherche dans les sections des pages man, la recherche de commandes et des pages man qui leur sont associées, la configuration de l'accès aux sources des pages man et au système man, l'utilisation de la documentation système stockée dans /usr/doc et dans les emplacements associés, la détermination des documents qu'il faut garder dans /usr/doc (voir la section 5.7 et le chapitre 17 ; vous devriez aussi étudier la page de manuel de la commande `man`).

Obj 2 : Trouver de la documentation relative à Linux sur Internet *Poids de l'objectif : 2*

Trouver et utiliser de la documentation Linux grâce aux sources telles que le "Linux Documentation Project", les sites de fabriquants et de tiers, les groupes de discussions, les archives des groupes de discussion, les listes de diffusion (chapitre 14).

Obj 3 : Écrire de la documentation système *Poids de l'objectif : 1*

Écrire de la documentation et maintenir les journaux pour les conventions locales, les procédures, la configuration et les changements de configuration, la localisation des fichiers, les applications et les script shells.(Vous devriez apprendre comment écrire vous-même une page de man. Il existe de nombreuses page de man à copier en tant qu'exemples. Il est difficile de dire ce que le LPI entend par “écrire une documentation système”).

Obj 4 : Fournir un support utilisateur *Poids de l'objectif : 1*

Fournir de l'assistance technique aux utilisateurs par téléphone, courriel, et contact direct. (Ceci n'a pas été vu. Fournir une aide à des utilisateurs peut être réalisé en répondant à des questions sur une liste de diffusion ou sur un forum, par exemple).

47.2.5 Sujet 2.11 : Tâches administratives.

Obj 1 : Gérer les comptes utilisateurs et les groupes, ainsi que les fichiers systèmes relatifs *Poids de l'objectif : 7*

Ajouter, supprimer et suspendre des comptes utilisateurs ; ajouter et retirer des groupes ; changer les informations des utilisateurs et des groupes dans les fichiers passwd et group ; créer des comptes spéciaux et des comptes limités. Ceci comporte l'utilisation des commandes : useradd, userdel, groupadd, gpasswd, passwd, et des fichiers : passwd, group, shadow, et gshadow (chapitre 12 ; vous devriez revoir en détail les pages de man d'`useradd` et de `groupadd`).

Obj 2 : Paramétrer l'environnement utilisateur et les variables d'environnement du système *Poids de l'objectif : 4*

Modifier les profils "global" et "utilisateurs" pour fixer les variables d'environnements ; maintenir le répertoire skel pour la création des nouveaux comptes utilisateurs ; placer les commandes appropriées dans le chemin. Ceci implique l'édition des fichiers /etc/profile et /etc/skel (chapitre 12 et section 21.8).

Obj 3 : Configurer et utiliser les fichiers de journalisation système pour répondre à des besoins administratifs et de sécurité *Poids de l'objectif : 3*

Configurer le type et le niveau d'information à journaliser, vérifier manuellement les fichiers de journalisation pour trouver des traces d'une activité ; automatiser la rotation et l'archivage des journaux ; effectuer un suivi des problèmes trouvés dans les journaux. Ceci suppose la modification de /etc/syslog.conf (Sections 22.4.8 et 22.4.9).

Obj 4 : Automatiser les tâches administration système par la programmation des travaux à effectuer dans le futur *Poids de l'objectif : 4*

Utiliser cron pour lancer des tâches à intervalles réguliers ; utiliser at pour lancer des taches à des dates précises ; gérer les taches cron et at ; configurer l'accès des utilisateurs aux services cron et at (chapitre 38).

Obj 5 : Maintenir une stratégie efficace de sauvegarde des données *Poids de l'objectif : 3*

Prévoir une stratégie de sauvegarde ; sauvegarder automatiquement les systèmes de fichiers sur différents supports ; réaliser des sauvegardes partielles et manuelles ; vérifier l'intégrité des fichiers de sauvegarde ; restaurer partiellement ou complètement une sauvegarde (section 5.17 et chapitre 19).

47.3 Détails de l'examen pour 102.

47.4 Linux général (Partie II)

47.4.1 Sujet 1.1 : Matériel et architecture.

Obj 1 : Configurer le matériel bas-niveau *Poids de l'objectif : 3*

Démontrer une bonne compréhension des réglages importants du BIOS ; fixer la date et l'heure ; s'assurer que les IRQ et les adresses I/O sont correctes pour tous

les ports y compris les ports séries et parallèles ; noter les IRQ ; être au courant des problèmes relatifs aux disques qui ont plus de 1024 cylindres (chapitres 5 et 43).

Obj 2 : Paramétrer les périphériques SCSI et les cartes réseaux *Poids de l'objectif : 4*

Manipuler le BIOS SCSI pour détecter les ID SCSI utilisées et disponibles ; fixer l'ID SCSI à la valeur correcte pour le périphérique de démarrage et pour n'importe quel autre périphérique le nécessitant ; formater un disque SCSI (bas niveau avec les outils d'installation du fabriquant) ; le partitionner et le formater avec les outils fdisk et mke2fs de LINUX. Paramétrer les cartes réseaux en fixant les I/O et les interruptions ainsi que les DMA si nécessaire (en utilisant les utilitaires du fabriquant). (Sections 43.6.3 et 43.6.9. Chaque fabricant de matériel a ses propres outils.)

Obj 3 : Configurer modem et cartes sons *Poids de l'objectif : 3*

Vérifier que les périphériques sont supportés (particulièrement que le modem n'est PAS un winmodem) ; vérifier que le modem et la carte son utilisent des IRQ, I/0, DMA uniques et corrects. Si la carte son est PnP, installer et lancer sndconfig et isapnp ; configurer un modem pour la numérotation sortante, configurer un modem pour les connections sortante PPP, SLIP, CSLIP ; paramétrer le port série pour 115.2 Kbps (sections 43.6.1, 43.6.12 et 43.7 ainsi que les chapitres 35 et 41).

47.4.2 Sujet 2.2 : Installation de LINUX et gestion des paquets.

Obj 1 : Conception d'un partitionnement d'un disque dur *Poids de l'objectif : 2*

Etablir un schéma de partitionnement dépendant du matériel et de l'utilisation du système (nombre de disques, taille des partitions, points de montages, emplacement du noyau, espace de swap) pour un système LINUX (chapitre 20).

Obj 2 : Installation d'un boot loader *Poids de l'objectif : 3*

Sélectionner, installer et configurer un chargeur de démarrage sur l'emplacement disque approprié. Fournir des choix et des options de récupération (comme une disquette de réamorçage). Ceci suppose l'utilisation de la commande lilo et la configuration de /etc/lilo.conf (chapitre 32).

Obj 3 : Compiler et installer des programmes depuis les sources *Poids de l'objectif : 5*

Gérer des archives (compressées) de fichiers (décompresser des "tarballs"), spécialement les sources des paquets GNU. Les installer et les configurer sur votre système. Faire des personnalisations simples du Makefile si nécessaire (comme des chemins, des inclusions supplémentaires de répertoires), compiler et installer des exécutables. Ceci suppose l'utilisation des commandes : gnuzip, tar, ./configure, make, make install ainsi que la configuration des fichiers de Makefile (chapitre 25).

Obj 4 : Gestion des bibliothèques partagées *Poids de l'objectif : 3*

Déterminer les dépendances d'un programme vis-à-vis des bibliothèques partagées, et installer celles qui sont nécessaires. Ceci suppose l'utilisation des commandes : ldd, ldconfig et la configuration du fichier /etc/ld.so.conf (chapitre 24).

Obj 5 : Utiliser le système de gestion de paquets Debian *Poids de l'objectif : 5*

Utilisation du système de gestion de paquets Debian, depuis la ligne de commande (dpkg) et avec des outils interactifs (dselect). Etre capable de trouver un paquet contenant des fichiers spécifiques ou un logiciel; les sélectionner et les récupérer depuis des archives; les installer, les mettre à jour, ou les supprimer; obtenir les informations d'état comme la version, le contenu, les dépendances, l'intégrité, l'état d'installation; déterminer quels paquets sont installés et par quel paquets un fichier a été installé. Être capable d'installer des paquets non-Debian sur un système Debian (chapitre 25).

Ceci suppose l'utilisation des commandes : dpkg, dselect, apt, apt-get, alien. Ceci suppose la lecture et l'édition des fichiers des répertoires : /var/lib/dpkg/*.

Obj 6 :Utiliser le système de gestion de paquets Redhat (rpm) *Poids de l'objectif : 6*

Utiliser rpm depuis la ligne de commande. Être familier avec les tâches suivantes : installer et enlever un paquet, déterminer la version d'un paquet et la version des logiciels qu'il contient, afficher la liste des fichiers d'un paquet, afficher la liste des fichiers de documentation d'un paquet, trouver le paquet qui a installé un fichier donné, trouver les paquets installés sur le système (tous les paquets ou un sous-ensemble de paquets), identifier dans quel paquet on peut trouver un programme ou un fichier donné, vérifier l'intégrité d'un paquet, vérifier la signature PGP ou GPG d'un paquet, mettre à jour un paquet. Ceci suppose l'utilisation des commandes et programmes : rpm, grep (chapitre 25).

47.4.3 Sujet 1.5 : Noyau

Obj 1 : Gestion des modules du noyau au démarrage *Poids de l'objectif : 3*

Apprendre quelles sont les fonctionnalités disponibles grâce aux modules du noyau; insérer et supprimer les modules qui doivent l'être. Ceci suppose l'utilisation des commandes : lsmod, insmod, rmmod, modinfo, modprobe ainsi que la connaissance des fichiers : /etc/modules.conf, /etc/conf.modules (en fonction des distributions), /lib/modules/{kernel-version}/modules.dep (chapitre 43).

Obj 2 : Reconfigurer, compiler et installer un noyau et des modules personnalisés *Poids de l'objectif : 4*

Obtenir et installer les sources et les en-têtes officiels du noyau (récupérés depuis un site, un CD, kernel.org, ou un fournisseur); paramétrer le noyau (le reconfigurer depuis le fichier .config existant si nécessaire en utilisant oldconfig, menuconfig ou xconfig); compiler un nouveau noyau LINUX et ses modules; installer le nouveau noyau et les modules au bon endroit; reconfigurer et lancer lilo.

Remarque : ceci ne nécessite pas le passage à une nouvelle version du noyau depuis les sources ou un patch. Ceci suppose l'utilisation des commandes : make (dep, clean, menuconfig, bzImage, modules, modules_install), depmod, lilo ainsi que la connaissance et la configuration des fichiers : /usr/src/linux/.config, /usr/src/linux/Makefile, /lib/modules/{kernelversion}/modules.dep, /etc/conf.modules, /etc/modules.conf et /etc/lilo.conf (chapitre 25).

47.4.4 Sujet 1.7 : Edition de textes, mise en oeuvre, impression.

Obj 1 : Réaliser les opérations élémentaires de modification de fichiers avec vi *Poids de l'objectif : 2*

Modification de fichiers textes avec vi. Ceci suppose de connaître sous vi : la navigation, les modes basiques, l'insertion, la modification et la suppression de texte, la recherche de texte et la copie de texte (chapitre 7).

Obj 2 : Gérer les imprimantes et les files d'impression *Poids de l'objectif : 2*

Surveiller et gérer les files d'impression et les travaux d'impression utilisateur ; régler les problèmes d'impression courants. Ceci suppose l'utilisation des commandes : lpc, lpq, lprm et lpr et la connaissance de la syntaxe du fichier /etc/printcap (chapitre 22).

Obj 3 : Imprimer des fichiers *Poids de l'objectif : 1*

Soumettre des travaux d'impression, convertir des fichiers textes en postscript pour l'impression. Ceci suppose l'utilisation de lpr (section 22.6).

Obj 4 : Installer et configurer des imprimantes locales et distantes *Poids de l'objectif : 3*

Installer un démon d'impression, installer et configurer un filtre d'impression (typiquement : apsfilter, magicfilter). Rendre les imprimantes locales et distantes accessibles pour un système LINUX (imprimantes PostScript, non-PostScript ou Samba). Ceci suppose la connaissance du démon lpd et la modification ou la connaissance de la structure des fichiers et répertoires : /etc/printcap, /etc/apsfilterrc, /usr/lib/apsfilter/filter/*/, /etc/magicfilter/*/, /var/spool/lpd/*/ (section 22.9.2).

47.4.5 Sujet 1.9 : Shell, scripts, programmation, compilation.

Obj 1 : Paramétrer et utiliser l'environnement shell *Poids de l'objectif : 4*

Personnaliser son environnent shell : assigner une valeur à des variables d'environnement (par exemple PATH) à la connexion ou à l'invocation d'un nouveau shell ; écrire des fonctions en bash pour les commandes fréquemment utilisées. Ceci suppose l'édition des fichiers du répertoire utilisateur : .bash_profile, .bash_login, .profile, .bashrc, .bash_logout, .inputrc (chapitre 21).

Obj 2 : Personnaliser ou écrire des scripts simples *Poids de l'objectif : 5*

Personnalisation de scripts (comme la modification de chemin pour des scripts dans n'importe quel langage), ou écriture de nouveaux (quoique simples) scripts en (ba)sh. En plus d'utiliser la syntaxe standard de sh (boucles, test), vous devrez savoir utiliser : la substitution de commandes et les tests des valeurs de retour des fonctions, le test du statut de fichiers, et le courriel au superutilisateur sous certaines conditions. Assurez-vous que le bon interpréteur est appellé à la première ligne (#!), et prenez en compte l'emplacement, le propriétaire, et les droits d'exécution et suid du script (chapitre 21 ; setuid est vu aux sections 34.2 et 37.2.10 sous un angle un peu plus pratique).

47.4.6 Sujet 2.10 : X

Obj 1 : Installer et configurer XFree86 *Poids de l'objectif : 4*

Vérifier que la carte vidéo et le moniteur sont supportés par le serveur X, et installer le bon serveur X ; installer un serveur de polices de caractères pour X ; installer les polices de caractères nécessaires pour X (ceci peut nécessiter une modification manuelle du fichier etc/X11/XF86Config dans la section "Files") ; personnaliser et optimiser X pour la carte vidéo et le moniteur. Commandes : XF86Setup, xf86config. Fichiers : /etc/X11/XF86Config, .xresources (chapitre 44).

Obj 2 : Configurer XDM *Poids de l'objectif : 1*

Activer et désactiver xdm, changer le message d'accueil d'xdm ; changer l'image de fond pour xdm ; paramétrer xdm pour utiliser des stations graphiques X (consulter la page de man d'xdm pour une information avancée).

Obj 3 : Identifier et tuer les applications X persistantes *Poids de l'objectif : 1*

Identifier et tuer les applications X qui ne se termineront pas après la fin de la session X. Exemple : netscape, tkrat, etc.

Obj 4 : Installer et personnaliser l'environnement d'un gestionnaire de fenêtres *Poids de l'objectif : 4*

Sélectionner et personnaliser l'environnement gestionnaire de fenêtres et/ou bureaux par défaut du système ; démontrer une compréhension des procédures de personnalisation des menus des gestionnaires de fenêtres ; configurer les menus pour le gestionnaire de fenêtres ; sélectionner et configurer le terminal X désiré (xterm, rxvt, aterm etc.) ; vérifier et résoudre les problèmes de dépendances en bibliothèques des applications X ; exporter un affichage X à l'écran vers une station de travail distante. Fichiers : .xinitrc, .Xdefaults, divers fichier .rc (les pages de man des commandes `xinit`, `startx` et `xdm` fournissent l'information).

47.4.7 Sujet 1.12 : Bases du réseau.

Obj 1 : Base en TCP/IP *Poids de l'objectif : 4*

Montrer une compréhension des masques de réseau et ce qu'ils signifient (c'est-à-dire, déterminer l'adresse d'une machine à partir du masque de sous-réseau) ; comprendre les protocoles TCP/IP de base (TCP, UDP, ICMP) et aussi PPP ; démontrer une compréhension du but et de l'utilisation des ports les plus communs trouvés dans /etc/services (20, 21, 23, 25, 53, 80, 110, 119, 139, 143, 161) ; démontrer une compréhension correcte des fonctions et des applications liées à la route par défaut. Exécuter les tâches TCP/IP de base : FTP, anonymous FTP, telnet, host, ping, dig, traceroute, whois (chapitres 26 et 27).

Obj 2 : néant.

Obj 3 : Configuration et correction des problèmes de TCP/IP *Poids de l'objectif : 10*

Démontrer une compréhension des techniques nécessaires pour afficher, configurer et vérifier qu'une interface réseau est opérationnelle ; changer, afficher et configurer une table de routage ; corriger une route par défaut mal configurée. Manuellement ajouter / démarrer / arrêter / redémarrer / effacer / reconfigurer une interface réseau, et configurer LINUX comme client DHCP, comme hôte TCP/IP et déboguer

les problèmes associés. Ceci peut faire entrer en compte la configuration et l'étude des fichiers et répertoires : /etc/hostname, /etc/hosts, /etc/networks, /etc/host.conf, /etc/resolv.conf, et d'autres fichiers de configuration réseau propres à votre distribution. Ceci suppose l'utilisation des commandes et programmes suivants : dhcpd, host, hostname (domainname, dnsdomainname), ifconfig, netstat, ping, route, traceroute et des scripts réseau lancés durant l'initialisation (chapitres 26 et 28).

Obj 4 : Configurer and utiliser PPP *Poids de l'objectif : 4*

Définir la séquence de dialogue pour se connecter (un exemple de connexion étant donné), paramétrer les commandes devant être lancées automatiquement à la connexion, initier et terminer une connexion PPP, initier et terminer une connexion ISDN, paramétrer PPP automatiquement pour se reconnecter automatiquement après déconnexion (chapitre 42).

47.4.8 Sujet 1.13 : Services réseau.

Obj 1 : Configurer et gérer inetd et les services associés *Poids de l'objectif : 5*

Configurer les services disponibles depuis inetd, utiliser les interfaces TCP (TCP-wrappers) pour permettre ou refuser l'accès à des services sur la base de filtre par hôtes. Manuellement démarrer, arrêter et redémarrer les services internet, configurer les services réseau de base comme telnet et ftp. Ceci suppose la gestion d'inetd.conf, hosts.allow et hosts.deny (chapitre 30).

Obj 2 : configuration et usage de base de sendmail *Poids de l'objectif : 5*

Modifier des paramètres simples dans les fichiers de configuration de sendmail (modifier la valeur DS pour le "Smart Host", si nécessaire), créer des alias de courriel, gérer la file d'attente des messages électroniques, démarrer et arrêter sendmail, configurer le suivi du courriel (.forward), réaliser la correction de problèmes élémentaires de sendmail. Ceci suppose l'utilisation des commandes mailq, sendmail et newaliases et l'utilisation des alias et des fichiers de config du répertoire mail/ (chapitre 31).

Obj 3 : configuration et usage de base d'Apache *Poids de l'objectif : 3*

Modifier des paramètres simples dans les fichiers de configuration d'Apache ; démarrer, arrêter et redémarrer httpd ; obtenir le démarrage automatique d'httpd lors de l'initialisation. Ceci ne comprend pas la configuration avancée d'Apache mais suppose la gestion des fichiers de configuration d'Apache (chapitre 37).

Obj 4 : Gestion efficace des démons NFS, smb, et nmb *Poids de l'objectif : 4*

Monter des systèmes de fichiers distants en utilisant NFS, configurer NFS pour exporter des systèmes de fichiers locaux, démarrer, arrêter et redémarrer le serveur NFS. Installer et configurer Samba en utilisant les outils IHM inclus ou, par édition directe du fichier /etc/smb.conf. (Remarque : ceci exclut délibérément les spécificités avancées des domaines NT mais comprend le partage simple des répertoires /home et des imprimantes, ainsi que le paramétrage de nmbd comme client WINS) (chapitres 29 et 40).

Obj 5 : Configuration et usage élémentaire des services DNS *Poids de l'objectif : 3*

Configurer la recherche de noms en utilisant les fichiers /etc/hosts, /etc/resolv.conf, /etc/host.conf, et /etc/nsswitch.conf ; régler les problèmes d'un serveur de noms local en mode "cache" uniquement. Ceci nécessite la compréhension du processus d'enregistrement des noms de domaine et du processus de résolution de noms par translation. Ceci nécessite la compréhension des différences clés entre les fichiers de configuration de bind 4 et bind 8. Ceci suppose l'utilisation des commandes nslookup, host et des fichiers named.boot (v. 4) ou named.conf (v. 8) (chapitres 28 et 41).

47.4.9 Sujet 1.14 : Sécurité.

Obj 1 : Réaliser les tâches d'administration relative à la sécurité *Poids de l'objectif : 4*

Configurer et utiliser l'interface TCP (*TCP-wrapper*) pour verrouiller le système, afficher tous les fichiers dont le bit SUID est activé, déterminer si un paquet (.rpm ou .deb) a été corrompu. Vérifier les nouveaux paquets avant l'installation, utiliser setgid sur des répertoires pour garder cohérents les droits des groupes, changer le mot de passe d'un utilisateur, fixer des dates d'expiration sur les mots de passes utilisateurs, obtenir, installer et configurer ssh (chapitre 45).

Obj 2 : Paramétrer la sécurité de la machine *Poids de l'objectif : 4*

Mettre en place les mots de passe cryptés (shadow password), éteindre les services réseaux inutiles par inetd, définir le bon alias courriel pour root et régler syslogd, surveiller CERT et BUGTRAQ, mettre à jour immédiatement les binaires concernés lorsque un problème de sécurité est trouvé (chapitre 45).

Obj 3 : Setup user level security *Poids de l'objectif : 2*

Fixer des limites aux connexions, processus, et limiter l'usage mémoire des utilisateurs (section 12.7.5).

Chapitre 48

Certification RHCE - Références croisées.

L'acronyme RHCE signifie *RedHat Certified Engineer*.

RedHat a developpé un grand nombre de cours qui se chevauchent, certains étant moins lourds que d'autres, avec des documents plus accessibles. Ces cours, centrés autour de RedHat, ne sont pas toujours applicables à d'autres distributions. Dans certains secteurs, les habilitations RedHat sont plus exigentes que ce n'est le cas avec le LPI. Donc, il est pertinent de réaliser un contrôle de vos connaissances vis-à-vis de ce qui est demandé pour les habilitations RedHat. L'information contenue dans les sections qui suivent provient de discussions avec des personnes ayant suivi les cours RedHat. Il est très possible que les données présentées ici doivent être adaptées. En aucun cas, vous ne devriez les considérer comme définitives. Dès lors, visitez http ://redhat.com/training/rhce/courses/ pour un guide officiel.

Pour chaque objectif, le(s) chapitre(s) ou la (les) section(s) du livre sont indiqués entre parenthèses.

48.1 RH020, RH030, RH033, RH120, RH130, RH133.

Ces cours sortent du cadre de ce livre car ils concernent Linux du point de vue de l'utilisateur final et du bureau. Bien qu'ils incluent une description des tâches administratives, il y a peu d'aspects techniques. Ils privilégient les programmes de configuration graphiques pour les tâches administratives. Un des objectifs de ces cours est de configuer les applets de Gnome. S'y discute aussi l'usage de l'éditeur `pico`.

48.2 RH300.

Cette certification concerne les administrateurs non-Linux qui souhaitent élargir leur champ d'administration au système Linux. Les pré-requis de cette habilitation penchent vers la compréhension des alternatives et des caractéristiques propres à Linux, plutôt que vers des configurations Linux complexes. Notez que dans ce qui suit le terme *Guide d'Installation RedHat* a été contracté en GIRH. Il se rapport à l'aide associée à l'installeur RedHat (ce qui pour les systèmes RedHat 6.2 correspond au guide HTML

du CD-ROM de la distribution). Il se rapporte aussi à la documentation complète accessible sur http ://www.redhat.com/support/manuals.

48.2.1 Unité 1 : Sélection du matériel et installation de RedHat.

- Trouver de la documention sur l'internet. Utiliser les HOWTO pour identifier le matériel compatible (chapitre 17).
- Connaissance des architectures supportées et du support SMP (chapitre 43).
- Utilisation de `kudzu` (nous n'avons pas vu `kudzu` et avons recommandé de le désinstaller).
- Notions relatives au matériel –IRQ, PCI, EISA, AGP et ports E/S (chapitres 4 et 43).
- `isapnp`, `pciscan` (chapitre 43).
- Notions de support Linux pour PCMCIA, PS/2, lecteurs de bandes, scanners, USB (chapitre 43).
- Notions de périphériques série, parallèle, SCSI, IDE, CD-ROM et lecteurs de disquettes et leur liste dans `/dev` (chapitre 19).
- `hdparm` (voir `hdparm`(8)).
- Notions de géométrie IDE, limitations du BIOS (chapitre 20).
- Secteurs de disques, structure des partitions. Usage de `fdisk`, `cfdisk` et des assistants de configuration des disques (chapitre 20).
- Partitionnement d'un disque (chapitre 20).
- Gestion des partitions swap, natives ou étrangères durant l'installation (GIRH).
- Notion de distribution des répertoires dans les partitions (chapitre 20).
- Configuration de `lilo` à l'installation (le chapitre 32 se rapporte à lilo en général).
- Configuration du BIOS (chapitre 4).
- Compréhension de la notion de différentes images disque. Création et amorçage sur des images disque depuis leurs fichiers `boot.img`, `bootnet.img` ou `pcmcia.img` (GIRH).
- Utilisation de l'installeur pour créer un périphérique RAID (GIRH).
- Sélection de paquets logiciels (GIRH).
- Configuration d'**X** (chapitre 44 et GIRH).

48.2.2 Unité 2 : Configuration et administration.

- Utilisation de `setup`, `mouseconfig`, `Xconfigurator`, `kdbconfig`, `timeconfig`, `netconfig`, `authconfig`, `sndconfig`. (Il s'agit d'utilitaires de plus haut niveau que ceux vus au chapitre 43 et autres. Essayez ces commandes à titre de démonstration).
- Comprendre `/etc/sysconfig/network-scripts/ifcg-*` (chapitre 26).
- Utiliser `netcfg` et `ifconfig` (chapitre 26).
- Utiliser `ifup`, `ifdown`, `rp3`, `usernet` et `usernetctl` (chapitre 26).
- Utiliser `pnpdump`, `isapnp`, et modifier `/etc/ispnp.conf` (chapitre 43).
- Compréhension des notions associées aux fichiers `/etc/conf.modules`, `esd` et `kaudioserver` (chapitre 43, pages de man).
- Utiliser `mount`, modifier `/etc/fstab` (chapitre 20).
- Utiliser `lpr`, `lpc`, `lprm`, `printtool` et comprendre les notions associées à `/etc/-printcap` (chapitre 22).
- Notion de consoles virtuelles ; changements dans `/etc/inittab` (chapitre 32).
- Utiliser `useradd`, `userdel`, `usermod` et `passwd` (chapitre 12).

- Créer des comptes manuellement, avec `userconf` et avec `linuxconf` (l'utilisation d'outils graphiques n'a été encouragée dans le présent livre).
- Comprendre `/etc/passwd`, `/etc/group`, `/etc/skel` et le contenu de ces fichiers (chapitre 12).
- Usage général de `linux.conf` (l'utilisation d'outils graphiques n'a été encouragée dans le présent livre).
- Utiliser `cron`, `anacron`, modifier les fichiers `/var/spool/cron/<utilisateur>` ainsi que `/etc/crontab`, `tmpwatch`, `logrotate`. Localiser les travaux de `cron`.
- Utiliser `syslogd`, `klogd`, `/etc/syslog.conf`, `swatch` et `logcheck`.
- Comprendre et utiliser `rpm`. Sommes de contrôle, affichage de fichiers, forçage, dépendances, requêtes, vérifier les sémaphores des requêtes. Installation de FTP et HTTP, `rmpfind`, `gnorpm` et `kpackage` (chapitre 25).
- Construire des fichiers `.src.rpm`. Personnaliser et reconstruire des paquets (voir le RPM-HOWTO).
- `/usr/sbin/up2date` (l'usage de ce paquet n'a pas été encouragé dans le présent ouvrage).
- Trouver de la documentation (chapitre 17).

48.2.3 Unité 3 : Méthodes d'installation alternative.

- Ordinateurs portables, PCMCIA, `cardmanager` et `apm`. (Voir le GIRH, le PCMCIA-HOWTO et le Laptop-HOWTO).
- Systèmes multiboot, options d'amorçage, configuration d'une image d'amorçage alternative (chapitre 32).
- Installation réseau au moyen de `netboot.img` (GIRH).
- Installation d'une console série (GRIH ?).
- Notions de démarrage rapide.

48.2.4 Unité 4 : Noyau.

- Notions de système de fichiers `/proc` et les divers sous-répertoires (voir la section 43.4 et les entrées de l'index à `/proc`). Réglages des paramètres avec `/etc/sysctl.conf` (voir `sysctl.conf`(5)).
- Quotas disque. `quota`, `quotaon`, `quotaoff`, `edquota`, `repquota`, `quotaawarn`, `quotastats`. (Les quotas n'ont pas été vus dans le présent ouvrage mais il est facile d'apprendre leur maniement grâce au Quota mini-HOWTO).
- Séquences d'initialisation des scripts de démarrage. `inittab`, commutation vers d'autres niveaux d'exécution. Compréhension des notions sous-jacentes aux fichiers `/etc/rc.d/`. Scripts SysV, `chkconfig`, `ntsysv`, `tksysv`, `ksysv` (chapitre 33).
- Configuration du logiciel de RAID. Utilisation de `raidtools` pour activer et tester les périphériques RAID (voir le RAID-HOWTO).
- Gestion des modules. `modprobe`, `depmod`, `lsmod`, `insmod`, `rmmod` ; `kernelcfg`. Modification du fichier `/etc/conf.modules`, faire des alias, commande `option` (chapitre 43).
- Les sources du noyau. Versions `rpm`, système de numérotation des versions du noyau. Configurer, compiler, installer un noyau (chapitre 43).

48.2.5 Unité 5 : Service de base du réseau.

- Notions relatives à TCP/IP. `inetd`. Notions de ports et cartographie des ports des services (chapitres 26 et 27).
- `apache`, fichiers de configuration, hôtes virtuels (chapitre 37).

- `sendmail`, fichiers de configuration, `mailconf`, notions de macro `m4` (chapitre 31).
- Notions relatives à POP et IMAP (chapitres 30 et 31).
- Configuration de `named` (chapitre 41).
- Configuration FTP (nous n'avons pas vu FTP du fait qu'il existe de nombreux logiciels serveurs disponibles. Nous avons recommandé d'essayer `vsftp`).
- `/etc/rc.d/init.d/netfs` (chapitre 29).
- `smbd`, partage de fichiers et d'imprimantes. Notions de sécurité. Utilisation de `testparam`, `smbclient`, `nmblookup`, `smbmount`, notions d'authentification Windows (chapitre 40).
- `dhcpd` et BOOTP, fichiers de configuration. Configuration avec `netcfg`, `netconfig` ou `linuxconf`. Utilisation de `pump` (voir le mini-HOWTO DHCP).
- Comprendre les notions de transfert et de mise en cache de `squid` (la configuration du fichier `/etc/squid/squid.conf` fournit une documentation détaillée de la mise en place de `squid`).
- `lpd`, `mars-nwe` (chapitre 22).

48.2.6 Unité 6 : Le système X Window.

- Architecture du serveur **X** (section 44).
- Utilisation de `Xconfigurator`, `xf86config`, `XF86Setup` et notions relatives à `XF86Config` ou `Xorg-x11.` (section 44.6.3).
- Connaissance des différents gestionnaires de fenêtres, modification du fichier `/etc/sysconfig/desktop`. Compréhension des notions relatives aux différentes interfaces utilisateurs : Gnome et KDE. Utilisation de `switchdesk` (section 44.3.4).
- init, niveau d'exécution 5. Fichiers de configuration utilisateur `.xsession` et `.Xclients` (voir `xinit(1)`, `xdm(1)`, `startx(1)` et lecture des scripts sous `/etc/X11/xinit/` et `/etc/X11/xdm`).
- Utilisation d'`xhost` (section 44.3.4). Problèmes de sécurité. Variable d'environnement `DISPLAY`. Ecrans à distance (section 44.3.5).
- `xfs` (section 44.12).

48.2.7 Unité 7 : Sécurité.

- Utilisation de `tcp_wrappers` (chapitre 30). Limitations d'accès pour les hôtes et les utilisateurs. Accès PAM. Limitations des ports avec `iptables` (voir le Firewall-HOWTO).
- PAM. Modification de `/etc/pam.d`, `/etc/security`. Documentation PAM (voir `/usr/share/doc/pam-0.72/txts/pam.txt`).
- NIS et ses fichiers de configuration. `ypbind`, `yppasswd`, `ypserv`, `yppasswdd`, `makedbm`, `yppush` (voir le NIS-HOWTO).
- LDAP. Paquet logiciel OpenLDAP, `slapd`, `ldapd`, `slurp`, et les fichiers de configuration. Intégration de PAM.
- inetd. Modifier `/etc/inetd.conf`, interface `tcp_wrappers`. Modifier `/etc/host.allow` et `/etc/hosts.deny`. `portmap`, `tcpdhck`, `tcpmatch`, `twist` (voir le LDAP-HOWTO).
- Serveur et client `ssh`. Notions de sécurité (chapitres 13 et 45).

48.2.8 Unité 8 : Pare-feu, routage et "cluster", problèmes.

- Routages statique et dynamique. `/etc/sysconfig/static-routes`. Utilisation de `linuxconf` et `netconfig` pour modifier les routes. (nous n'avons pas encouragé l'utilisation d'outils graphiques de configuration dans le présent ouvrage).

- Notions de redirection. Redirections pour d'autres protocoles : X.25, frame-relay, RNIS et PPP (voir le chapitre 42).
- `iptables` et notions de mise en place de règles. Ajout, suppression, affichage, nettoyage de règles.
- Notions relatives à la "Haute Disponibilité" (High Availability). `lvs`, `pulse`, `nanny`, fichiers de configuration et configuration basées sur le web. Piranha, notions sur le basculement.
- Notions relatives aux grappes de machines (HPC : *High Performance Cluster* ; grappe à haute performance). Machine virtuelle parallèle pour la recherche fondée sur les calculs.
- Dépannage : réseau (chapitre 26), **X** (chapitre 44), amorçage (chapitre 32), DNS (chapitres 28 et 41), authentification (chapitre 12), corruption du système de fichiers (section 20.5).
- `mkbootdisk` et notions de disquette de sauvetage. Utilisation de l'environnement des disques de sauvetage et commandes disponibles (voir `mkbootdisk`(8)).

48.3 RH220 (RH253 Partie I).

RH220 est le module consacré au réseau. Il couvre les services de manière peu approfondie, probablement pour que l'étudiant acquiert l'armature nécessaire à la configuration des services.

48.3.1 Unité 1 : DNS.

Un traitement de bind, analogue au **Sujet 1.13**, **Obj 5** de LPI (page 47.4.8). Une compréhension détaillée est attendue concernant le Domain Name System, les enregistrements `SOA`, `NS`, `A`, `CNAME`, `PTR`, `MX` et `HINFO`, la capacité à gérer des serveurs de domaine, les serveurs en antémémoire seule et la configuration des partages de charge par rotation (chapitre 41).

48.3.2 Unité 2 : Samba.

Vue d'ensemble et notions relatives aux services SMB. Configuration de Samba pour le partage de fichiers et d'imprimantes. Utilisation des outils client de Samba. Utilisation de `linuxconf` et `swat`. Modification de `/etc/smb.conf`. Compréhension des types de partage. Support Wins. Mise en place de l'authentification. Utilisation des utilitaires client (chapitre 40).

48.3.3 Unité 3 : NIS.

Compréhension de NIS. Configurations maître et esclave de NIS. Maniement des utilitaires client. LDAP. Paquet OpenLDAP, `slapd`, `ldapd`, `slurpd`, et fichiers de configuration (voir le NIS-HOWTO).

48.3.4 Unité 4 : Sendmail et procmail.

Compréhension de la mise en réserve (*spooling*) et du transfert de courriel. Compréhension du mécanisme de tous les fichiers de `sendmail`. Modification du fichier de configuration pour la configuration d'un client (redirection ou *forwarding*). Modification de `/etc/sendmail.mc`, `/etc/mail/virtualusertable`, `/etc/mail/access`. Analyse des fichiers de journalisation. Création d'un dossier `.procmail` simple, redirection de courriel. (chapitre 31). Voir aussi la *Sendmail FAQ* http ://www.sendmail.org/faq/ et `procmail`(1), `procmaildir`(6), `procmailex`(5).

48.3.5 Unité 5 : Apache.

Configurer des hôtes virtuels. Ajouter des types MIME. Manipuler l'accès aux répertoires et la gestion des alias de répertoires. Permettre la limitation de l'accès CGI. Installation de bases de données d'utilisateurs et gestion des mots de passe. Compréhension des modules importants (chapitre 37).

48.3.6 Unité 6 : pppd et DHCP.

Installation d'un serveur `pppd` de base. Ajouter des comptes utilisateur pour des communications entrantes. Limitation sur les utilisateurs. Compréhension de `dhcpd`, des fichiers et des notions liées à BOOTUP. Configuration avec `netcfg`, `netconfig` ou `linuxconf`. Utilisation de `pump`. Modification de `/etc/dhcpd.conf` (chapitre 42, voir aussi le HOWTO consacré à DHCP).

48.4 RH250 (RH253 Partie II).

RH250 est un module relatif à la sécurité. Il concerne les bases de l'administration du point de vue de la sécurité.

48.4.1 Unité 1 : Introduction.

Compréhension des impératifs de sécurité. Terminologie de base : `hacker`, `cracker`, `déni de service`, `virus`, `cheval de Troie`, `ver`. Sécurité physique et politiques de sécurité (chapitre 45).

48.4.2 Unité 2 : Sécurité de l'utilisateur local.

Comprendre les notions de compte utilisateur, restreindre les accès basés sur les groupes. Modifier les fichiers de configuration pam. `/etc/nologin` ; modification du fichier `/etc/security`. Utilisation du groupe console, `cug` ; configuration et utilisation de `clobberd` et `sudo`. Vérification des connexions dans les fichiers de journalisation. Utilisation de `last` (chapitres 12 et 45).

48.4.3 Unité 3 : Fichiers et sécurité du système de fichiers.

Traitement exhaustif des groupes et des droits. `chattr` et `lsattr` ; utilisation de `find` pour localiser un problème de droits ; utilisation de `tmpwatch`. Installation de `tripwire`. Gestion des exports de NFS pour le contrôle de l'accès (chapitres 15, 29 et 45).

48.4.4 Unité 4 : Sécurité des mots de passe et cryptage.

Termes associés au chiffrement : *clés publique et privée*, *GPG*, *hachage à sens unique*, *MD5*. `xhost`, `xauth`. Notions et caractéristiques relatives à `ssh`. Notions de craquage de mots de passe (section 12.3 et chapitre 13).

48.4.5 Unité 5 : Sécurité des processus et contrôle.

Utilisation de PAM pour fixer des limites sur les ressources (section 12.7.5). Gestion de l'usage de la mémoire et de la consommation CPU ; `top`, `gtop`, `kpm`, `xosview`, `xload`, `xsysinfo`, `xsysinfo.last`, `ac`, `accton`, `lastcom` (chapitre 10). Gestion des logs avec `swatch` (voir `swatch`(5) et `swatch`(8)).

48.4.6 Unité 6 : Construire un pare-feu.

Notions relatives à `iptables`. Ajouts, suppressions, affichage et nettoyage des règles. Redirections, masquage. Options du noyau pour le support de pare-feu. Routages statique et dynamique (voir le Firewall-HOWTO). `/etc/sysconfig/static-routes`. Utilisation de `linuxconf` et de `netcfg` pour modifier le routage. `tcp_wrappers` (chapitre 30).

48.4.7 Unité 7 : Outils de sécurité.

Notions relatives à `nessus` ; SAINT, SARA, SATAN. Notions relatives à `identd`. Utilisation de `sniffit`, `tcpdump`, `traceroute`, `ping -f`, `etheral`, `iptraf`, `mk-ftp-stats`, `lurkftp`, `mrtg`, `netwatch`, `webalizer`, `trafshow` (ces outils peuvent être recherchés sur l'internet).

Chapitre 49

Foire-aux-questions Linux.

Les capacités de Linux *sont en progrès constants. Aussi, prenez la peine de consulter les différentes ressources internet citées dans le livre pour obtenir une information à jour.*

49.1 Survol général de Linux.

Cette section décrit des questions d'intérêt général qui concerne Linux dans son ensemble.

49.1.1 Qu'est-ce que Linux ?

Linux est le coeur d'un système d'exploitation Unix –libre– pour PCs et d'autres plate-formes. Le développement de ce système d'exploitation a commencé en 1984 ; il s'agit du projet GNU de la *Free Software Foundation* (FSF). Le noyau Linux, nommé ainsi d'après son auteur, Linus Torvalds, a été développé dès 1991. Sa première version utilisable a été publiée en 1993. Linux est souvent appelé GNU/Linux pour indiquer la combinaison des programmes libres du projet GNU et du noyau .

Les systèmes Unix ont été développés dans les années '60 et constituent un standard de l'industrie. Linux est dit "conforme à POSIX", ce qui signifie qu'il se conforme à certaines normes de l'informatique, mises au point par les mondes industriel et académique. La conséquence est que Linux est largement compatible avec d'autres systèmes Unix (le même programme peut aisément être porté et exécuté sur d'autres systèmes Unix avec peu –voire pas– de modifications). Linux fonctionne en réseau avec d'autres Unix, sans arrangement spécial.

Il existe des Unix commerciaux : IRIX (de Silicon Graphics), Solaris ou SunOS (de Sun Microsystem, qui fonctionne sur des stations de travail à architecture SPARC), HP-Unix (pour les serveurs HP), OSF (qui fonctionne sur les machines Alpha) et AIX pour les architectures PowerPC/RS6000.

Il existe également des Unix libres qui gagnent en popularité en raison de leur exceptionnelle stabilité : FreeBSD, NetBSD et OpenBSD.

Les systèmes Linux sont multi-tâches et multi-utilisateurs, ce qui signifie que différents utilisateurs peuvent employer des programmes différents ou identiques en se connectant simultanément à la même machine.

49.1.2 Que sont les systèmes UNIX ? Que peut faire LINUX ?

Les systèmes UNIX forment la colonne vertébrale de l'internet. L'industrie lourde, les missions à applications critiques, les universités utilisent UNIX. Les serveurs de haut-de-gamme, les ordinateurs centraux et les super-ordinateurs utilisent UNIX. Ces systèmes sont utilisés par les grandes sociétés fournissant des accès internet aussi bien que par de petits entreprises. UNIX est un standard pour le matériel neuf et les nouvelles plate-formes car ce système est le plus portable. Les systèmes UNIX sont utilisés pour supporter de grandes bases de données, de fichiers, et comme serveurs Internet. UNIX est aussi utilisé pour la visualisation et les rendus graphiques haut de gamme (ce système d'exploitation est utilisé par l'industrie cinématographique d'Hollywood). Les industries et les universités l'employent pour des simulations à caractère scientifique et pour le calcul à haute charge dans des grappes de machines. Le marché des systèmes d'exploitation embarqués (de petits ordinateurs sans opérateur qui sont implantés dans des appareils) s'est récemment tourné vers LINUX, ces systèmes sont utilisés par millions.

En soi, LINUX peut fonctionner comme serveur web, serveur de fichiers, serveur SMB (WinNT), serveur Novell, serveur d'impression, serveur FTP, serveur de courriel, serveur SQL, serveur POP, pare-feu, serveur de masquarade d'adresses, routeur, etc. Il fait ce que d'autres serveurs font, mais plus efficacement et de manière plus sûre. Les interfaces graphiques pour les utilisateurs sont agréables et sont les plus fonctionnelles lorsqu'il faut travailler en réseau. LINUX est désormais prêt pour "le bureau" de l'utilisateur particulier.

49.1.3 Sur quelles plate-formes LINUX fonctionne-t-il en ce compris les PCs ?

LINUX fonctionne sur :
- 386/486/Pentium (à 32 et 64 bits),
- AMD 64 bits,
- DEC Alpha 64 bits,
- Motorola 680x0, y compris les Commodore Amiga, Atari-ST/TT/Falcon et HP Apollo 68K,
- Sun Microsystems SPARC, ce qui comprend sun4c, sun4m, sun4d et sun4u. Les machines multiprocesseurs sont supportées pleinement en 64-bits sur UltraSPARC,
- ARM (Advanced Risc Machine),
- MIPS R3000/R4000, y compris les machines de Silicon Graphics,
- PowerPC,
- les mainframes IBM 390,
- ETRAX-100.

D'autres projets sont en cours à divers niveaux de développement (difficultés d'installation, absence de ressources graphiques). Chaque mois environ, une annonce est faite sur la capacité de LINUX à fonctionner sur des architectures plus ou moins ésotériques. Consultez *Linux Weekly News* (http ://lwn.net/) pour être au courant de ces développements.

49.1.4 Que signifient GNU/LINUX et LINUX ?

Voir aussi les sous-sections 49.1.1 et 49.2.2.

En 1984, la Free Software Foundation (FSF) envisagea de créer un système libre de la famille des UNIX. C'est parce que ces personnes ont fait un énorme effort de développement des paquets fondamentaux que le Logiciel Libre de type UNIX existe.

C'est grâce à ces personnes qu'une licence libre, complète, légale et applicable aux logiciels libres existe. Du fait que de nombreux composants d'une distribution LINUX proviennent des outils GNU développés avant le noyau LINUX lui-même, il serait injuste de réduire l'appelation d'une distribution à LINUX. Le terme GNU/LINUX est à la fois plus juste et plus respectueux de tout le travail accompli.

49.1.5 Quelles pages web dois-je consulter ?

Des milliers de page web sont dévolues à LINUX. Des dizaines de milliers de pages sont consacrées aux logiciels libres. La quantité d'information est prodigieuse :

- il y a trois sites pour LINUX "en général" :
  - la page d'*Alan Cox* http ://www.linux/org.uk/
  - *Linux Online* http ://www.linux.org/
  - *Linux International* http ://www.li.org/
- Pour le noyau, consultez :
  - *Linux Headquarters* http ://www.linuxhq.com/
- Un site important à visiter :
  - *FSF Home Page* http ://www.gnu.org/
    qui est le site de la Free Software Foundation et qui explique le but et le philosophie des logiciels libres (c'est-à-dire modifiables et redistribuables librement).
- Voici à présent des sites relatifs aux logiciels LINUX (libres et propriétaires) :
  - *Fresh Meat* http ://freshmeat.org/
  - *Source Forge* http ://www.sourceforge.net/
  - *Tu Cows* http ://linux.tucows.com/
  - *Scientific Applications for Linux* (SAL) http ://SAL.KachinaTech.COM/index.-shtml
- Les annonces relatives aux nouveaux logiciels se font sur :
  - *Fresh Meat* http ://freshmeat.net/
  - Les *Linux Weekly News* apportent une information à jour concernant différents problèmes à propos de Linux :
  - *Linux Weekly News* http ://lwn.net
- Les trois projets majeurs relatifs au bureau :
  - *Gnome Desktop* http ://www.gnome.org/
  - *KDE Desktop* http ://www.kde.org/
  - *GNUstep* http ://gnustep.org/

Ne vous arrêtez pas à cette liste d'adresses, il en existe des centaines d'autres.

49.1.6 Qu'est-ce que Debian, RedHat, SuSE, etc ? Expliquez-moi les différences.

Toutes les applications, les programmes serveurs et les utilitaires qui composent une machine LINUX complète sont des logiciels libres, compilés pour fonctionner avec un noyau LINUX. La plupart de ces programmes sont fonctionnels sous n'importe quel type d'UNIX.

Par conséquent, de nombreux efforts ont été (et continuent d'être) faits pour empaqueter tous les utilitaires nécessaires à un système UNIX, sous forme d'un florilège, appelé communément une distribution. Une distribution est gravée sur un ou plusieurs CD-ROM. Ainsi, des centaines de paquets sont-ils associés dans une distribution (par exemple, le serveur web Apache et le navigateur Mozilla en feront partie).

Voici quelques distributions populaires en 2005 :

- Debian GNU/Linux : http ://www.debian.org/
- Fedora http ://fedora.redhat.com/
- Gentoo Linux http ://www.gentoo.org/

- Mandriva http ://www.mandriva.com/
- RedHat http ://www.redhat.com/
- Slackware http ://www.slackware.com/
- SuSE http ://www.novell.com/fr-fr/linux/suse/
- TurboLinux http ://www.turbolinux.com/

Il existe environ 200 distributions LINUX répertoriées sur la site http ://distrowatch.com/. Certaines sont des routeurs tenant sur une disquette ou des disques de sauvegarde, d'autres sont des modifications de distributions populaires. Par exemple, Knoppix et Ubuntu sont des variantes de Debian. D'autres distributions sont orientées vers une thématique donnée : la sécurité, le travail en temps réel, etc.

49.1.7 Qui a developpé LINUX ?

LINUX a été très largement développé par la *Free Software Foundation* dont l'adresse web est http ://www.gnu.org/.

Le site intitulé *Orbiten Free Software Survey* http ://www.orbiten.org renseigne sur les différents contributeurs. Cette information est basée sur une analyse des paquets logiciels libres. Les listes qui suivent reprennent les 20 premiers contributeurs par quantité de code écrit :

Série	Auteur(s)	Octets	%	Projets
1	Free Software Foundation, Inc.	125565525	11,246	546
2	Sun Microsystems, Inc.	20663713	1,85	66
3	The Regents of the University of California	15192791	1,36	156
4	Gordon Matzigkeit	13599203	1,218	267
5	Paul Houle	11647591	1,043	1
6	Thomas G. Lane	8746848	0,783	17
7	The Massachusetts Insitute of Technology	8513597	0,762	38
8	Ulrich Drepper	6253344	0,56	142
9	Lyle Johnson	5906249	0,528	1
10	Peter Miller	5871392	0,525	3
11	Eric Young	5607745	0,502	48
12	login-belabas	5429114	0,486	2
13	Lucent technologies, Inc	4991582	0,447	5
14	Linus Torvalds	4898977	0,438	10
15	(uncredited-gdb)	4806436	0,43	1
16	Aladdin Enterprises	4580332	0,41	27
17	Tim Hudson	4454381	0,398	26
18	Carnegie Mellon University	4272613	0,382	23
19	James E. Wilson, Robert A. Koeneke	4272412	0,382	2
20	ID Software, Inc	4038969	0,361	1

La liste suivante contient les 20 contributeurs les plus performants par nombre de projets :

Série	Auteur(s)	Octets	%	Projets
1	Free Software Foundation, Inc.	125565525	11,246	546
2	Gordon Matzigkeit	13599203	1,218	267
3	The Regents of the University of California	15192791	1,36	156
4	Ulrich Drepper	6253344	0,56	142
5	Roland Mcgrath	2644911	0,236	99
6	Sun Microsystems, Inc.	20663713	1,85	66
7	RSA Data Security, Inc.	898817	0,08	59
8	Martijn Pieterse	452661	0,04	50
9	eric Young	5607745	0,502	48
10	login-vern	3499616	0,313	47
11	jot@cray	691862	0,061	47
12	Alfredo K. Kojima	280990	0,025	40
13	The Massachusetts Institute of Technology	8513597	0,762	38
14	Digital Equipment Corporation	2182333	0,195	37
15	David J. Mackenzie	337388	0,03	37
16	Rich Salz	365595	0,032	35
17	Jean-Loup Gailly	2256335	0,202	31
18	eggert@twinsun	387923	0,034	30
19	Josh Macdonald	1994755	0,178	28
20	Peter Mattis, Spencer Kimball	1981094	0,177	28

Les deux tables qui précèdent ne sont que des estimations assez grossières mais elles donnent une idée des origines variées des contributions.

49.1.8 Pourquoi ne devrais-je pas utiliser LINUX ?

Si vous êtes un individu isolé n'ayant personne pour vous assister en cas de problèmes et si vous n'êtes pas désireux d'apprendre les bases d'UNIX, vous ne devriez pas essayer LINUX.

49.2 LINUX, GNU et les licences.

Cette partie tente de répondre aux questions se rapportant à la nature des logiciels libres et à celle de GNU.

49.2.1 Qu'est-ce que la licence LINUX ?

Le noyau LINUX est distribué sous la Licence Publique Générale de GNU (GNU *General Public License* ou GNU GPL) dont une traduction non-officielle est reproduite au chapitre 50 du présent ouvrage. Le texte original anglais, qui est le seul à être officiel, peut être téléchargé sur http ://www.gnu.org/.

Pour la plupart, tous les autres logiciels d'une distribution LINUX typique sont sous la licence GNU GPL ou GNU LGPL (voir ci-dessous).

Il existe d'autres licences avec des perspectives commerciales ou morales. Leurs acronymes sont répertoriés ci-dessous sans que l'ordre n'ait de signification :

PD dans le domaine public (*Public Domain*),

Shareware il existe un droit d'auteur, sans restrictions, (partagiciel)

MIT licence du consortium MIT (similaire aux licences BSD mais sans publicité sur les conditions),

BSD droits d'auteurs des Régents de Berkeley (utilisés sur le code de BSD),

Artistic License identique à la licence artistique Perl,

FRS droits d'auteurs, redistribution libre, possibilité de restrictions sur la redistribution de sources modifiées,

GPL GNU General Public License,

GPL+LGPL GNU GPL et GPL sur les bibliothèques,

restricted moins libre que toutes les licences citées ci-dessus.

Le lecteur trouvera plus d'informations sur ces licenses sur le site *Metalab License List* à l'adresse ftp ://metalab.unc.edu/pub/linux/LICENSES.

49.2.2 Qu'est-ce que GNU ?

GNU (prononcez comme pour le mammifère) est un acronyme pour GNU is Not UNIX. Le logo de la Free Software Foundation est un gnou. GNU est un acronyme récursif.

Richard Stallman est le fondateur de la Free Software Foundation (FSF) et le créateur de la GNU GPL. Un des objectifs de la FSF est de promouvoir et de développer des alternatives aux logiciels propriétaires. Le projet GNU est un effort de création *ab initio* d'un système d'exploitation de type UNIX. Le projet fût lancé en 1984.

Lorsqu'ils sont sous la licence GNU GPL, les logiciels sont libres. Les logiciels GNU sont construits avec une qualité supérieure à celle de leurs équivalents propriétaires.

GNU est aussi devenu un mouvement dans le monde du logiciel. Quand le terme GNU est mentionné, il évoque l'assemblée de personnes extrêmement compétentes qui produisent des logiciels largement supérieurs en qualité à tout ce que peuvent produire des entreprises de grande taille qui réalisent des logiciels depuis de nombreuses années. Le terme GNU évoque aussi le développement ouvert, coopératif, étendu au monde entier, encourageant l'analyse par d'autres programmeurs, la cohérence et la portabilité. GNU est synonyme de réalisations aussi méticuleuses que possible garantissant des solutions durables plutôt que des corrections rapides. GNU mise sur la qualité plutôt que sur les expédients tape-à-l'oeil.

GNU pratique aussi une saine irrévérance à l'endroit des dates-butoirs et des agendas de mise à disposition des logiciels.

49.2.3 Pourquoi un logiciel GNU est-il meilleur qu'un logiciel propriétaire ?

Les logiciels propriétaires sont souvent considérés comme inférieurs dans le monde du logiciel libre pour plusieurs raisons :

- le processus de développement d'un logiciel propriétaire se fait à l'abri de tout examen minutieux externe,
- les utilisateurs sont incapables d'y ajouter de nouvelles propriétés,
- les utilisateurs ne peuvent pas corriger les bogues des logiciels propriétaires,
- ils ne peuvent pas partager leur logiciel propriétaire.

Le résultat de ces limitations est que les logiciels propriétaires :

- ne sont pas conformes aux bonnes normes des technologies de l'information,
- sont incompatibles avec d'autres logiciels propriétaires,
- sont bogués,
- ne peuvent être corrigés,
- coûtent plus qu'ils ne valent,

- peuvent effectuer des opérations à l'insu des utilisateurs,
- sont insécurisés,
- essayent d'être meilleurs que d'autres logiciels propriétaires sans rencontrer les véritables nécessités pratiques et techniques,
- engendrent une perte de temps par duplication de logiciels propriétaires concurrents,
- ne permettent pas de construire sur une base existante, en raison de leur caractère fermé par les licences propriétaires.

A l'inverse, les logiciels GNU sont ouverts et sont examinables de manière très minutieuse par les utilisateurs. Ceux-ci peuvent corriger les bogues librement et améliorer le logiciel pour leur propre usage et, par là-même, permettre aux autres utilisateurs de bénéficier des améliorations apportées. De nombreux développeurs dont l'expertise s'exerce dans des domaines différents collaborent afin de trouver la meilleure méthode pour améliorer les logiciels libres. Les normes académiques et industrielles sont prises en compte de manière à élaborer des logiciels libres cohérents et compatibles. L'effort de collaboration entre différents développeurs signifie que le code est partagé et que les efforts de correction ne sont pas doublés inutilement. Les utilisateurs ont une proximité très grande avec les développeurs, ce qui garantit une réactivité optimale pour effectuer des corrections. Les besoins des utilisateurs sont donc respectés. Du fait que le code source est visible par tous, les développeurs s'avèrent très consciencieux et le code est très propre.

Que le logiciel libre soit de qualité supérieure au logiciel propriétaire provient l'analyse continue du logiciel libre par d'autres développeurs. Le développement peut prendre plus de temps du fait que plusieurs personnes analysent le meilleur moyen de parvenir à une solution optimale. Le temps ainsi investi garantit un produit plus sûr.

Une autre raison à la supériorité des logiciels GNU est qu'ils sont écrits par des membres très qualifiés appartenant à des institutions académiques. Enfin, bon nombre de logiciels libres sont développés par des personnes insatisfaites de la solution commerciale. Leur plaisir à écrire un code efficace et propre dont ils sont responsables est une puissante motivation.

49.2.4 Expliquez-moi les restrictions de la licence libre GPL de GNU pour Linux ?

Voici un extrait de la licence GPL :

"Lorsque nous parlons de logiciels libres, nous nous référons à la liberté et non au prix. Nos licences GPL sont élaborées de manière telle à s'assurer que vous aurez la liberté de distribuer des copies des logiciels libres (et demander une contre-partie financière pour ce service, si vous le souhaitez), que vous recevrez les codes sources, que vous pourrez modifier les logiciels ou en utiliser des parties à incorporer dans de nouveaux programmes, et que vous serez au courant du fait que vous pourrez faire tout cela.

Pour protéger vos droits, nous devons établir des restrictions pour empêcher que quiconque refuse ces droits et vous demande d'y renoncer. Ces restrictions vous transfèrent des responsabilités si vous distribuez des copies de ces logiciels ou si vous les modifiez.

Par exemple, si vous distribuez des copies d'un logiciel libre, que ce soit gratuitement ou contre une contribution financière, vous devez donner tous les droits que vous avez sur le logiciel. Vous devez vous assurez que les personnes reçoivent aussi les codes sources. Par ailleurs, vous devez faire connaître les termes de la GPL de manière à ce que ces personnes connaissent leurs droits".

49.2.5 Si LINUX est libre, comment des sociétés commerciales peuvent-elles faire de l'argent en vendant des CD-Rom ?

Voir la sous-section "Ou puis-je trouver LINUX ?", à la page 613.

49.2.6 Que se passerait-il si LINUX Torvalds changeait les droits d'auteur sur le noyau ? Pourrait-il les vendre à une société commerciale ?

Cette situation est impossible. En raison des termes légaux de la GPL, la distribution de LINUX sous des droits d'auteurs différents requerrait le consentement de toutes les personnes (plus de 200) ayant contribué jusqu'ici à l'élaboration du code source du noyau. Ces personnes sont réparties dans le monde entier, ce qui fait qu'une demande de modification sur les droits du noyau est logistiquement infaisable. Même si cela se faisait, les nouveaux développeurs du noyau manifesteraient une méfiance à ce projet et continueraient à travailler au noyau. Ce nouveau noyau libre rencontrerait l'adhésion de si nombreux développeurs qu'il deviendrait la nouvelle norme, avec ou sans Linus.

49.2.7 Que se passerait-il si Linus Torvalds arrêtait son aide au développement de LINUX ? Et si le développement du noyau cessait ?

Il y a de très nombreux développeurs possédant une connaissance approfondie du travail réalisé par Linus. De manière très probable, un noyau dur de développeurs reprendrait la tâche, là où l'aurait laissée Linus. LINUX pourrait même être morcelé et les composantes réparties entre différentes équipes si un désaccord relatif à des problèmes de programmation se manifestait. Plus tard, le projet se reconstituerait. Cela s'est déjà produit avec d'autres paquets logiciels sans que ne se manifestent d'effets négatifs. En tout cas, cela n'affecterait pas l'utilisateur final, car un logiciel GNU est basé sur la cohérence et l'amélioration continuelle. De plus, cela n'affecte pas l'utilisateur final car celui-ci a choisi une distribution LINUX construite par des personnes qui ont déjà rencontré les problèmes que nous venons d'évoquer.

49.2.8 Quelle est la différence entre "logiciel libre" et "logiciel open source" ?

Le terme "*Open Source*" est un attrape-nigaud en raison du terme "*open*" (ouvert) qui induit une confusion avec "*free*" (libre). Le terme "*Open Source*" se rapporte à un logiciel propriétaire dont le code source peut en tout ou en partie accompagner le logiciel. Cependant, le code n'est pas libre dans la mesure où nul ne peut le modifier ni le redistribuer. Parfois, ce terme signifie "logiciel du domaine publique".

Les partisans de l'*Open Source* militent en faveur de la supériorité du modèle de développement *Open Source*.

Les partisans de GNU répugnent à utiliser le terme *Open Source*. Le terme "logiciel libre", au sens de la liberté de modification et de redistribution, est préférable et nécessite l'usage d'un droit d'auteur de la même nature que la GPL. Malheureusement, le terme "logiciel libre" demande une explication approfondie (et de ce point de vue n'entre pas dans la logique du *marketing*).

Les partisans des logiciels libres militent en faveur d'une responsabilité morale permettant au code source s'être disponible et redistribuable de manière à ce que ceux qui en bénéficient perpétuent le mouvement.

Le terme *shareware* ou partagiciel désigne un logiciel sous forme binaire totalement non-libre mais redistribuable. Il ne s'agit en aucune manière d'un *logiciel libre*.

49.3 Distributions LINUX.

Cette section traite de questions relatives à la manière dont les paquets logiciels sont regroupés et distribués. La question de savoir comment obtenir LINUX est abordée.

49.3.1 Si chacun modifie les sources en permanence, cela n'est-il pas préjudiciable à l'utilisateur ? Comment le système est-il protégé des bogues logiciels ?

En tant qu'utilisateur et à l'instar de ce que vous faites avec les logiciels Windows, vous ne téléchargez pas ceux qui n'ont pas été dûment testés. Quand vous obtenez LINUX, c'est le plus souvent sous la forme d'une distribution sur CDs ou DVD. Chacun des paquets a été sélectionné par le vendeur de la distribution comme étant un paquet d'origine, sous forme stable. C'est d'ailleurs là que réside la responsabilité de ceux qui créent des distributions.

Notez qu'aucune société commerciale ne surveille l'évolution de LINUX. Chaque contributeur s'emploie à la mission qu'il s'est assignée. Cependant, un paquet ne trouvera sa place dans une distribution que si quelqu'un estime qu'il sera utile. Quand un paquet est-il jugé utile? Essentiellement quand il a été employé sur une période suffisamment longue. Ainsi, seuls des paquets de bonne facture et testés sont inclus dans une distribution.

Les mainteneurs de paquets s'employent à garantir que des versions officielles sont téléchargeables à partir de leur site web et ils chargent les versions originales sur les serveurs FTP désignés.

Il n'y a donc pas de risque qu'une personne étrangère à la liste des mainteneurs puisse modifier un paquet original et altérer les noms des contributeurs du paquet.

Pour les paranoïaques qui soupçonnent que le paquet téléchargé n'est pas l'original déposé par les contributeurs, il existe une signature digitale associées au paquet. Les cas pour lesquels des vandales auraient substitué un paquet original par un paquet altéré sont en réalité très rares et parfaitement évitables.

49.3.2 Il y a de nombreuses distributions LINUX. Des problèmes d'incompatibilité sont-ils à prévoir ?

Voir aussi la question suivante.

Le noyau LINUX (version vanilla) en est à la version 2.6.16 au moment où ce texte est rédigé. Les seules autres versions stables sont les noyaux 2.4 (et 2.2). La version du noyau n'affecte pas l'utilisateur. Les programmes LINUX fonctionnent indépendamment de la version du noyau. Les versions de noyau ont des propriétes logicelles propres mais n'interfèrent pas avec les programmes utilitaires.

Chaque distribution possède son système de version. Au moment où ces lignes sont écrites, la dernière version stable de RedHat est l'Enterprise-Linux-4. Debian en est à la version stable 3.1 dite Sarge, Madriva à la version 10.2. Le changement de numérotation chez un distributeur provient du fait que de nouvelles versions de

paquets (ou bien une méthode d'installation mieux adaptée) ont été incluses. Parfois, les changement sont très subtils.

L'implémentation de la bibliothèque **C** LINUX est appelée `glibc`. Lorsque RedHat a sorti sa version 5.0 (en 1998), il y a eu un changement de l'ancienne bibilothèque `libc5` vers `glibc`. Etant donné que tous les paquets dépendent de cette bibliothèque, certains auraient pu penser que cela introduisait des incompatibilités. Cependant, plusieurs versions d'une même bibliothèque peuvent co-exister sur un même système. Cette transition n'a en réalité pas introduit d'incompatibilités. Du coup, d'autres distributeurs ont effectué le passage à `glibc` (en fait : `libc6`).

La communauté LINUX a aussi édité un document appelé le *système de fichiers normalisés* (SFN ou, FHS en anglais). La majorité des vendeurs essayent de se conformer à cette norme, ce qui fait que les systèmes LINUX sont très proches les uns des autres. Il n'y a donc pas de problèmes rhédibitoires de compatibilité entre distributions LINUX.

49.3.3 Un programme d'une distribution fonctionne-t-il sur une autre distribution ? A quel point les distributions sont-elles compatibles ?

Les différentes distributions sont très similaires et sont compatibles concernant les binaires (pourvu qu'ils correspondent au même type de processeur). Donc, les binaires LINUX compilés sur un système donné fonctionneront sur un autre système LINUX de même architecture matérielle. Ceci contraste singulièrement avec la situation des différents UNIX (par exemple, Sun et IRIX). Des utilitaires permettent de convertir un paquet destiné à une distribution de manière à installer ce dernier sur une autre distribution. Cependant, certaines distributions sont destinées à un matériel spécial ; par conséquent, leurs binaires ne fonctionnent que pour ce matériel. Néanmoins, tout logiciel écrit pour LINUX peut être recompilé sans modification pour une autre distribution LINUX et avec des modifications tout-à-fait mineures pour d'autres systèmes UNIX. La règle de base est la suivante : si vous disposez de trois paquets que vous devez faire fonctionner sur trois distributions différentes, il est simple de réaliser des ajustements. Si les paquets à ajuster se comptent pas dizaines, il y a un problème.

49.3.4 Quelle est la meilleure distribution ?

Si vous êtes tout débutant, Mandriva est une des distributions les plus populaires et elle est extrêmement facile à installer. RedHat est également bien supportée dans un contexte industriel.

Les qualités de quelques distributions bien connues et appréciées sont :

Mandriva Cette distribution est basée sur RedHat avec des paquets supplémentaires et une mise-à-jour raisonnablement aisée. Elle est rapidement devenue populaire en raison de la facilité d'installation et d'une très bonne détection du matériel. Système de paquets binaires `.rpm`.

Debian Il s'agit d'une distribution très performante. Complètement libre, très bien structurée, à développement communautaire exclusivement, elle est conforme aux normes les plus poussées. Nettement plus difficile à installer qu'une Mandriva, la distribution Debian possède un système de gestion de paquets largement supérieur. Cette distribution est connue pour son niveau technique d'excellence et sa stabilité légendaire. Système de paquets binaires `.deb`.

RedHat C'est une des distributions les plus populaires. Aujourd'hui commerciale (et axée sur les entreprises), elle a cédé une branche LINUX communautaire appelée Fedora. Système de paquets binaires `.rpm`.

Slackware C'est la première distribution LINUX. Elle est considérée comme celle qui intègre les paquets les plus récents. Difficile à installer et à gérer, elle est appréciée des connaisseurs. Paquets sources `.tar.gz`.

Gentoo Très avancée techniquement par son système de gestion de paquets appelé Portage, Gentoo combine les avantages de Debian et de BSD. Son installation est assez difficile (quoique richement documentée), car chaque étape demande une intervention en ligne de commandes. La responsabilité de l'administrateur est mise à contribution pour personnaliser le système à chaque niveau (d'où une grande flexibilité : compilation des paquets, optimisation selon l'architecture matérielle, contrôle des sémaphores de la variable USE, etc). Elle permet une maîtrise très poussée de tout le système. Système de paquets `.ebuild` qui appellent les sources par téléchargements sur l'internet pour les compiler en fonction des sémaphores fixés par l'administrateur.

Les distributions à paquets RPM (RedHat, Mandriva, etc.) ont ceci d'intéressant que presque tous les développeurs fournissent des paquets `.rpm` (le type de fichiers binaires fournit par RedHat). Les paquets Debian (`.deb`) sont généralement fournis mais moins souvent que les `.rpm`. En revanche, les paquets Debian sont principalement crées par des personnes appartenant à l'équipe de développement de Debian et donc elles adhèrent aux hauts standards de qualité de cette distribution.

TurboLinux, SuSE (désormais sous la houlette de Novell) et quelques autres distributions sont également très appréciées. Il existe des analyses de toutes ces distributions (*benchmarks*) sur l'internet.

Beaucoup d'autres distributions méritent d'être installées [...].

49.3.5 Où puis-je obtenir LINUX ?

Dès le moment où vous avez choisi votre distribution (voir la sous-section qui précède), vous devez la télécharger, l'acheter ou emprunter les CDs à quelqu'un. Les distributions commerciales peuvent contenir des logiciels propriétaires que vous ne souhaiterez peut-être pas et que de toute façon, vous ne pourrez pas redistribuer librement. Cependant, Gentoo, Fedora, Debian et Slackware sont commises à la liberté et n'incluront pas de logiciels non redistribuables. Donc, sentez-vous libre de redistribuer les CDs comme bon vous semble.

Notez que la GPL ne stipule pas que les logiciels doivent être gratuits. Il est permis de demander une contribution pour des frais de distribution, d'installation et de gestion des logiciels. Le terme libre ne signifie pas gratuité : il garantit seulement la non-interdiction de redistribuer et/ou de modifier un logiciel libre GNU.

Citons un miroir international pour les distributions Linux : *Metalab distributions mirror* ftp ://metalab.unc.edu/pub/Linux/distributions/. Consultez également les ressources mentionnées au chapitre 14, à la sous-section "Quelle page web dois-je consulter ?" à la page 605, et les entrées de l'index se rapportant aux sites web.

Le téléchargement à partir d'un site web par FTP peut être plus ou moins long selon le type de connexion dont vous disposez. Si votre connexion est à faible débit, renseignez-vous pour acquérir une distribution LINUX sur CDs ou DVD. Assurez-vous d'obtenir une version récente. Il ne faut jamais installer une version obsolète.

49.3.6 Comment puis-je installer LINUX ?

Les futurs utilisateurs de LINUX où qu'ils soient, ont besoin de renseignements pour savoir comment installer LINUX. Depuis le début, la communauté du Libre a produit une documentation fournie. Ou se trouve-t-elle ?

La plupart des distributions proposent des guides d'installation assez détaillés, c'est d'ailleurs pourquoi nous n'avons pas discuté le processus d'installation dans cet

ouvrage. Si vous naviguez parmi les fichiers des CDs ou du DVD, vous trouverez de la documentation. Par ailleurs, le site web de votre distribution vous propose une documentation complète.

Essayez aussi de voir ce que vous restitue l'internet lorsque vous faites une recherche sur "installation linux". Lisez le guide d'installation en détail. Il décrit chaque aspect du partitionnement, du *dual-boot*, et de bien d'autres points.

Soyez attentif au fait que chaque distribution possède sa propre méthode d'installation.

49.4 Aides pour LINUX.

Cette partie explique où trouver de l'aide.

49.4.1 Où peut-on obtenir de l'aide pour LINUX ? Mon logiciel propriétaire est "supporté" : comment LINUX peut-il être compétitif ?

Il existe une aide pour LINUX au sein de la communauté d'utilisateurs LINUX. Avec les systèmes commerciaux, les utilisateurs sont très réservés lorsqu'il s'agit de partager leur connaissances car ils sentent qu'ils ne possèdent rien, bien qu'ayant dépensé de l'argent pour acheter une ou plusieurs licences logicielles.

A l'inverse, les utilisateurs de LINUX sont très souvent prêts à s'entraider. Il est donc possible d'avoir, à partir de l'internet, une aide infiniment meilleure que cela n'est le cas avec des vendeurs de logiciels commerciaux. En majorité, les paquets contiennent des listes de courriel relatives aux développeurs. Ces derniers sont disponibles pour répondre à des questions venant des utilisateurs. Il existe des listes de diffusion par villes ou par régions sur lesquelles les réponses sont éditées quelques heures tout au plus après que les questions aient été posées. Les nouveaux utilisateurs de LINUX découvrent que l'aide abonde et, rapidement, ils participent aux discussions amicales à propos des problèmes informatiques qu'ils peuvent rencontrer.

Les groupes de discussions (*newsgroups*) fournissent de l'aide quand des problèmes à propos de LINUX se manifestent et une assistance est également fournie aux novices. Il existe de très nombreux groupes de discussions. L'audience y est plutôt très grande.

L'internet est aussi un outil d'aide très important pour LINUX. Du fait que les utilisateurs interagissent en permanence et échangent des trucs et astuces pour contourner des problèmes techniques, 99% des problèmes rencontrés par un utilisateur sont très probablement déjà documentés et ont trouvé une solution à laquelle on peut accéder via les archives des listes de diffusion.

Enfin, de nombreux professionnels fournissent également une assistance souvent tournées vers des solutions d'entreprises.

49.5 LINUX comparé à d'autres systèmes.

Cette partie discute les mérites des différents systèmes UNIX et de NT.

49.5.1 Quel est le UNIX le plus populaire dans le monde ?

Sans aucun doute, le nombre de machines fonctionnant sous LINUX est-il plusieurs fois plus grand que le nombre de machines ayant pour OS, UNIX.

49.5.2 Combien de LINUX existe-t-il ?

La réponse est difficile à établir. Une étude a montré qu'en 2004, LINUX représentait environ 4% des OS. Actuellement, Linux domine le marché des systèmes embarqués, si bien que ce système dépassera sous peu le total des autres systèmes d'exploitation. Il apparaît que le nombre d'utilisateurs de LINUX double chaque année. Il en découle un grand intérêt chez les particuliers, les entreprises, les Institutions, l'abonnement aux journaux spécialisés, la visite des sites web, l'attention dans les médias, les demandes de supports, la demande logicielle, etc.

Aujourd'hui environ 30% des serveurs web fonctionnent sous LINUX.

49.5.3 Quel est le coût d'installation et d'exploitation d'un système LINUX comparativement à d'autres systèmes propriétaires non-UNIX ?

Bien que LINUX soit libre, une bonne connaissance d'UNIX est requise pour installer et configurer un serveur de manière sûre. Ceci tend à vous prendre du temps et à élever les charges financières.

En revanche, un serveur Windows ou OS/2, par exemple, nécessite une licence.

De nombreux argumentaires utilisés pour établir le coût d'un serveur ne parviennent pas à prendre en compte ce qui se passe durant la vie complète d'un serveur. De ce fait, il existe des rapports aux conclusions opposées qui soit affirment que LINUX ne coûte rien, soit établissent que LINUX est inutilisable en raison du coût de l'expertise requise. La vérité est entre ces deux extrêmes. Le coût total d'un serveur inclut les aspects suivants :

- le coût de la licence du système d'exploitation,
- le coût des logiciels dédiés qui fournissent des fonctions non-supportées à l'origine par le système d'exploitation,
- le coût du matériel,
- la disponibilité du matériel utilisé et la compatibilité du système d'exploitation avec ce matériel,
- les coûts d'installation,
- les coûts d'entretien,
- les coûts liés à l'arrêt du serveur pour cause de bogues logiciels,
- les coûts liés à l'arrêt du serveur pour cause de problèmes de sécurité,
- les coûts de réparations,
- le prix des logiciels de mises-à-jour,
- le coût lié à l'activité multi-serveur : LINUX peut faire fonctionner simultanément sur la même machine différents services (fichiers, courriel, web), ce qui évite l'usage de serveurs dédiés. Il s'agit d'une économie substantielle.

Lorsque l'ensemble de ces facteurs est pris en compte, n'importe quelle entreprise devrait constater l'importante économie résultant du choix d'un serveur LINUX comparativement aux systèmes commerciaux.

49.5.4 Quel est le coût d'installation et d'exploitation d'un système LINUX comparativement à d'autres systèmes UNIX ?

Voir aussi la question précédente.

Les systèmes propriétaires UNIX ne sont pas aussi conviviaux que LINUX. Ce dernier système d'exploitation est beaucoup plus facile à entretenir que n'importe quel système UNIX commercial en raison d'un usage assez répandu et donc d'un accès à l'expertise qui s'en trouve facilité. LINUX présente aussi un projet de documentation

beaucoup plus convivial et développé que celui de tous les autres UNIX. Les interfaces utilisateurs et les commandes sont mieux construites.

Le résultat est qu'en dépit d'un fonctionnement aussi bon que celui de LINUX, les UNIX sont plus exigeants à entretenir.

Les systèmes UNIX qui fonctionnent sur du matériel spécialisé ne vous permettent jamais d'avoir un bon retour sur investissement (rapport coût/performance).

49.5.5 Comment comparer LINUX par rapport à d'autres systèmes en termes de performance ?

Typiquement, LINUX s'avère de 50 à 100% meilleur que d'autres systèmes d'exploitation sur le même matériel. Il n'existe aucune exception commerciale à cette règle concernant les PCs.

Il y a eu un grand nombre de tentatives maladroites pour montrer que LINUX est meilleur ou moins bon que d'autres plateformes. Jusqu'ici, il n'y a pas eu d'études concluantes dans un sens ou dans l'autre. Usuellement, ces études sont biaisées. Selon des tests réalisées de manière supposée indépendante, LINUX s'avère meilleur qu'NT en tant que serveur web, serveur de fichiers et serveur de base de données et ce, avec une marge appréciable.

En général, les améliorations en terme de performances d'une machine LINUX sont tout-à-fait perceptibles pour les utilisateurs et administrateurs. Il est spécialement notable que les accès au système de fichiers sont plus courts avec LINUX et l'écart se creuse encore lorsque le nombre de services augmente. Par ailleurs, LINUX excelle lorsqu'il est en charge de plusieurs services simultanés.

Il y a eu naguère des critiques sur la capacité de LINUX à fonctionner en mode multiprocesseur et sur le système de fichiers non-journalisé (depuis ext3, cela est d'ailleurs terminé). Ces point sont discutés ci-dessous.

Selon nous (en se basant sur notre expérience, des discussions et les nouveaux développements), les opérations critiques effectuées par LINUX sont scrupuleusement optimisées –bien plus que cela n'est exigé par des organismes commerciaux. Par conséquent, si votre matériel ne fonctionne pas de manière tout-à-fait optimale, les pertes de performances resteront marginales sous LINUX.

Il n'est pas forcément pertinent de débattre de ces aspects de rapidité lorsqu'on sait les autres avantages décisifs à l'actif de LINUX.

49.5.6 Qu'en est-il de SMP et des fichiers de journalisation ? LINUX est-il prêt pour l'entreprise ?

On a souvent reproché à Linux de ne pas bien fonctionner en mode multiprocesseur et par conséquent de ne pas être comparable à d'autres systèmes d'exploitation. Cela a été vrai mais ce n'est plus le cas depuis les noyaux 2.4 (c'est-à-dire depuis janvier 2001).

LINUX possède aussi un système de journalisation du système de fichiers appelé ReiserFS. Cela signifie qu'en cas de panne d'alimentation électrique, la probabilité est extrêment faible d'assister à une corruption du système de fichiers ou qu'une intervention manuelle doive avoir lieu pour résoudre la corruption de fichiers.

49.5.7 Est-il vrai que LINUX ne supporte que 2 Go de mémoire et 128 Mo de swap ?

LINUX supporte jusqu'à 64 Go de mémoire avec 1 Go de mémoire non-partagée par processus. Si vous devez avoir plus de mémoire que cela, vous devriez acquérir un processeur à 64 bits (DEC Alpha, Sun UltraSPARC, AMD 64, IA 64).

LINUX accepte autant de mémoire swap que vous le souhaitez depuis les noyaux 2.4.

49.5.8 Les UNIX ne sont-ils pas des antiquités ? Leur modèle de sécurité n'est-il pas obsolète ?

Les principes de base pour le développement des systèmes d'exploitation n'ont pas changé depuis plus de 40 ans. Seul le milieu académique s'intéresse à des modèles théoriques en sciences informatiques, les industries effectuant la mise en oeuvre.

Il existe de nombreux paradigmes de développement de systèmes d'exploitation, certains étant plus ou moins complexes, d'autres plus ou moins tournés vers une mise en oeuvre pratique. Parmi les systèmes d'exploitation pour serveurs, UNIX est sans aucun doute le modèle le plus flexible, le plus adaptable et celui qui autorise la plus grande manoeuvrabilité en termes de sécurité et de structure de fichiers.

49.5.9 LINUX et FreeBSD sont-ils comparables ?

FreeBSD (ainsi qu'OpenBSD et NetBSD) est une distribution similaire à LINUX dans la mesure où elle regroupe de nombreux paquets GNU. Pour la plupart, les paquets disponibles dans les distributions LINUX le sont aussi dans le cas de FreeBSD. FreeBSD n'est pas seulement un noyau mais aussi une distribution, un modèle de développement, un système d'exploitation normalisé et une infrastructure communautaire. FreeBSD devrait plutôt être comparé à Debian qu'à LINUX (au sens général).

La comparaison entre lex noyaux FreeBSD et LINUX met en évidence une différence d'organisation et d'implémentation concernant certaines fonctions. Le choix du noyau se fera selon votre champ d'activité. L'architecture de FreeBSD est considérée comme étant meilleure que celle de LINUX, mais il existe deux points forts en faveur de LINUX : le fait d'avoir été porté sur toutes les architectures matérielles modernes et le fait de supporter davantage de matériel. Il n'est pas sûr que les différences entre ces deux noyaux soient d'une grande importante dans la vie pratique de l'informaticien.

Un autre point important est que les mainteneurs de FreeBSD s'attachent à très bien sécuriser FreeBSD, ce que ne font pas forcément les vendeurs de distributions LINUX. D'où, le fait que FreeBSD est une alternative très intéressante en termes de sécurité.

La GPL met en évidence le fait que la license BSD permet à des entreprises commerciales d'utiliser BSD sans l'obligation de redistribuer les codes sources modifiés.

Cependant, ceci ne met pas en doute le fait qu'il vaut mieux utiliser BSD ou LINUX plutôt qu'un système propriétaire.

49.6 Migration vers LINUX.

49.6.1 Quels sont les principaux problèmes lorsqu'on migre de systèmes non-UNIX vers LINUX ?

En majorité, les entreprises tendent à sous-estimer leur dépendance vis-à-vis de Windows. Une entreprise fonctionne avec des individus qui apprennent ou s'échangent petit à petit des astuces informatiques et ce, sur de longues périodes. Pour beaucoup de personnes, la notion d'ordinateur est synonyme de *clic* sur le bouton "Sauver sous" ou sur le bouton "Mes documents". LINUX éloigne un peu l'utilisateur de cette optique (quoique les bureaux KDE, Gnome, XFCE présentent des interfaces graphiques très intuitives) de telle sorte que les personnes ayant appris à se servir de logiciels donnés sous un système d'exploitation donné peinent parfois à s'y retrouver dans un nouvel

environnement. Typiquement, une secrétaire prendra quelques semaines avant de se rassurer à nouveau devant son nouvel environnement LINUX. Un administrateur devra compter un peu plus de temps.

Alors que Windows n'offre pas une très large gamme d'options concernant les bureaux et les suites bureautiques, deux environnements de bureau LINUX seront peut-être aussi différents que le sont les bureaux de Windows 98 et d'Apple Macintosh. Les entreprises devront donc prendre leur décision avec soin avant de passer à LINUX notamment concernant la normalisation des environnements et la personnalisation des logiciels employés.

Notez que les documents Word, Excel et PowerPoint peuvent être lus (et modifiés après conversion) dans des applications LINUX, mais les formatages complexes ne seront pas convertis à 100%. Par exemple, les tailles de polices, les sauts de pages et les espaces peuvent ne pas être exactement respectés. [NdT : il est à observer que de grands progrès ont été accomplis avec la suite bureautique OpenOffice.org basée sur StarOffice de Sun.]

LINUX se comporte de manière transparente avec les partages de fichiers Windows, si bien que sur ce point, il n'y a pas de difficultés de migration vers LINUX.

Les interfaces graphiques écrites spécialement pour Windows sont difficiles à porter sous un système UNIX. Le projet Wine permet à présent d'utiliser des appplication Windows sous UNIX et, Borland a développé Kylix, une version LINUX de Delphi. Il y a beaucoup d'exemples de versions LINUX relatives aux langages Windows. Cependant, toute application interfacée avec des outils propriétaires et écrite dans un langage propriétaire est très difficilement portable. Le développeur qui effectue le portage devra être un expert en développement UNIX *et* en développement Windows. De telles personnes sont rares et leurs services sont chers.

49.6.2 Quels sont les principaux problèmes lorsqu'on migre des systèmes UNIX vers LINUX ?

Ce qui suit résulte de l'expérience acquise par Paul Sheer lors de la migration de trois entreprises de grande taille vers LINUX.

Les logiciels commerciaux UNIX qui ont été portés sous LINUX ne posent au pire que des problèmes mineurs. Vous pourrez compter sur une amélioration des performances et une réduction des coûts. Vous ne devriez donc pas hésiter à installer LINUX.

Les personnes gérant l'entreprise demanderont que l'acquisition de qualifications en LINUX se fasse par des formations spécifiques. On oublie souvent que les utilisateurs n'ont qu'une faible expérience d'UNIX. Par exemple, il est tout-à-fait faisable d'exécuter Apache (un paquet pour serveur web) sur des systèmes IRIX ou Sun Microsystems. Cependant, les responsables d'entreprises s'attendront à ce que leur personnel soit capable de configurer un "serveur web" LINUX de manière à éviter des frais de license relatifs aux logiciels de type serveurs web.

Il est important de savoir si le personnel a une compréhension réelle des réseaux TCP/IP et des systèmes UNIX dont vous dépendez de manière à éviter la technique des " essais et erreurs". Fondamentalement, LINUX est un système UNIX, qui plus est très convivial. Aussi, les difficultés avec LINUX ne sont-elles pas plus grandes qu'avec n'importe quel autre système UNIX.

S'il s'averait que les compétences UNIX du personnel de l'entreprise migrant sous Linux sont incomplètes, un bon livre devrait permettre de résoudre bien des questions.

Beaucoup d'entreprises développent aussi des applications "maison" spécifiques à leurs besoins. Dans ce cadre, le premier souci des développeurs consiste à faire tourner ces applications sous le nouvel environnement. C'est souvent là que le code se comporte de manière inattendue s'il a été mal écrit initialement. Dans ce cas, il est essentiel de louer les services d'un développeur expérimenté qui est à l'aise avec les outils de compilation de GCC.

Les applications UNIX de l'entreprise qui sont bien écrites (en ce compris les applications graphiques) tolèrent un portage aisé sous LINUX et, bien sûr, sous d'autres UNIX.

49.6.3 Comment procéderait un superviseur après avoir pris la décision de migrer vers LINUX ?

Avant l'installation de quelque machine que ce soit sous LINUX, il est nécessaire d'identifier ce que fait chaque utilisateur avec son ordinateur. Ceci peut paraître fastidieux mais cela permet de capitaliser d'importants renseignements. Si vous recourrez à des applications personnalisées, il est nécessaire d'identifier le rôle de ces applications et établir une liste de leurs spécifications et de leurs champs d'application.

L'étape suivante consiste à encourager les pratiques d'interopérabilité. Il se peut que votre entreprise ne puisse migrer de suite sous LINUX mais il est possible de faire l'économie de bien des efforts en anticipant un maximum de possibilités. Par exemple : faire en sorte qu'une police de caractères soit sauvée dans un format qui ne se limite pas à un paquet logiciel donné, c'est-à-dire un format portable.

Vous devriez commencer par limiter l'usage de programme ou d'outils qui n'ont pas d'équivalents UNIX. Les serveurs SMTP et POP/IMAP sont un standard de l'internet et ils peuvent être remplacés par des serveurs Linux. Les serveurs de fichiers SMB peuvent être remplacés par des serveurs Samba sous LINUX. Il y a aussi les comptes courriel accessibles depuis le web et les logiciels de travail en groupe (collecticiel) qui fonctionnnent sous LINUX et qui sont accessibles via les clients web comme Mozilla, Firefox, Internet Explorer, etc. Il existe aussi des traitements de texte qui existent à la fois en version Windows et UNIX et dont les opérations se font de manière identique sur les deux OS.

Il faudra amener vos développeurs à tester leurs pages web sur Mozilla/Firefox aussi bien que sur Internet Explorer. N'acceptez pas les outils de développement qui seraient liés de trop près à un système d'exploitation donné et qui ne possèdent pas d'équivalents UNIX. Il existe des outils de développement libres qui sont plus efficaces que les *Environnements de Développement Intégré* (appelés IDE ou *Integrated Development Environment*). Utilisez-les de manière préférentielle. Si votre entreprise fait du développement avec un langage compilable (par opposition aux langages interprétés comme bash ou Python, par exemple), vérifiez que vos développeurs s'assurent du fait que la compilation s'effectue proprement avec diverses marques de compilateurs. Ceci n'aura pas seulement pour effet d'améliorer la qualité du code mais aussi sa portabilité.

Soyez conscient du fait que votre personnel pourrait utiliser milles excuses pour ne pas avoir à apprendre quelque chose de nouveau. Rendez disponibles les ouvrages nécessaires. Identifiez les problèmes courants et mettez en place les techniques pour les résoudre rapidement et efficacement. Faites l'apprentissage des capacités de LINUX en lisant les publications sur l'internet. Un gestionnaire dans une entreprise qui n'est pas préparé à cela ne devrait pas attendre de son équipe de faire mieux que lui.

49.7 Aspects techniques.

Cette partie traite de diverses questions spécifiques et techniques.

49.7.1 Les CDs LINUX sont-ils lisibles avec MS-Windows ?

Oui. Vous pouvez naviguez dans la documentation d'installation sur le CD ou le DVD en utilisant Internet Explorer (ou Firefox pour Windows). Les logiciels LINUX s'accordent très bien des formats de disquettes Windows et du format de CD ISO9660.

49.7.2 Puis-je utiliser LINUX et Windows sur la même machine ?

Oui. Linux et Windows peuvent très bien occuper des partitions différentes d'un même disque dur. Cependant, Windows doit se trouver sur une partition primaire. A l'amorçage, une invite vous proposera le choix du système d'exploitation sur lequel vous désirez travailler.

49.7.3 De combien d'espace dois-je disposer pour installer LINUX ?

Une distribution typique avec un système graphique **X**-Window (l'environnement graphique d'UNIX) occupe environ 1 Go. Un serveur en réseau qui ne requiert pas **X** occupe typiquement de 100 à 300 Mo. LINUX peut fonctionner sur une disquette –soit 1,4 Mo– en permettant divers services réseau.

49.7.4 Quelles sont les exigences en matériel ?

LINUX fontionne sur différentes plate-formes matérielles, comme nous l'avons vu ci-avant. En général, et de nos jours, les utilisateurs achètent un PC d'entrée de gamme avec 256 Mo de RAM ce qui permet aisément de tirer parti du Système **X**-Window (l'environnement graphique d'UNIX).

Un Pentium-II à 300 MHz avec 64 Mo de RAM et une carte graphique de 2Mo consitute déjà une bonne machine LINUX (résolution d'écran 1024x768 en 16 bpp). Un Go de disque dur sera nécessaire.

Si vous utilisez du matériel de plus faible capacité, une machine susceptible de fonctionner en mode graphique avec **X** ne devrait pas contenir un processeur inférieur à 100 MHz (Intel 486) et 8 Mo de RAM. Les serveurs réseau peuvent fonctionner avec 4 Mo de RAM et une architecture 386 pour 200 Mo de disque dur. Notez que cela exigera un effort important de configuration (ce qui n'est pas recommandé pour débuter) et que les distributions récentes avec des programmes compilés pour Pentium ne fonctionneront pas sur des architectures 386. Si vous souhaitez utiliser un 386, il faudra recompiler votre propre noyau et divers paquets.

49.7.5 Quel matériel est-il supporté ? Mes cartes son, vidéo et réseau fonctionneront-elles ?

A 95 %, le matériel pour PC est fonctionnel avec LINUX. En général, le matériel associés aux noms de marques commerciales populaires sera opérationnel sur LINUX. [...] Des distributions comme SuSE, Mandriva, Gentoo, Knoppix, Redhat, Fedora permettent une détection aisée du matériel. Néanmoins pour vérifier la compatibilité matérielle, vérifiez sur le site http ://www.tldp.org/HOWTO/Hardware-HOWTO/ ou http ://www.freenix.fr/unix/linux/HOWTO/Hardware-HOWTO.html.

Il se peut que ces sites ne soient pas totalement à jour.

49.7.6 Puis-je consulter mes fichiers Windows, OS/2 et MS-DOS sous LINUX ?

LINUX peut lire et écrire sur les systèmes de fichiers de ces OS. Dans le cas du système de fichiers NTFS de Windows XP, il faut néanmoins utiliser un noyau 2.6 pour réaliser des opérations d'écriture. Par ailleurs, LINUX supporte les systèmes de fichiers d'autres UNIX, d'Amiga et d'OS/2.

49.7.7 Puis-je exécuter des programmes DOS sous LINUX ?

LINUX possède un émulateur avancé pour DOS. La plupart des applications DOS 16-bits et 32-bits pourront être exécutées. De même pour un grand nombre de jeux fonctionnant sous DOS.

Le paquet qui permet l'émulation de DOS sous LINUX s'appelle `dosemu`. Il exécute les applications DOS de manière plus rapide que DOS car l'accès au système de fichiers LINUX et les appels systèmes sont plus rapide que ceux effectués par DOS lui-même.

`dosemu` peut aussi être exécuté dans un environnement **X** tout comme DOS peut être exécuté sous Windows.

49.7.8 Puis-je recompiler des programmes Windows sous Linux ?

Oui. WineLib est une partie du paquet Wine (voir si-dessus) et permet aux applications **C** de Windows d'être recompilées pour travailler sous LINUX. Apparemment ceci fonctionne très bien et aucun changement du code source n'est nécessaire. Notez qu'XWine est une interface graphique pour Wine. Voir aussi Bochs.

49.7.9 Puis-je exécuter des programmes Windows sous Linux ?

Oui et non.

Il existe des émulateurs commerciaux qui permettent de faire fonctionner une machine Windows 98. Depuis peu, Win4Lin permet de faire fonctionner Windows 2000 et Windows XP sous Linux. Cela peut entraîner des ralentissements d'exécution et, par ailleurs, il vous faut encore une licence pour Windows.

Avec le projet Wine (et l'interface graphique XWine) vous disposerez d'un émulateur qui est une alternative libre à Windows, les binaires en 16 ou 32 bits sont exécutés sans perte significative de performances. Le programme est développé depuis de nombreuses années et la majorité des programmes Windows simples sont exécutés sans erreur.

Cela signifie que vous pouvez exécuter Minesweep sous LINUX, le programme apparaissant dans l'environnement **X** avec vos autres applications LINUX. Le programme apparaît exactement comme c'est le cas sous Windows. Vous ne devez plus acheter Windows. Vous pourrez également copier et coller des données entre les applications Windows et LINUX. Cependant, de nombreuxes applications écrites pour Windows, complexes et de grande taille ne fonctionnent pas correctement sous LINUX et "plantent". Sous émulation, Microsoft Office 2000 est connu pour fonctionner de manière stable.

De nombreuses jeux Windows fonctionnent parfaitement bien sous LINUX par le truchement d'une émulation, y compris des applications graphiques avec accélaration 3D.

Consultez le site *Wine Headquarters* sur http ://www.winehq.com/faq.html pour davantage d'informations.

49.7.10 J'ai entendu que LINUX ne souffre pas de l'attaque par virus. Est-il vrai qu'il n'y a pas de protection anti-virus sur UNIX ?

Un virus est un programme qui se réplique en modifiant le système sur lequel il est exécuté. Il peut produire d'autres dommages. Les virus sont des programmes de petite

taille qui exploitent l'ingéniérie sociale, la logistique et la facilité qu'ont les ordinateurs à effectuer aveuglément des tâches indésirables.

Du fait qu'un système UNIX ne possèdent pas à l'origine cette facilité d'exécution, les virus adressés à UNIX ou LINUX ne peuvent pas être exécutés inconsciemment par les utilisateurs (et encore, s'il sétaient exécutés, seraient-ils confinés à la session du dit-utilisateur sans endommager le système dans son ensemble). Ainsi, de manière intrinsèque, UNIX restreint les droits d'accès aux fichiers en dehors de l'espace utilisateur, de sorte qu'un virus n'a rien à infecter.

Cependant, bien que LINUX ne puisse exécuter de lui-même un virus, il peut être capable de passer un virus destiné à une machine Windows si la machine LINUX agit comme serveur de courriels ou de fichiers. Pour éviter ce type de problème, de nombreux programmes de détection de virus sont disponibles. Ceci désigne donc les "antivirus pour Linux".

En revanche, la mauvaise configuration d'une machine permet parfois à un "cracker" intelligent d'attaquer cette machine et d'éventuellement y gagner des droits importants. Ce cracker essayera donc d'attaquer d'autres machines en utilisant des programmes particuliers. Le cracker peut aller plus loin en forçant les machines compromises à exécuter des programmes indésirables. A ce stade, les programmes indésirables seront des "vers", des programmes contrecarrant la sécurité et capables d'exploiter un trou de sécurité récursivement sur un réseau. Voir ci-dessous.

Ultérieurement à l'attaque dont nous venons de parler, des utilisateurs en grand nombre peuvent utiliser la même application de bureau propriétaire qui présente partout la même vulnérabilité. Si vous utilisez LINUX, seul l'espace utilisateur sera touché ; pas LINUX en soi.

Rappelez-vous qu'avec une compréhension minimale de LINUX, il est facile de détecter et réparer un système corrompu sans avoir à effectuer une opération comme la réinstallation d'un système complet ou l'achat et la mise-à-jour d'antivirus.

49.7.11 LINUX est-il aussi sécurisé que les autres serveurs ?

LINUX est aussi sécurisé que les autres systèmes UNIX. Différentes questions doivent être abordées en ce qui concerne la sécurtié de LINUX.

Du fait que les logiciels GNU sont libres, tout programmeur peut facilement étudier le code source des services critiques pour le système.

D'un côté, n'importe quel programmeur peut détecter une vulnérabilité qui peut être exploitée pour compromettre la sécurité d'un système. Cela peut conduire certaines personnes à conclure que LINUX est un système moins sécurisé car des trous de sécurité peuvent être détectés par des individus mal intentionnés.

D'un autre côté, d'autres programmeurs peuvent détecter des vulnérabilités et avertir les développeurs du programme de manière à ce que ceux-ci corrigent rapidement la (ou les) failles. Le terme "rapidement" signifie très souvent : de l'ordre de quelques heures après l'identification de la vulnérabilité. Cela peut conduire à penser que LINUX est donc plus sécurisé puisque les vulnérabilités sont déclarées ouvertement, identifiées et corrigées rapidement.

Beaucoup de personnes préfèrent avoir accès au code source pour contribuer à le sécuriser. Elles préfèrent aussi savoir les opérations que réalise tel ou tel logiciel.

Un autre problème avec les serveurs LINUX est qu'ils peuvent être configurés par des personnes qui ne prennent pas tout le soin nécessaire ni le temps de suivre les guides élémentaires de sécurité, même si ces conseils de sécurité sont largement diffusés et très faciles à suivre. Ces serveurs sont des canards boîteux qui subissent inévitablement des attaques.

En outre, quand un trou de sécurité est découvert, certains administrateurs système n'observent pas toujours les consignes annoncées par la communauté LINUX. En ne mettant pas leur système à jour , ils ouvrent la porte aux opportunistes.

Vous pouvez sécuriser très fortement votre système en suivant des règles simples, comme par exemple, en étant attentif aux services en cours, en ne permettant pas aux mots de passe d'être compromis et en installant des utilitaires pour fermer la porte à d'éventuelles vulnérabilités.

En raison du caractère communautaire associés à la "méthode" LINUX, il existe une franchise et une honnêteté naturelles vis-à-vis des problèmes de sécurité. Il n'y a pas de cas où des mainteneurs de logiciels puissent fermer les yeux sur des problèmes de sécurité pour des raisons commerciales. C'est pourquoi vous pouvez faire davantage confiance à LINUX qu'à toute autre institution commerciale qui pense qu'il y a beaucoup à perdre par la révélation des vulnérabilités de leurs logiciels.

Chapitre 50

La licence publique générale GNU.

Le texte qui suit, parce qu'il n'est pas en anglais, n'est pas officiel. Il est extrait *in extenso* de http ://www.linux-france.org/article/these/gpl.html. Seule la version anglaise est légale. Le lecteur se rapportera donc à http ://www.gnu.org/copyleft/gpl.html. Le seul motif pour mentionner un texte français est d'aider le lecteur francophone à mieux comprendre la GPL : il s'agit de la version 2 (Juin 1991)

50.1 Preambule.

Les licences d'utilisation de la plupart des programmes sont définies pour limiter ou supprimer toute liberté à l'utilisateur. À l'inverse, la Licence Publique Générale (*General Public License*) est destinée à vous garantir la liberté de partager et de modifier les logiciels libres, et de s'assurer que ces logiciels sont effectivement accessibles à tout utilisateur.

Cette Licence Publique Générale s'applique à la plupart des programmes de la *Free Software Foundation*, comme à tout autre programme dont l'auteur l'aura décidé (d'autres logiciels de la FSF sont couverts pour leur part par la Licence Publique Générale pour Bibliothèques GNU (LGPL)). Vous pouvez aussi appliquer les termes de cette Licence à vos propres programmes, si vous le désirez.

Liberté des logiciels ne signifie pas nécessairement gratuité. Notre Licence est conçue pour vous assurer la liberté de distribuer des copies des programmes, gratuitement ou non, de recevoir le code source ou de pouvoir l'obtenir, de modifier les programmes ou d'en utiliser des éléments dans de nouveaux programmes libres, en sachant que vous y êtes autorisé.

Afin de garantir ces droits, nous avons dû introduire des restrictions interdisant à quiconque de vous les refuser ou de vous demander d'y renoncer. Ces restrictions vous imposent en retour certaines obligations si vous distribuez ou

modifiez des copies de programmes protégés par la Licence. En d'autre termes, il vous incombera en ce cas de :

* transmettre aux destinataires tous les droits que vous possédez,

* expédier aux destinataires le code source ou bien tenir celui-ci à leur disposition,

* leur remettre cette Licence afin qu'ils prennent connaissance de leurs droits.

Nous protégeons vos droits de deux façons : d'abord par le droit de copie du logiciel, ensuite par la remise de cette Licence qui vous autorise légalement à copier, distribuer et/ou modifier le logiciel.

En outre, pour protéger chaque auteur ainsi que la FSF, nous affirmons solennellement que le programme concerné ne fait l'objet d'aucune garantie. Si un tiers le modifie puis le redistribue, tous ceux qui en recevront une copie doivent savoir qu'il ne s'agit pas de l'original afin qu'une copie défectueuse n'entache pas la réputation de l'auteur du logiciel.

Enfin, tout programme libre est sans cesse menacé par des dépôts de brevets. Nous souhaitons à tout prix éviter que des distributeurs puissent déposer des brevets sur les Logiciels Libres pour leur propre compte. Pour éviter cela, nous stipulons bien que tout dépôt éventuel de brevet doit accorder expressément à tout un chacun le libre usage du produit.

Les dispositions précises et les conditions de copie, de distribution et de modification de nos logiciels sont les suivantes :

50.2 Stipulations et conditions relatives à la copie, la distribution et la modification GNU.

Article 0 La présente Licence s'applique à tout Programme (ou autre travail) où figure une note, placée par le détenteur des droits, stipulant que ledit Programme ou travail peut être distribué selon les termes de la présente Licence. Le terme Programme désigne aussi bien le Programme lui-même que tout travail qui en est dérivé selon la loi, c'est-à-dire tout ouvrage reproduisant le Programme ou une partie de celui-ci, à l'identique ou bien modifié, et/ou traduit dans une autre langue (la traduction est considérée comme une modification). Chaque personne concernée par la Licence Publique Générale sera désignée par le terme Vous.

Les activités autres que copie, distribution et modification ne sont pas couvertes par la présente Licence et sortent de son cadre. Rien ne restreint l'utilisation du Programme et les données issues de celui-ci ne sont couvertes que si leur contenu constitue un travail basé sur le logiciel (indépendemment du fait d'avoir été réalisé en lançant le Programme). Tout dépend de ce que le Programme est censé produire.

Article 1. Vous pouvez copier et distribuer des copies conformes du code source du Programme, tel que Vous l'avez reçu, sur n'importe quel support, à condition de placer sur chaque copie un copyright approprié et une restriction de garantie, de ne pas modifier ou omettre toutes les stipulations se référant à la présente Licence et à la limitation de garantie, et de fournir avec toute copie du Programme un exemplaire de la Licence.

Vous pouvez demander une rétribution financière pour la réalisation de la copie et demeurez libre de proposer une garantie assurée par vos soins,

moyennant finances.

Article 2. Vous pouvez modifier votre copie ou vos copies du Programme ou partie de celui-ci, ou d'un travail basé sur ce Programme, et copier et distribuer ces modifications selon les termes de l'article 1, à condition de Vous conformer également aux conditions suivantes :

- a) Ajouter aux fichiers modifiés l'indication très claire des modifications effectuées, ainsi que la date de chaque changement.
- b) Distribuer sous les termes de la Licence Publique Générale l'ensemble de toute réalisation contenant tout ou partie du Programme, avec ou sans modifications.
- c) Si le Programme modifié lit des commandes de manière interactive lors de son exécution, faire en sorte qu'il affiche, lors d'une invocation ordinaire, le copyright approprié en indiquant clairement la limitation de garantie (ou la garantie que Vous Vous engagez à fournir Vous-même), qu'il stipule que tout utilisateur peut librement redistribuer le Programme selon les conditions de la Licence Publique Générale GNU, et qu'il montre à tout utilisateur comment lire une copie de celle-ci (exception : si le Programme original est interactif mais n'affiche pas un tel message en temps normal, tout travail dérivé de ce Programme ne sera pas non plus contraint de l'afficher).

Toutes ces conditions s'appliquent à l'ensemble des modifications. Si des éléments identifiables de ce travail ne sont pas dérivés du Programme et peuvent être raisonnablement considérés comme indépendants, la présente Licence ne s'applique pas à ces éléments lorsque Vous les distribuez seuls. Mais, si Vous distribuez ces mêmes éléments comme partie d'un ensemble cohérent dont le reste est basé sur un Programme soumis à la Licence, ils lui sont également soumis, et la Licence s'étend ainsi à l'ensemble du produit, quel qu'en soit l'auteur. Cet article n'a pas pour but de s'approprier ou de contester vos droits sur un travail entièrement réalisé par Vous, mais plutôt d'ouvrir droit à un contrôle de la libre distribution de tout travail dérivé ou collectif basé sur le Programme.
En outre, toute fusion d'un autre travail, non basé sur le Programme, avec le Programme (ou avec un travail dérivé de ce dernier), effectuée sur un support de stockage ou de distribution, ne fait pas tomber cet autre travail sous le contrôle de la Licence.

Article 3. Vous pouvez copier et distribuer le Programme (ou tout travail dérivé selon les conditions énoncées dans l'article 1) sous forme de code objet ou exécutable, selon les termes des articles 0 et 1, à condition de respecter l'une des clauses suivantes :

- a) Fournir le code source complet du Programme, sous une forme lisible par un ordinateur et selon les termes des articles 0 et 1, sur un support habituellement utilisé pour l'échange de données ; ou, *
- b) Faire une offre écrite, valable pendant au moins trois ans, prévoyant de donner à tout tiers qui en fera la demande une copie, sous forme lisible par un ordinateur, du code source correspondant, pour un tarif n'excédant pas le coût de la copie, selon les termes des articles 0 et 1, sur un support couramment utilisé pour l'échange de données informatiques ou,
- c) Informer le destinataire de l'endroit où le code source peut être obtenu (cette solution n'est recevable que dans le cas d'une distribution non

commerciale, et uniquement si Vous avez reçu le Programme sous forme de code objet ou exécutable avec l'offre prévue à l'alinéa b ci-dessus).

Le code source d'un travail désigne la forme de cet ouvrage sous laquelle les modifications sont les plus aisées. Sont ainsi désignés la totalité du code source de tous les modules composant un Programme exécutable, de même que tout fichier de définition associé, ainsi que les scripts utilisés pour effectuer la compilation et l'installation du Programme exécutable. Toutefois, l'environnement standard de développement du système d'exploitation mis en oeuvre (source ou binaire) – compilateurs, bibliothèques, noyau, etc. – constitue une exception, sauf si ces éléments sont diffusés en même temps que le Programme exécutable.

Si la distribution de l'exécutable ou du code objet consiste à offrir un accès permettant de copier le Programme depuis un endroit particulier, l'offre d'un accès équivalent pour se procurer le code source au même endroit est considéré comme une distribution de ce code source, même si l'utilisateur choisit de ne pas profiter de cette offre.

Article 4. Vous ne pouvez pas copier, modifier, céder, déposer ou distribuer le Programme d'une autre manière que l'autorise la Licence Publique Générale. Toute tentative de ce type annule immédiatement vos droits d'utilisation du Programme sous cette Licence. Toutefois, les tiers ayant reçu de Vous des copies du Programme ou le droit d'utiliser ces copies continueront à bénéficier de leur droit d'utilisation tant qu'ils respecteront pleinement les conditions de la Licence.

Article 5. Ne l'ayant pas signée, Vous n'êtes pas obligé d'accepter cette Licence. Cependant, rien d'autre ne Vous autorise à modifier ou distribuer le Programme ou quelque travaux dérivés : la loi l'interdit tant que Vous n'acceptez pas les termes de cette Licence. En conséquence, en modifiant ou en distribuant le Programme (ou tout travail basé sur lui), Vous acceptez implicitement tous les termes et conditions de cette Licence.

Article 6. La diffusion d'un Programme (ou de tout travail dérivé) suppose l'envoi simultané d'une licence autorisant la copie, la distribution ou la modification du Programme, aux termes et conditions de la Licence. Vous n'avez pas le droit d'imposer de restrictions supplémentaires aux droits transmis au destinataire. Vous n'êtes pas responsable du respect de la Licence par un tiers.

Article 7. Si, à la suite d'une décision de Justice, d'une plainte en contrefaçon ou pour toute autre raison (liée ou non à la contrefaçon), des conditions Vous sont imposées (que ce soit par ordonnance, accord amiable ou autre) qui se révèlent incompatibles avec les termes de la présente Licence, Vous n'êtes pas pour autant dégagé des obligations liées à celle-ci : si Vous ne pouvez concilier vos obligations légales ou autres avec les conditions de cette Licence, Vous ne devez pas distribuer le Programme.

Si une partie quelconque de cet article est invalidée ou inapplicable pour quelque raison que ce soit, le reste de l'article continue de s'appliquer et l'intégralité de l'article s'appliquera en toute autre circonstance.

Le présent article n'a pas pour but de Vous pousser à enfreindre des droits ou des dispositions légales ni en contester la validité ; son seul objectif est de protéger l'intégrité du système de distribution du Logiciel Libre. De nombreuses personnes ont généreusement contribué à la large gamme de

Programmes distribuée de cette façon en toute confiance ; il appartient à chaque auteur/donateur de décider de diffuser ses Programmes selon les critères de son choix.

Article 8. Si la distribution et/ou l'utilisation du Programme est limitée dans certains pays par des brevets ou des droits sur des interfaces, le détenteur original des droits qui place le Programme sous la Licence Publique Générale peut ajouter explicitement une clause de limitation géographique excluant ces pays. Dans ce cas, cette clause devient une partie intégrante de la Licence.

Article 9. La Free Software Foundation se réserve le droit de publier périodiquement des mises à jour ou de nouvelles versions de la Licence. Rédigées dans le même esprit que la présente version, elles seront cependant susceptibles d'en modifier certains détails à mesure que de nouveaux problèmes se font jour.

Chaque version possède un numéro distinct. Si le Programme précise un numéro de version de cette Licence et « toute version ultérieure », Vous avez le choix de suivre les termes et conditions de cette version ou de toute autre version plus récente publiée par la Free Software Foundation. Si le Programme ne spécifie aucun numéro de version, Vous pouvez alors choisir l'une quelconque des versions publiées par la Free Software Foundation.

Article 10. Si Vous désirez incorporer des éléments du Programme dans d'autres Programmes libres dont les conditions de distribution diffèrent, Vous devez écrire à l'auteur pour lui en demander la permission. Pour ce qui est des Programmes directement déposés par la Free Software Foundation, écrivez-nous : une exception est toujours envisageable. Notre décision sera basée sur notre volonté de préserver la liberté de notre Programme ou de ses dérivés et celle de promouvoir le partage et la réutilisation du logiciel en général.

Article 11. Parce que l'utilisation de ce Programme est libre et gratuite, aucune garantie n'est fournie, comme le permet la loi. Sauf mention écrite, les détenteurs du copyright et/ou les tiers fournissent le Programme en l'état, sans aucune sorte de garantie explicite ou implicite, y compris les garanties de commercialisation ou d'adaptation dans un but particulier. Vous assumez tous les risques quant à la qualité et aux effets du Programme. Si le Programme est défectueux, Vous assumez le coût de tous les services, corrections ou réparations nécessaires.

Article 12. Sauf lorsqu'explicitement prévu par la Loi ou accepté par écrit, ni le détenteur des droits, ni quiconque autorisé à modifier et/ou redistribuer le Programme comme il est permis ci-dessus ne pourra être tenu pour responsable de tout dommage direct, indirect, secondaire ou accessoire (pertes financières dues au manque à gagner, à l'interruption d'activités ou à la perte de données, etc., découlant de l'utilisation du Programme ou de l'impossibilité d'utiliser celui-ci).

50.3 Comment appliquer ces directives à vos nouveaux programmes.

Si vous développez un nouveau programme et désirez en faire bénéficier tout un chacun, la meilleure méthode est d'en faire un Logiciel Libre que tout le monde pourra redistribuer et modifier selon les termes de la Licence Publique Générale.

Pour cela, insérez les indications suivantes dans votre programme (il est préférable et plus sûr de les faire figurer au début de chaque fichier source ; dans tous les cas, chaque module source devra comporter au minimum la ligne de « copyright » et indiquer où résident toutes les autres indications) : ((une ligne pour donner le nom du programme et donner une idée de sa finalité)) Copyright (C) 20xx ((nom de l'auteur))

Ce programme est libre, vous pouvez le redistribuer et/ou le modifier selon les termes de la Licence Publique Générale GNU publiée par la Free Software Foundation (version 2 ou bien toute autre version ultérieure choisie par vous).

Ce programme est distribué car potentiellement utile, mais SANS AUCUNE GARANTIE, ni explicite ni implicite, y compris les garanties de commercialisation ou d'adaptation dans un but spécifique. Reportez-vous à la Licence Publique Générale GNU pour plus de détails.

Vous devez avoir reçu une copie de la Licence Publique Générale GNU en même temps que ce programme ; si ce n'est pas le cas, écrivez à la Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, États-Unis.

Ajoutez également votre adresse électronique, le cas échéant, ainsi que votre adresse postale.

Si le programme est interactif, faites-lui afficher un court avertissement du type de celui-ci à chaque invocation : ...(nom du programme) version 69, Copyright (C) 20aa nom de l'auteur ...(nom du programme) est fourni sans AUCUNE GARANTIE. Pour plus de détails, tapez 'g'.

Ce programme est libre et vous êtes encouragé à le redistribuer sous certaines conditions ; tapez 'c' pour plus de détails.

Les commandes hypothétiques 'g' et 'c' doivent afficher les sections appropriées de la Licence Publique Générale GNU. Bien entendu, vous pouvez implanter ces commandes comme bon vous semble : options dans un menu, ou bien accessibles d'un clic de souris, etc., tout dépend de votre programme.

Si vous officiez en tant que programmeur, n'omettez pas de demander à votre employeur, votre établissement scolaire ou autres de signer une décharge stipulant leur renoncement aux droits qu'ils pourraient avoir sur le programme : ...((employeur, école...)) déclare par la présente ne pas revendiquer de droits sur le programme « (nom du programme) » réalisé par ...((nom de l'auteur)). ((signature du responsable)), ...((date)), ...((nom et qualité du responsable)).

La Licence Publique Générale ne permet pas d'inclure votre programme dans des logiciels sous licence commerciale spécifique. Si votre programme est une fonction de bibliothèque, vous jugerez probablement plus judicieux de le faire relever de la Licence Générale de Bibliothèque GNU (LGPL) plutôt que de la présente.

Index