Guide d’installation et de configuration de Linux

Table des matières

Liste des tableaux

Remarques de l’auteur

Il se peut que certaines informations fournies dans ce document soit spécifiques à mes distributions de Linux. À titre indicatif, j’utilise une distribution SuSE 7.0 et une distribution Slackware 7.0, toutes deux avec le noyau 2.4.0. de Linux. Je me suis efforcé de rendre ce document générique et indépendant de la distribution par comparaison, mais je ne ne peux pas le garantir. En particulier, il est connu que la structure des systèmes de fichiers de SuSE n’est pas tout à fait standard, et que la Slackware n’utilise pas la notion de niveaux d’exécution par défaut. De même, certaines informations pourront être spécifiques à la configuration matérielle des machines que j’utilise. Néanmoins, la plupart des informations fournies ici s’appliqueront à toutes les distributions de Linux. Elles permettront également aux personnes qui n’ont jamais vu Linux de débroussailler un peu le terrain.

Je remercie d’avance les gens qui pourront m’envoyer des remarques concernant des imprécisions, voire les horreurs et les âneries que j’ai pu écrire. Plus je recevrai de critiques constructives et de propositions, plus ce document a de chances de s’améliorer. Bien entendu, les modifications ne seront faites que si j’ai le temps de les faire…

Vous pourrez trouver la dernière version de ce document sur mon site Web (http ://casteyde.christian.free.fr).

Chapitre 1. Introduction

L'installation de Linux est une opération relativement compliquée, qui n'est pas à la portée de tout le monde. Même si la qualité des distributions actuellement vendues s'est grandement accrue ces derniers temps, au point que n'importe qui peut installer un système Linux viable sans trop de problème, la configuration du système pour obtenir un fonctionnement correct exige un travail assez important. En particulier, les distributions actuelles éprouvent encore quelques difficultés pour optimiser les périphériques exotiques, et souvent seules les fonctionnalités de base sont correctement configurées après une installation classique. Par ailleurs, la plupart des applications sont développées par des groupes de programmeurs indépendants, et bien que ce soit justement le rôle des distribution de réaliser l'intégration de tous ces composants dans un environnement homogène, celle-ci n'est pas forcément parfaite. Les outils de configuration des distributions vous permettront sans doute de configurer votre système de base simplement, mais pour aller au delà, il faudra sans doute intervenir manuellement.

Néanmoins, il faut reconnaître que celui qui installe Linux à partir d'une distribution sur un ordinateur assez vieux (c'est à dire un ordinateur qui ne dispose pas des derniers périphériques et cartes graphiques à la mode), ou dont les constituants sont de marque courante, obtient rapidement un système fonctionnel et capable de réaliser la plupart des opérations qu'il désire. En particulier, celui qui utilise son ordinateur pour travailler (j'entends par là écrire des lettres, les imprimer, naviguer sur Internet pour récupérer des informations, ou programmer) peut parfaitement se contenter de l'installation par défaut. Ce type de situation ne convient pas à tout le monde : la plupart des gens disposent de cartes graphiques récentes (surtout depuis l'avènement des jeux 3D) ou de périphériques spécifiques. Tout le monde ne se place pas uniquement dans le cadre d'une utilisation professionnelle, et il est absurde de disposer d'une carte son et de ne pas pouvoir l'utiliser. Et c'est là que le bât blesse ! Si l'on désire que Linux reconnaisse ces matériels exotiques, il va falloir mettre les mains dans le cambouis et avoir une bonne dose de patience. Ce problème de configuration apparaît malheureusement principalement pour les particuliers, qui souvent dispose de machines hétéroclites et absolument non standard. Dans le cadre d'une entreprise, il existe des personnes qualifiées pour résoudre ce type de problème, mais ce sont des informaticiens, et de plus les machines sont souvent homogènes, ce qui permet d'apporter des solutions génériques.

En conclusion, il faut être informaticien ou amateur très éclairé pour installer Linux sur une machine de particulier et pour le configurer de manière optimale. La situation est d'autant plus grave que la plupart des gens ne connaissent pas Linux, et qu'il est toujours difficile d'apprendre et de prendre de nouvelles habitudes. Je veux dire par là que même une tâche très simple à réaliser peut prendre un certain temps, car tout simplement on ne l'a jamais faite. Celui qui a installé trois fois MS Windows sait parfaitement le faire à présent, et il pense que c'est relativement facile. Et pourtant, il réalise souvent des tâches d'une complexité qui dépasse, là aussi, le commun des mortels.

Heureusement, et c'est là la force de Linux, ces opérations ne doivent être effectuées qu'une seule fois. On n'a absolument pas besoin de changer la configuration à chaque instant, comme c'est le cas sous MS Windows, parce que le système est globalement beaucoup plus stable. Il ne plante quasiment jamais, les applications ne peuvent pas le corrompre, et sa qualité supprime le besoin permanent de mettre à jour une partie du système. En clair, quand on en a un qui fonctionne, on le garde, non pas parce que c'est un enfer à installer et à configurer, mais tout simplement parce que ce n'est pas

nécessaire de le changer.

En résumé, on peut affirmer que :

- Linux est un système simple à installer sur des machines standards ;
- sa configuration sur une machine plus exotique requiert parfois une intervention manuelle ;
- dans la plupart des cas, cette intervention n'est pas très difficile à réaliser ;
- cependant, elle peut dérouter les personnes qui ne les ont jamais effectuées ;
- mais le jeu en vaut la chandelle, parce que le système est réellement solide.

L'objet de ce document est de donner les connaissances de base nécessaires à l'installation de Linux sur un ordinateur de particulier ou un petit serveur. Il est supposé que l'utilisateur a déjà utilisé un autre système d'exploitation, par exemple MS Windows. Cependant, aucune notion avancée d'informatique n'est nécessaire. Tout sera expliqué au fur et à mesure des besoins, et si nécessité est, des compléments d'informations seront données pour permettre la compréhension des opérations à effectuer. Néanmoins, les notions qui seront abordées ici ne seront pas simples, et il est possible que la plupart des personnes qui n'ont pas une grande habitude de l'informatique aient quelques difficultés à les assimiler. Ceci dit, à vaincre sans peine, on triomphe sans gloire, et l'installation de Linux vous procurera le plaisir d'apprendre.

Ce document est structuré en huit parties distinctes, qui correspondent essentiellement aux grandes étapes que vous suivrez pour installer Linux. La première partie a pour but de clarifier un peu les notions ayant trait aux logiciels libres. Elle tente d'expliquer pourquoi ces logiciels existent, et pourquoi ils font partie des meilleurs logiciels actuels. La deuxième partie décrit les concepts de base de la plupart des systèmes d'exploitation Unix. Elle ne traite pas de l'installation à proprement parler, mais sa lecture est recommandée pour tous ceux qui n'ont jamais vu un tel système. La troisième partie décrit l'installation du système de base de Linux. À l'issue de cette partie, vous devez disposer d'un système fonctionnel, utilisable mais non optimisé et ne permettant pas forcément d'utiliser tous vos périphériques. La quatrième partie constitue un petit court d'Unix pour les utilisateurs. Bien que, comme les deux premières parties, elle ne traite pas de l'installation à proprement parler, sa lecture est vivement recommandée. En effet, vous aurez à utiliser les commandes décrites dans ce chapitre pour réaliser la configuration de votre nouveau système. La cinquième partie décrit les opérations de configuration du système de base. Cette partie est en relation étroite avec la huitième partie, qui traite des fonctionnalités du noyau. Elle pourra être relue avec bénéfice lorsque vous aurez configuré une première fois votre noyau. La sixième partie traite de la configuration du réseau sous Linux. Le réseau est réellement l'un des aspects les plus importants de Linux, et nécessite donc une attention toute particulière. La septième partie donne les notions de base sur les mécanismes de compilation, qui seront utilisées dans la huitième partie. Elle décrit également la manière de faire pour compiler la dernière version de GCC, le compilateur C/C++ de GNU. La huitième partie décrit la compilation du noyau du système, opération indispensable pour obtenir un système qui « colle » à la machine et aux fonctionnalités désirées. Enfin, la neuvième et dernière partie vous décrit la procédure d'installation de XWindow, l'environnement graphique de Linux. L'installation des polices Truetype y est aussi présentée.

Chapitre 2. GNU, Linux et les logiciels libres

Vous entendrez souvent parler de la licence « GNU » et de la « Free Software Foundation » dans le monde de Linux. Pour bien comprendre ce qu'est la Free Software Foundation et ce que signifie la licence GNU, il est nécessaire de faire une brève présentation.

La Free Software Foundation est une organisation dont le but est de développer des logiciels libres. Le terme de « libre » signifie clairement que chacun peut en faire ce qu'il veut du logiciel, y compris le modifier. La vente n'est absolument pas interdite, et il faut donc faire la distinction entre libre et gratuit. Ceci dit, les logiciels libres sont de facto gratuits, car ils sont librement redistribuables par quiconque en possède une copie.

Afin de protéger les logiciels libres, la Free Software Foundation a rédigé la licence GNU (en fait, cette licence a originellement été rédigée pour le projet GNU, qui sera décrit plus loin). Cette licence stipule que le logiciel libre peut être redistribué, utilisé, modifié librement, pourvu que celui qui en bénéficie accorde les mêmes droits à ceux à qui il fournit les copies du logiciel, qu'il l'ait modifié ou non. Cette licence empêche donc l'aliénation du logiciel et sa transformation en logiciel propriétaire, de quelque manière que ce soit. Ceci implique que tout logiciel libre modifié par une autre personne que son auteur reste libre, et le restera à jamais. Ainsi, il est impossible qu'une société commerciale puisse un jour s'approprier un logiciel libre, même si elle l'améliore. Afin de garantir la liberté de modification des logiciels libres, la licence GNU impose aux distributeurs de fournir leur code source, c'est à dire tous les fichiers utilisés pour créer le logiciel. Vous trouverez de plus amples informations sur la notion de code source dans le Chapitre 8. Si vous désirez lire la licence GNU, vous pouvez en trouver une copie dans le fichier `/usr/src/linux/COPYING` une fois que vous aurez installé Linux.

Les programmes distribués sous la licence GNU ont souvent été écrits par des passionnés d'informatique, et ont été améliorés au fil du temps par les différentes personnes qui les ont utilisés. Il est même courant de voir des entreprises participer au développement des logiciels libres, pour diverses raisons. On peut se demander quel est l'intérêt pour une entreprise de développer un logiciel libre dont elle ne tirera aucun profit. En général, plusieurs critères peuvent entrer en ligne de compte. Ce peut être la publicité, le sponsoring, ou tout simplement parce que le logiciel est utilisé en interne et requiert une amélioration. D'autres sociétés fournissent un support et un service après-vente payants sur les logiciels libres qu'elles distribuent (c'est notamment le cas pour les distributions de Linux). Quoi qu'il en soit, cette participation collective a toujours pour but d'améliorer le logiciel. De plus, les gens qui développent un logiciel libre sont souvent des spécialistes de ce type de logiciel, ou des utilisateurs avertis. Il est donc logique qu'au bout d'un certain temps, les logiciels libres fassent partie des meilleurs logiciels dans leur catégorie. Un autre caractéristique de ce modèle de développement est l'accessibilité au code source des programmes et la liberté de le modifier. Cette facilité est à la source de la rapidité avec laquelle les bogues et les erreurs sont identifiées, et de la réactivité du monde des logiciels libres en général. On peut considérer qu'un logiciel utilisé par un grand nombre de personnes est virtuellement « sans bogues » connu, puisque si tel était le cas il serait immédiatement corrigé. Ce modèle de développement est donc certainement celui qui conduit à l'une des plus grande qualité logicielle qui soit.

Comme on l'a déjà dit, la licence GNU a été créée par la Free Software Foundation dans le cadre du projet GNU. Ce projet, toujours en cours de réalisation, a pour but d'écrire un système Unix libre et indépendant des Unix commerciaux. Précisons ici que le terme « Unix » caractérise un ensemble de systèmes d'exploitation, qui disposent tous à peu près des mêmes fonctionnalités et de la même manière d'y accéder. La Free Software Foundation a déjà écrit la plupart des utilitaires Unix dans le cadre du projet GNU, mais le c½ur du système (ce que l'on appelle le noyau) est toujours en cours de réalisation. Pour information, ce noyau se nomme « Hurd ».

Cependant, d'autres noyaux sont disponibles, avec lesquels les commandes GNU peuvent être utilisées. Parmi ces noyaux, il existe bien entendu Linux, qui a été écrit par le Finlandais Linus Torvalds lorsqu'il était étudiant, et amélioré par des programmeurs du monde entier sur Internet. Linux est un noyau parmi tant d'autres, à ceci près qu'il est, lui aussi, distribué sous la licence GNU, bien que n'ayant rien avoir avec la Free Software Foundation.

Ceci signifie qu'en fait il serait plus exact de parler du système « GNU/Linux » que de « Linux ». Sous cette dénomination, il est clair que ce système est constitué des outils GNU fonctionnant sous le noyau Linux. C'est donc un ensemble de logiciels libres provenant de plusieurs sources distinctes. Cependant, il est très courant d'entendre parler de « Linux » tout court, par abus de langage et par souci de simplicité. Bien entendu, cette dénomination est proscrite sur les sites Internet de la Free Software Foundation.

Précisons que la licence GNU n'est pas la seule licence permettant de distribuer des logiciels libres. Il existe d'autres licences, dont les termes sont à peu près similaires. Par exemple, la licence FreeBSD (un autre système Unix libre) exige également la distribution des sources, mais interdit l'exploitation commerciale des logiciels. Ceci explique l'absence de distributions de FreeBSD, et le fait qu'il est quasiment inconnu du grand public bien qu'il soit techniquement meilleur que Linux. Quelques outils fournis avec Linux sont distribués sous cette licence.

Enfin, pour information, le terme « GNU » est l'abréviation de l'anglais « GNU's Not Unix ». Cette curieuse phrase rappelle que le projet GNU est de réaliser un système Unix différent des autres. Vous remarquerez que cette définition est récursive, c'est à dire qu'elle utilise le mot « GNU » elle-même. Ceci doit être attribué au goût des développeurs de la Free Software Foundation pour ce genre de définitions infiniment récursives. Vous ne saurez sans doute jamais les raisons qui les ont poussés à choisir la lettre 'G' dans leur définition. Ceci étant, « GNU » ce prononce « gnou » en anglais, et vous trouverez donc souvent la représentation d'un gnou sur les sites Internet de GNU.

Chapitre 3. Concepts de base

Ce chapitre décrit les principes de base qu'il faut connaître pour bien comprendre Linux. Les informations qui sont données ici ne sont pas réellement spécifiques à Linux. Elles s'appliquent souvent aux systèmes Unix en général. Il est donc recommandé de le lire, surtout si l'on n'a jamais vu ni utilisé un système Unix. En particulier, les utilisateurs chevronnés de Windows risquent d'être légèrement déroutés. L'architecture du système sera présentée, ainsi que la sécurité et la notion d'utilisateur. Viendront ensuite les descriptions des principales fonctionnalités du système de fichiers et de sa structure.

3.1. Architecture du système

Comme tout logiciel d'une certaine taille, Linux est d'une grande complexité. Tous les systèmes d'exploitation récents sont constitués d'un grand ensemble de composants qui interagissent et dont la mise en place peut être soit indispensable au bon fonctionnement du système, soit facultative, soit tout simplement inutile étant donné la configuration matérielle et les besoins des utilisateurs. Cette complexité implique un grand nombre d'erreurs, d'anomalies et de dysfonctionnement possibles. En effet, pour qu'un système informatique fonctionne correctement, il faut tout prévoir pour donner une action appropriée à tous les événements possibles. Ceci n'est pas humainement réalisable quand le système devient trop complexe.

Pour résoudre ce problème, il est courant de subdiviser le système en composants indépendants, dont le mauvais fonctionnement potentiel ne peut perturber que partiellement les autres parties du système. Des points de synchronisation à partir desquels le système peut reprendre un fonctionnement normal après une erreur sont prévus. Ces points de synchronisation permettent simplement d'assurer la viabilité du système, même en cas d'erreur imprévue. Pour quelques systèmes, le seul point de synchronisation est le point de départ, à savoir le démarrage de l'ordinateur. De tels systèmes doivent donc souvent être redémarrés, parfois pour une cause mineure (erreur d'un logiciel, modification de la configuration du système, ajout d'un composant au système). Ce n'est pas le cas de Linux, qui dans le pire des cas détruit le composant qui a généré l'erreur sans toucher aux autres parties du système. Le point de synchronisation de Linux est donc le redémarrage d'une partie du système uniquement, ce qui assure ainsi une grande stabilité du système complet.

Il va de soi que lorsqu'un composant se plante, ceux qui l'utilisent risquent fort de se retrouver dans un état d'erreur assez difficile à gérer. Ceci peut souvent provoquer leur propre perte. Par conséquent, plus un composant est utilisé, plus il doit être fiable. Or il est un composant à la base de tout dans Linux : le noyau (« kernel » en anglais). C'est le c½ur du système, et en fait c'est précisément le système Linux. Heureusement, ce composant est d'une très, très grande fiabilité, et il n'est pas rare de voir un système Linux fonctionner plusieurs mois ou années sur des serveurs. Cette fiabilité provient du modèle de développement de Linux, qui est ouvert à tout le monde (chacun peut récupérer, lire, modifier, compléter ou corriger le noyau à condition de savoir bien programmer). À partir d'une taille critique en terme de nombre d'utilisateurs, taille que Linux a atteinte, il existe suffisamment de développeurs pour détecter et corriger les erreurs. Ainsi, dès qu'une erreur est détectée, elle est souvent corrigée dans les jours qui suivent, ce qui assure une grande qualité.

Le noyau gère quasiment tout (mémoire, disques, systèmes de fichiers, réseau, clavier, droits des

utilisateurs, etc…), mais il n'est pas exploitable tel quel. Il n'est par exemple pas capable d'offrir une interface utilisateur permettant de lui donner les commandes qu'il doit exécuter. Ces opérations sont du ressort d'autres modules développés par la Free Software Foundation. Parmi ces modules, on trouve le « shell » (ce qui signifie grosso modo « environnement utilisateur »). Le shell est capable de lire des commandes saisies au clavier, de les exécuter, et d'afficher leur résultat à l'écran. En général, les programmes capables de réaliser ces opérations sont appelés des interpréteurs de commandes. Mais le shell est bien plus que cela, car il peut être programmé, et il peut gérer les processus (en arrêter un, en lancer un autre, etc…). En fait, les commandes que le shell peut exécuter sont en nombre très réduit. La plupart des commandes sont tout simplement d'autres programmes.

Les programmes que l'on peut utiliser dans le shell sont des programmes dits « en ligne de commande », parce qu'ils sont propres à être utilisés dans les lignes de commandes que l'on saisit au clavier dans le shell. Ces programmes sont, encore une fois, développés soit par la Free Software Foundation, soit par des bénévoles, toujours sous la licence GNU. Toutes ces commandes sont des commandes compatibles Unix. Ces commandes sont absolument essentielles pour pouvoir utiliser le système, mais elles sont assez rébarbatives et peu d'utilisateurs acceptent de s'en contenter.

C'est pour cela qu'une couche graphique a été développée, pour introduire une interface graphique plus conviviale (mais cependant légèrement moins puissante en termes de fonctionnalités) : XWindow. Encore une fois, cette couche logicielle est constituée de plusieurs composants, dont la base est le serveur X. Le serveur X est un programme capable de fournir les services graphiques (d'où le nom de serveur) aux autres applications. Plusieurs implémentations concurrentes existent. L'une d'elles est particulièrement utilisée sous Linux, puisqu'elle est libre (comme tout le reste) : XFree86. À vrai dire, un serveur X ne fait pas grand chose d'autre que de réaliser des affichages sous les ordres d'autres programmes. D'autres composants permettent donc d'obtenir des fonctionnalités de plus haut niveau.

Le gestionnaire de fenêtre (« Window Manager » en anglais) est le composant qui se place juste au dessus du serveur X. Il est en charge, comme son nom l'indique, de gérer les fenêtres des applications graphiques sous X. C'est le gestionnaire de fenêtres qui prend en charge la gestion des décorations des fenêtres de premier niveau (c'est à dire des fenêtres principales des programmes). Par exemple, il s'occupe d'afficher les bords, la barre de titre, les boutons d'iconisation et de restauration, etc… des fenêtres. C'est également lui qui s'occupe du positionnement des fenêtres, et qui donc permet à l'utilisateur de déplacer et de réduire les fenêtres des applications graphiques. L'utilisateur est libre d'utiliser le gestionnaire de fenêtre qu'il désire, selon ses propres goûts et ses désirs, la différence est souvent une pure question de style.

Il existe des environnements graphiques complets, qui, en plus d'un gestionnaire de fenêtre souvent extrêmement puissant, fournissent la plupart des outils classiques que l'on est en droit d'attendre d'un système graphique moderne. Ainsi, ces environnements comprennent des éditeurs, des outils de configuration, des navigateurs Internet, des logiciels multimédia, etc… En plus de ces applications, ils fournissent un cadre standard pour les applications graphiques qui savent communiquer avec eux. Ce cadre permet d'améliorer l'intégration des diverses applications entre elles, et c'est la raison pour laquelle on appelle souvent ces environnements des gestionnaires de bureau. KDE et Gnome sont des exemples de tels environnements de travail.

Enfin, au dessus de toutes ces couches logicielles, on trouve les applications X, qui sont aussi diverses que variées (traitement de texte, tableurs, logiciels de dessin, etc…). Quelques-unes de ces

applications sont simplement des « front-end » d'applications en lignes de commande, c'est à dire des interfaces graphiques à des programmes non graphiques existants. Ce concept permet d'avoir un composant unique, et plusieurs interfaces différentes pour ce composant, et en plus de rendre indépendant la fonctionnalité de l'interface utilisateur. Encore une fois, la stabilité en est d'autant plus accrue.

Bon nombre d'applications pour XWindow sont libres, ou utilisables librement à des fins non commerciales. Ceci signifie qu'un particulier a le droit de les utiliser tant qu'il ne s'en sert pas pour un travail qu'il revendra. Comme c'est le cas de la plupart des particuliers, on peut considérer qu'il est actuellement possible, avec Linux, d'avoir un environnement logiciel complet, fiable et performant… pour un prix de revient minime.

En résumé, un système GNU/Linux est structuré de la manière suivante :

- le noyau Linux ;
- le shell ;
- les programmes en ligne de commande, dont le serveur XWindow ;
- le gestionnaire de fenêtre ;
- le gestionnaire de bureau ;
- les applications XWindow.

Il n'est pas évident d'établir un parallèle avec MS Windows, puisque ce système est réellement monolithique. Cependant, on peut considérer que le noyau Linux correspond aux modules KERNEL ou IO.SYS de Windows, que le shell correspond aux interpréteurs de commandes COMMAND.COM ou CMD.EXE, que les programmes en ligne de commande aux programmes DOS ou console classiques (xcopy, fdisk, format, etc…), que le serveur X correspond au couple pilote de carte graphique - GDI, que le gestionnaire de fenêtre correspond au module USER, et le gestionnaire de bureau à l'explorateur, les fonctionnalités d'OLE et aux programmes fournis avec Windows. La différence essentielle vient du fait que le shell est à peine programmable sous Windows, que les commandes DOS ont tendance à accéder aux ressources de la machine directement, sans passer par le noyau, que le driver de carte graphique, la GDI et le module USER sont tous intégrés dans le système au lieu d'en être séparés (ce qui multiplie les chances de crash du système complet), et que la plupart des applications Windows ne peuvent fonctionner que dans l'environnement graphique. Elles sont donc entraînées par le système lorsque les modules graphiques de Windows plantent (je n'ai d'ailleurs jamais vu un processus DOS survivre à un crash de l'interface graphique de Windows).

En conclusion :

- les systèmes Unix, donc Linux, sont très structurés, plus simples à utiliser, à configurer, à maintenir et à développer ;
- ils sont très stables, car les composants de haut niveau n'interfèrent pas sur les composants de bas niveau ;
- ils sont faciles à personnaliser, puisque l'utilisateur a le choix des outils à chaque niveau ;

- Linux a de plus la particularité d'être parfaitement modifiable, puisque si l'on sait programmer, on peut personnaliser les composants du système ou en rajouter ;
- et il n'est pas propriétaire, c'est à dire que l'on n'est pas dépendant d'une société éditrice de logiciel pour résoudre un problème donné.

En bref, c'est la voie de la vérité.

3.2. Sécurité et utilisateurs

Linux est un système multi-utilisateurs. Ceci signifie que plusieurs personnes peuvent utiliser l'ordinateur simultanément (et pas uniquement les unes à la suite des autres), et que le système se charge de faire respecter la loi entre elles. Les ressources de la machine sont ainsi partagées équitablement, tant au niveau de la puissance de calcul qu'au niveau de la mémoire, du disque, des imprimantes, etc…

Évidemment, une question se pose : comment plusieurs utilisateurs peuvent-ils se partager le clavier et l'écran ? La réponse est simple : ils ne le peuvent pas. Par conséquent, il n'y a que trois solutions possibles : soit on connecte à l'ordinateur d'autres claviers et d'autres écrans (on appelle un couple clavier/écran un « terminal »), soit on accède au système par l'intermédiaire d'un autre ordinateur via le réseau, soit les utilisateurs lancent tour à tour leurs programmes. La dernière solution nécessite que les programmes ne soient pas interactifs : ils doivent être capable de fonctionner sans intervention de celui qui les a lancé.

La première hypothèse n'est pas sérieuse pour un particulier, il est d'ailleurs assez difficile de connecter un terminal supplémentaire sur un PC (c'est réalisable, par l'intermédiaire des ports série, mais ce n'est pas recommandé). La deuxième solution en revanche, est nettement plus envisageable. Il est parfaitement possible qu'un particulier dispose de deux PC connectés en réseau, et que deux membres de la même famille cherche à utiliser des ressources de la même machine (par exemple, une imprimante ou tout autre périphérique, un programme installé sur un seul ordinateur, des fichiers personnels, etc…). Quant à la troisième solution, elle est du domaine du quotidien, même pour un ordinateur sans réseau et avec un seul utilisateur. En effet, certains programmes sont lancés par le système pour effectuer des tâches de maintenance, et fonctionnent au nom de l'administrateur de la machine.

Il va de soi que pour être multi-utilisateurs, le système doit satisfaire certains critères :

- il doit être multitâche, c'est à dire qu'il doit être capable de faire fonctionner plusieurs programmes simultanément sur la même machine, en partageant les ressources de celle-ci ;
- il doit être fiable, car un arrêt du système peut déranger un nombre arbitraire de personnes, y compris celles qui ne sont pas à proximité de l'ordinateur ;
- il doit être sûr, car il ne faut pas que les erreurs ou les malveillances d'un utilisateur ne puisse déranger les autres.

Le multitâche est assuré au niveau du noyau. Chaque programme en cours d'exécution (on les appelle « processus ») fonctionne dans sa propre zone de mémoire, qui est complètement contrôlée par le noyau. Les ressources du processeur sont partagées entre les processus, et il est impossible à l'un

d'entre eux de monopoliser la mémoire, le disque ou quoi que ce soit. Les processus doivent toujours passer par le noyau pour effectuer une opération, ce qui permet un contrôle absolu.

La fiabilité est également assurée au niveau du noyau. Les zones de mémoire utilisées par chaque processus (encore appelées « espaces d'adressages ») sont bien distinctes et bien identifiées par le noyau. Ceci implique qu'il est impossible à un processus de perturber le fonctionnement d'un autre processus. Ainsi, si un processus fait une faute, il est purement et simplement terminé par le noyau. Ceci est sans appel : le noyau est le seul maître à bord.

Enfin, la sécurité est assurée par le noyau et par le système de fichiers. Au niveau du noyau, chaque utilisateur est identifié de manière unique par un numéro dans le système. Ce numéro est utilisé pour vérifier les droits de l'utilisateur, ou, autrement dit, ce qu'il peut faire. Les droits des utilisateurs comprennent la possibilité de lire ou écrire un fichier, d'accéder ou non à une ressource ou d'exécuter un programme. Il est également possible de créer un ou plusieurs « groupes » d'utilisateurs, et de donner des droits particuliers à ces groupes. Tous les utilisateurs qui font partie de ce groupe recevront les droits du groupe. Ainsi, des classes d'utilisateurs peuvent être facilement définies, et ces classes peuvent se voir attribuées un peu plus de privilèges que les utilisateurs normaux selon les nécessités. Il existe toutefois un utilisateur spécial, qui a tous les droits : l'administrateur du système (« root » en anglais). Aucune restriction ne s'applique à cet utilisateur, car il doit être capable de gérer le système, l'installer, l'arrêter, le mettre à jour si nécessaire, et de définir les utilisateurs et leurs droits.

Au niveau du système de fichiers, la sécurité est assurée par le stockage d'informations additionnelles pour chaque fichier ou répertoire. Ces informations permettent de connaître :

- le numéro de l'utilisateur qui possède le fichier ou le répertoire. En général, le propriétaire est simplement celui qui l'a créé. Cependant, l'administrateur peut changer le propriétaire d'un fichier à tout moment ;
- le numéro du groupe auquel appartient le fichier ou le répertoire. Tout fichier ou répertoire appartient à un groupe unique, qui est utilisé en pour calculer les droits des utilisateurs faisant partie de ce groupe. Par défaut, un fichier nouvellement créé par un utilisateur appartient au groupe des utilisateurs « users », ce qui convient dans la majorité des cas (attention, ce comportement varie selon les systèmes). L'utilisateur peut toutefois donner ses fichiers à n'importe quel groupe dont il fait partie ;
- les droits d'accès au fichier ou au répertoire pour le propriétaire, pour les utilisateurs faisant partie du groupe auquel appartient le fichier ou le répertoire, et pour tous les utilisateurs en général. Ces droits comprennent le droit de lecture (représenté par la lettre 'r', pour « Read only »), le droit d'écriture (représenté par la lettre 'w', pour « Writeable »), le droit d'exécution (représenté par la lettre 'x', pour « eXecutable ») et quelques attributs supplémentaires (qui seront détaillés plus loin).

Il n'est sans doute pas inutile de préciser un peu le fonctionnement des droits dans le système de fichiers. Le droit de lecture correspond à la possibilité d'ouvrir et de consulter un fichier, ou de lister le contenu d'un répertoire. Le droit d'écriture correspond à la possibilité de modifier un fichier, ou de créer ou supprimer un fichier d'un répertoire. Enfin, le droit d'exécution correspond à la possibilité d'exécuter un fichier contenant un programme, ou d'entrer dans un répertoire. On notera par exemple qu'il est possible d'obtenir la liste des fichiers d'un répertoire sans pouvoir s'y déplacer, ou encore

de modifier un fichier sans pouvoir le lire. On prendra garde également que le fait d'avoir le droit d'écriture sur un fichier ne donne pas le droit de le supprimer (cependant, on peut le vider !). Pour cela, il faut avoir le droit d'écriture sur le répertoire contenant ce fichier. Comme on le voit, les droits d'accès aux fichiers et aux répertoires sont très souples.

Ces droits sont attribués séparément pour le propriétaire, le groupe et les autres utilisateurs (c'est à dire les utilisateurs qui ne font pas partie du groupe auquel appartient le fichier). Il est donc possible de donner par exemple tous les droits au propriétaire d'un fichier, et seulement le droit de lecture aux autres utilisateurs. Cette configuration est celle qui est choisie par défaut lors de la création d'un fichier, elle assure que seul le propriétaire peut modifier ou exécuter ce fichier, tout en laissant la possibilité aux autres de le lire. Ce choix privilégie la sécurité des données de chacun en laissant le maximum de liberté aux autres. Si plusieurs personnes doivent travailler sur les mêmes fichiers, il suffit de les regrouper dans un groupe, de donner les fichiers sur lesquels ils doivent travailler à ce groupe, et de donner les droits d'écriture aux membres de ce groupe sur ces fichiers.

La sécurité du système est transitive, ceci signifie que tout programme lancé par un utilisateur s'exécute en son nom et reçoit donc les droits de cet utilisateur. Le processus correspondant se voit donc attribué les mêmes restrictions que l'utilisateur qui l'a lancé. Il dispose également des droits du groupe auquel le fichier du programme appartient. Il existe toutefois quelques exceptions à cette règle, pour les programmes dont le comportement est bien connu et qu'il est impossible de détourner de leur fonction initiale. C'est notamment le cas de quelques commandes systèmes (comme **passwd**, qui permet de changer de mot de passe), qui peuvent être lancées par les utilisateurs et qui s'exécutent toutefois au nom du système (dans le compte root). Il est donc impossible à un utilisateur de violer les règles de sécurité du système. Pour parvenir à ce comportement, il faut utiliser des attributs spéciaux sur les fichiers de ces programmes. Les attributs spéciaux sont décrits ci-dessous.

Le premier attribut spécial est le bit « setuid » (qui est l'abréviation de l'anglais « SET User IDentifier »). Il ne peut être placé qu'au niveau des droits du propriétaires sur le fichier. Il permet d'indiquer que le fichier est exécutable, et que lorsque le programme qu'il contient est lancé par un utilisateur, le processus correspondant s'exécute avec les droits du propriétaire du fichier et non pas avec ceux de l'utilisateur qui l'a lancé. Cependant, le système conserve tout de même le numéro de l'utilisateur réel qui a lancé le processus, ce qui fait que le programme peut savoir par qui il a été lancé et au nom de qui il s'exécute. Un processus dispose donc toujours de deux numéros d'utilisateur :

- le numéro de l'utilisateur réel (« real user id » en anglais), qui est le numéro de l'utilisateur qui a lancé le programme ;
- le numéro de l'utilisateur effectif (« effective user id » en anglais), qui est le numéro de l'utilisateur avec les droits duquel le processus fonctionne.

Le bit setuid permet donc simplement d'affecter le numéro du propriétaire du fichier au numéro d'utilisateur effectif du processus lorsqu'il est lancé. Le fait de conserver le numéro de l'utilisateur réel permet au programme de réaliser des vérifications de sécurité additionnelles. Par exemple, la commande **passwd**, qui permet de changer le mot de passe d'un utilisateur, a besoin des droits de l'utilisateur root pour enregistrer le nouveau mot de passe. Il dispose donc du bit setuid pour que tous les utilisateurs puissent l'utiliser. Cependant, même s'il s'exécute au nom de l'utilisateur root, il ne doit pas permettre à n'importe qui de changer le mot de passe des autres utilisateurs : seul l'utilisateur

root a le droit de faire cette opération. Il utilise donc le numéro de l'utilisateur réel qui a lancé la commande pour savoir si c'est bien l'utilisateur root qui l'a lancé. Le bit setuid est l'attribut le plus couramment utilisé, essentiellement pour certaines commandes systèmes. Il est représenté par la lettre 's' (comme « Setuid »), et il remplace le droit d'exécution ('x') des fichiers pour le propriétaire des fichiers (rappelons que le bit setuid implique que le fichier est exécutable). Il n'a aucune signification pour les répertoires.

Le deuxième attribut spécial est le bit « setgid » (qui est l'abréviation de l'anglais « SET Group IDentifier »). Ce bit fonctionne un peu de la même manière que le bit setuid, à ceci près qu'il fixe le numéro de groupe effectif du processus lancé à celui de son fichier exécutable. Cet attribut est également représenté par la lettre 's', et remplace le droit d'exécution ('x') pour les utilisateurs du groupe auquel appartient le fichier exécutable. Contrairement au bit setuid cependant, il a une signification pour les répertoires. Un répertoire disposant du bit setgid permet de faire en sorte que tous les fichiers qui sont créés dans ce répertoire se voient automatiquement attribués le même groupe que le répertoire. Ce bit est relativement peu utilisé.

Enfin, le troisième et dernier attribut spécial est le bit « sticky ». Cet attribut permet remplace l'attribut exécutable pour les autres utilisateurs que le propriétaire du fichier ou du répertoire et les membres du groupe auquel il appartient. Contrairement aux bits setuid et setgid, il est représenté par la lettre 't' (pour « sTickky »). Sa signification est assez spéciale : elle permet de faire en sorte que les programmes restent chargés en mémoire après leur terminaison, ce qui permet de les relancer plus rapidement. Afin de ne pas consommer la mémoire de manière permanente, le code du programme est placé automatiquement dans le swap s'il n'est toujours pas relancé après un certain temps, mais même dans ce cas, tous les calculs de chargement sont déjà effectués. Le lancement des programmes marqués de ce bit sera donc toujours accéléré. Sachez cependant ne pas abuser du bit sticky car la mémoire (même virtuelle) est encore une ressource rare. Pour les répertoires, sa signification est totalement différente : elle permet de restreindre les droits des utilisateurs sur les répertoires ayant ce bit positionné. Ce bit fait en sorte que même si un utilisateur dispose des droits d'écriture sur le répertoire, il ne peut pas supprimer tous les fichiers de ce répertoire. Les seuls fichiers qu'il est autorisé à supprimer sont ses propres fichiers. Bien entendu, il est toujours possible d'ajouter des fichiers dans le répertoire en question.

En pratique, les utilisateurs n'ont pas à se soucier des attributs des fichiers, et en fait même l'administrateur laisse souvent les attributs par défaut, car ils correspondent à la majorité des besoins de sécurité. L'administrateur a cependant le contrôle total sur les droits d'accès aux fichiers, sur le propriétaire et le groupe de chaque fichier. Il est également le seul utilisateur capable de créer un autre utilisateur ou un groupe, ainsi que de les détruire.

Les questions qui se posent évidemment sont les suivantes. Est-ce qu'un particulier a besoin de tout cela ? Ces fonctionnalités ne sont-elles pas des fonctionnalités réservées aux serveurs ? Est-ce qu'on ne risque pas de perdre beaucoup de temps pour définir les droits pour chaque utilisateur et pour chaque ressource du système ? La gestion de la sécurité ne consomme-t-elle pas trop de ressources ? Ces questions sont légitimes, mais en fait, il est très intéressant même pour un particulier de disposer de ces fonctionnalités. En effet, la sécurité permet tout simplement de protéger le système contre ses propres erreurs. Qui n'a pas effacé un jour un fichier important ou déplacé le répertoire Windows en bougeant légèrement la souris lors d'un double clic effectué trop lentement ? Avec Linux, on peut faire n'importe quoi, on est certain que le système restera intact. D'ailleurs la règle essentielle, même pour

un ordinateur utilisé par une seule personne, est de toujours créer un compte utilisateur normal et de ne jamais travailler sous le compte root. Cette sécurité est telle que Linux est le système d'exploitation idéal pour apprendre l'informatique à quelqu'un : savoir que le système protège tout ce qui est important permet aux débutants de prendre des initiatives sans craintes. Quant aux éventuels revers de médaille, ils sont absents : la gestion de la sécurité ne consomme quasiment aucune ressource, et sa configuration est élémentaire. Toutes les distributions s'installent de telle sorte que le système se protège des utilisateurs, et que ceux-ci soient indépendants les uns des autres.

En résumé :

- le multitâche est un confort indéniable. Il est appréciable de pouvoir utiliser son ordinateur même lorsqu'il est en train de faire une tâche longue en arrière plan ;
- la fiabilité est évidemment indispensable. Il est rassurant de se dire que même si un processus très gourmand en ressources fonctionne en arrière plan, il ne perturbera pas les autres processus ;
- la sécurité permet de se protéger de ses propres erreurs, pour un coût minimal. Il suffit de conserver les paramètres par défaut du système et de ne plus s'en soucier !

3.3. Fonctionnalités du système de fichiers

Les systèmes de fichiers des systèmes Unix sont très puissants. Ils assurent à la fois fonctionnalité, performances, et fiabilité. Il est sûrement bon de revenir sur les fonctionnalités fournies par les systèmes de fichiers Unix. En effet, peu d'utilisateurs savent exactement quels sont les services qu'ils peuvent fournir en général, et beaucoup croient que les systèmes de fichiers savent juste stocker des données dans des fichiers organisés dans une hiérarchie de répertoire. Heureusement, ils permettent de faire beaucoup mieux que cela !

Pour commencer, il faut préciser que Linux ne travaille pas directement avec les systèmes de fichiers physiques. En effet, il interpose systématiquement un système de fichiers intermédiaire, nommé « Virtual File System » (« VFS » en abrégé), qui permet aux applications d'accéder à différents systèmes de fichiers de manière indépendante de leur nature et de leur structure interne. Le système de fichiers virtuel ne fait pas grand chose en soi : il se contente de transférer les requêtes des applications vers les systèmes de fichiers réels. Il fournit donc une interface bien définie pour les applications, que celles-ci doivent utiliser. Les systèmes de fichiers réels, quant à eux, doivent simplement fournir les services dont le système de fichiers virtuel a besoin. Tous les systèmes de fichiers réels ne disposent pas de toutes les fonctionnalités demandées par le système de fichiers virtuel, dans ce cas, la requête de l'application désirant effectuer cette opération échouera tout simplement.

Comme on peut le constater, cette architecture est modulaire. Et comme on l'a vu pour l'architecture du système en général, ceci apporte beaucoup de bénéfices. Les plus évidents sont indiqués ci-dessous :

- Linux est capable de gérer plusieurs systèmes de fichiers réels. La seule condition est qu'ils doivent tous fournir les services de base exigés par le système de fichier virtuel ;

- les applications peuvent utiliser plusieurs de ces systèmes de fichiers réels de manière uniforme, puisqu'elles n'utilisent que le système de fichiers virtuel. Cela simplifie leur programmation, et permet d'éviter autant de bogues potentielles ;
- chaque système de fichiers réel étant indépendant des autres, il ne perturbe pas leur fonctionnement. En particulier, un système de fichiers corrompu ne corrompt pas les autres.

Avec cette architecture, un grand nombre de systèmes de fichiers ont été développé pour Linux. Parmi ces systèmes de fichiers, on retrouve les plus connus, à savoir :

- le système de fichiers EXT2, qui est le système de fichiers natif de Linux ;
- les systèmes de fichiers FAT, FAT32 et FAT32X (utilisés par les systèmes DOS et Windows) ;
- le système de fichiers NTFS en lecture seule pour l'instant (utilisé par Windows NT) ;
- le système de fichiers ISO9660, qui est utilisé par tous les CD-ROM. Les extensions permettant de gérer les noms longs sont également gérées. Ces extensions comprennent en particulier le système de fichiers Juliette (extension de Microsoft pour Windows 95) et Rock Ridge (extension de tous les systèmes Unix) ;
- le système de fichiers NFS (utilisé pour distribuer sur un réseau un système de fichier) ;
- le système de fichiers ReiserFS, qui supprime la notion de block disques et qui est journalisé (c'est à dire qu'il supporte les arrêts intempestifs du système).

Linux gère également d'autres systèmes de fichiers, utilisés par d'autres systèmes d'exploitation (Unix ou non). Il permet même d'intégrer un pseudo système de fichiers généré par le noyau. Ce système de fichiers est complètement fictif : sa structure et ses fichiers sont générés dynamiquement par le noyau lorsqu'une application y accède. Il est principalement utilisé par les applications pour lire des informations que le noyau met à leur disposition, ainsi que pour changer dynamiquement certains paramètres du noyau en les écrivant simplement dans les fichiers.

Le système de fichiers natif de Linux EXT2 est de loin le système de fichier le plus fonctionnel et le plus performant, car il a été hautement optimisé. Les principales fonctionnalités de EXT2 sont les suivantes :

- les accès aux fichiers sont rapides, même plus rapides que les systèmes de fichiers basés sur la FAT sous Windows, qui pourtant ne gèrent pas les droits des utilisateurs ni les autres fonctionnalités avancées des systèmes de fichiers Unix ;
- la fragmentation des fichiers est quasiment inexistante. En fait, la fragmentation des fichiers est si faible que l'on peut l'ignorer en pratique. Ceci provient de l'algorithme que EXT2 utilise pour allouer les blocs du disque dur lors de l'écriture dans un fichier : il recherche tout simplement à donner systématiquement les blocs les plus proches. Pour donner un ordre de grandeur de la fragmentation des fichiers sur une partition de 800 Mo, après installation, suppression, manipulation d'un grand nombre d'applications et de petits fichiers, le tout réalisé par plusieurs processus fonctionnant en même temps, elle reste inférieure à 1% sur 57571 fichiers (sur un total de 249856 fichiers que le système de fichiers pourrait contenir) ;

- quant à la fiabilité, elle est gérée grâce à un stockage redondant des principales structures de données internes. Ainsi, si une erreur apparaît dans le système de fichiers, les parties défectueuses peuvent être reconstituées à partir des informations sauvegardées. Cette réparation est réalisée automatiquement à chaque redémarrage de la machine si nécessaire.

Nous allons maintenant voir les quelques fonctionnalités additionnelles que EXT2 supporte. Notez bien ici que d'autres systèmes de fichiers Unix peuvent fournir ces fonctionnalités, mais qu'il ne sera question ici que de EXT2 par souci de simplification.

La première fonctionnalité intéressante est le support des droits d'accès aux fichiers pour les utilisateurs et pour les groupes. La gestion de ces droits est un impératif sous Unix. Comme on l'a déjà vu, ces droits comprennent les droits d'accès en lecture, en écriture et en exécution, plus les attributs spéciaux des fichiers. Ces droits sont fixés indépendamment pour l'utilisateur propriétaire du fichier, le groupe d'utilisateur propriétaire du fichier, et tous les autres utilisateurs.

Une autre fonctionnalité intéressante est la possibilité de réaliser des liens sur des fichiers ou des répertoires. Un lien est une référence à un fichier ou un répertoire existant, qui peut être manipulé exactement comme sa cible. Il existe deux sortes de liens : les liens physiques, qui sont réellement une référence sur les données du fichier, au niveau de la structure même du système de fichiers, et les liens symboliques, qui ne sont rien d'autre qu'un fichier additionnel contenant les informations nécessaires pour retrouver la cible.

Les liens physiques présentent les inconvénients de ne pas pouvoir référencer des répertoires, et de ne pouvoir référencer que des objets du même système de fichiers que celui dans lequel ils sont créés. La limitation sur les répertoires permet d'éviter de construire des cycles dans la structure du système de fichiers. Quant à la limitation à la frontière des systèmes de fichiers, elle est obligatoire puisque les liens physiques sont gérés directement au niveau de la structure du système de fichiers. En revanche, ils présentent des avantages certains :

- le déplacement des cibles ne les perturbe pas si celles-ci restent dans le même système de fichiers, parce que dans ce cas les données ne sont pas déplacés sur le disque ;
- la suppression de la cible ne détruit pas le lien physique. Tous les liens physiques sur un fichier ou un répertoire partagent la même structure de données du système de fichiers, et celle-ci n'est réellement détruite qu'à la destruction du dernier lien physique.

En fait, toute entrée de répertoire est un lien physique sur le contenu du fichier. Le fait d'avoir plusieurs liens physiques sur les mêmes données correspond à disposer de plusieurs entrées de répertoires donnant accès aux mêmes données dans le système de fichiers. Il serait possible de créer des liens physiques dans un système de fichiers FAT, mais ils seraient interprétés comme des références croisées par les outils de vérification de disque. Le système de fichiers FAT de Linux interdit donc la création des liens physiques, tout comme le font DOS et Windows.

Les liens symboliques, quant à eux, permettent de référencer des fichiers ou des répertoires se trouvant dans d'autres systèmes de fichiers que le leur. C'est pour cette raison qu'ils sont très couramment utilisés (en fait, les liens physiques ne sont quasiment pas utilisés, parce qu'il est très courant de faire un lien sur un répertoire, ce que seuls les liens symboliques savent faire). En revanche, ils sont

extrêmement dépendants de leur cible : si elle est supprimée ou déplacée, tous les liens symboliques qui s'y réfèrent deviennent invalides.

La référence sur le fichier ou le répertoire cible contenue dans les liens symboliques peut être soit relative à l'emplacement de la cible, soit absolue dans le système de fichiers. Chacune de ces méthodes a ses avantages et ses inconvénients : les liens symboliques qui contiennent des références relatives ne sont pas brisés lors d'un déplacement de la cible, pourvu qu'ils soient déplacés également et restent à la même position relative par rapport à celle-ci dans la hiérarchie du système de fichiers. En revanche, ils sont brisés s'ils sont déplacés et que la cible ne l'est pas. Les liens symboliques utilisant des références absolues sont systématiquement brisés lorsque la cible est déplacée, mais ils restent valides lorsqu'ils sont eux-mêmes déplacés. Comme en général c'est le comportement que l'on recherche, les liens symboliques sont toujours créés avec des références absolues, mais vous êtes libres de faire autrement si vous en ressentez le besoin. Sachez cependant que déplacer une source de données n'est jamais une bonne idée. Le tableau suivant récapitule les avantages et les inconvénients des différents types de liens :

Tableau 3-1. Caractéristiques des liens physiques et symboliques

Fonctionnalité	**Liens physiques**	**Liens symboliques**	
		Référence relative	**Référence absolue**
Peuvent être déplacés	**Oui**	**Avec la cible**	**Oui**
Suivent la cible	**Oui**	**Si déplacés avec elle**	**Non**
Gèrent la suppression de la cible	**Oui**	**Non**	**Non**
Peuvent référencer des cibles sur un autre système de fichiers	**Non**	**Oui**	**Oui**
Peuvent référencer des répertoires	**Non**	**Oui**	**Oui**

Le système de fichiers EXT2 donne également la possibilité de gérer des fichiers presque vides (« sparse files » en anglais) et les quotas. Ces deux fonctionnalités sont relativement peu utilisées par les particuliers, elles ne sont mentionnées ici qu'afin d'être complet. Les fichiers presque vides sont des fichiers contenant des données séparées par de grands espaces vides. Il est inutile de stocker ces espaces vides sur le disque, aussi EXT2 signale-t-il simplement que ce fichier contient des trous, et il ne stocke que les données réelles et la position des trous avec leur taille. Ceci constitue une économie de place non négligeable. Les applications classiques des fichiers presque vides sont les bases de données, qui utilisent souvent des fichiers structurés contenant relativement peu de données effectives. Les quotas quant à eux permettent d'attribuer un espace disque fixe à chaque utilisateur. Ce n'est réellement utile que pour les serveurs.

3.4. Structure du système de fichiers

Il est nécessaire de définir un peu les termes qui vont être utilisés dans cette section, car les systèmes

de fichiers Unix sont très différents des systèmes de fichiers du DOS et de Windows. La connaissance de ce vocabulaire est nécessaire pour la compréhension de la structure du système de fichiers de Linux.

Comme la plupart des systèmes de fichiers, les systèmes de fichiers Unix sont structurés hiérarchiquement, et regroupent les fichiers dans des répertoires. Il existe un répertoire racine, qui contient tous les fichiers, soit directement, soit indirectement dans ses sous-répertoires. C'est de ce répertoire que débute tous les chemins possibles dans le système de fichiers. Chaque fichier ou répertoire a un nom qui permet aux utilisateurs du système de l'identifier. Le seul répertoire qui n'a pas de nom est le répertoire racine. Les systèmes de fichiers Unix n'ont pas les mêmes limitations sur les noms que les systèmes de fichiers FAT et FAT32. Les noms des fichiers et des répertoires peuvent être très longs (jusqu'à 256 caractères par nom), et ils prennent en compte la casse des lettres.

Les fichiers contiennent des données au sens large, ce peut être des documents (texte, image, film, son), des programmes, des données utilisées par le système ou tout autre type de données imaginable. En fait, les répertoires sont eux-mêmes des fichiers spéciaux, interprétés par le système différemment des autres fichiers. Les données stockées dans les répertoires sont simplement les entrées de répertoires, qui caractérisent et permettent d'avoir accès aux autres fichiers et aux autres répertoires.

Les noms de répertoires et de fichiers sont séparés par un caractère spécial. Ce caractère est traditionnellement, sous Unix, la barre oblique de division (nommée « slash » en anglais) : '/'. Les utilisateurs du DOS et de Windows prendront garde ici au fait que Microsoft a préféré la barre oblique inverse (nommée « backslash » en anglais) '\', rendant ainsi tout ses systèmes incompatibles avec les systèmes Unix, et générant ainsi beaucoup de problèmes supplémentaires là ou ils n'était pas nécessaire d'en avoir (sincèrement, le coût de cette ânerie, ainsi que celle des marqueurs de fin de ligne dans les fichiers textes, doit atteindre des sommes astronomiques dans tous les projets de portage ou de développement d'applications portables). Comme le répertoire racine n'a pas de nom, il peut être accédé directement avec un simple slash :

```
/
```

La qualification complète d'un fichier se fait en précisant le nom du répertoire à chaque niveau et en séparant par des slash chacun de ces noms. Cette qualification porte le nom de « chemin » d'accès (« path » en anglais). L'exemple suivant vous montre l'allure d'un chemin d'accès typique sous Unix :

```
/home/dupond.jean/lettres/professionnelles/marketing/ventes1999.sdw
```

La longueur maximale d'un chemin d'accès est de 4 ko dans le système de fichiers EXT2.

Les utilisateurs du DOS et de Windows constateront ici que les chemines d'accès Unix ne comportent *pas* de spécification de lecteurs. Les systèmes de fichiers Unix sont dits mono-têtes, ce qui signifie qu'ils n'ont qu'un seul point de départ : le répertoire racine (alors que les systèmes Microsoft sont multi-têtes, puisqu'ils ont un point de départ par lecteur). Le fait de n'avoir qu'un seul point de départ est beaucoup plus simple, et permet, encore une fois, d'écrire les programmes plus simplement et donc avec moins de bogues potentielles. Je sens tout de suite venir la question de la part des habitués

du DOS : « Mais alors, comment spécifie-t-on le lecteur que l'on veut utiliser ? ». Cette question a deux réponses. Premièrement, on n'accède pas aux lecteurs, mais aux systèmes de fichiers. Les utilisateurs du DOS devront donc réapprendre qu'un lecteur représente un périphérique physique, et qu'il est possible qu'il contienne plusieurs systèmes de fichiers. Ils devront également se rendre compte qu'un système de fichiers n'est pas nécessairement stocké sur un lecteur : il peut être stocké dans un fichier (c'est le cas par exemple pour les images disques de CD-ROM), accessible par le réseau (c'est le cas des systèmes de fichiers réseau, « Network File System » en anglais), ou encore générés par un composant du système (c'est le cas des systèmes de fichiers virtuels du noyau). La question n'a donc pas beaucoup de sens telle quelle. Cependant, le problème de l'accès aux systèmes de fichiers se pose malgré tout. La réponse à ce problème ci est cette fois la suivante : pour accéder à un système de fichiers, il faut réaliser une opération que l'on nomme le « montage ». Cette opération associe le répertoire racine de ce système de fichiers à l'un des répertoires de l'arborescence existante. Ce répertoire est couramment appelé « point de montage ». Par exemple, il est courant de monter le lecteur de disquette dans le répertoire `/floppy/`. Ainsi, si la disquette contient le fichier `ventes1999.sdw`, ce fichier sera accessible grâce au chemin suivant :

```
/floppy/ventes1999.sdw
```

Cette solution permet d'accéder à tous les systèmes de fichiers de la même manière, à partir d'un seul répertoire racine, que ces systèmes de fichiers soit EXT2, FAT, ISO9660, NTFS ou Amiga… En pratique, c'est nettement plus simple.

L'opération de montage peut réaliser bien plus d'opération qu'une simple association d'un système de fichier à un point de montage. En effet, elle peut générer les opérations suivantes :

- chargement des pilotes pour ce système de fichiers ;
- allocation des zones de mémoire pour les entrées/sorties en mémoire ;
- masquage des fichiers et des répertoires éventuellement présents avant le montage dans le répertoire utilisé comme point de montage.

On prendra en particulier garde à toujours démonter les systèmes de fichiers pour les lecteurs amovibles. Linux utilise en effet une partie de la mémoire que l'on appelle tampons (« buffer » en anglais), pour y stocker des données des systèmes de fichiers montés, et il n'écrit ces données que lorsque c'est nécessaire. Ce mécanisme permet d'accélérer les lectures et les écritures sur les disques, mais a l'inconvénient de nécessiter une requête de vidange des tampons (opération que l'on appelle « sync ») avant de retirer le lecteur ou avant d'éteindre le système. Si on ne le fait pas, des données seront certainement perdues. Le système effectue le sync lorsqu'il s'arrête (par l'une des commandes **halt**, **shutdown** ou **reboot**), mais il ne le fait pas si on coupe le courant brutalement. C'est pour cela qu'il faut toujours arrêter le système proprement. De manière similaire, Linux empêche l'éjection des CD-ROM tant qu'ils sont montés. En revanche, il ne peut rien faire pour les lecteurs de disquettes, c'est à l'utilisateur de prendre garde à les démonter avant de retirer la disquette.

Deux derniers points auxquels les utilisateurs de DOS et Windows devront faire attention :

- les fichiers ne sont pas identifiés par leurs extensions. Un nom de fichier peut contenir un ou plusieurs points, et une extension peut être arbitrairement longue. En particulier, un nom de fichier peut commencer par un point. Dans ce cas, ce fichier sera considéré comme caché par les programmes, et on ne les verra que si on le demande explicitement ;
- les systèmes de fichiers Unix font la distinction entre les majuscules et les minuscules. Il faut donc prendre garde à la manière dont on écrit les noms de fichiers et de répertoires. Cependant, la plupart des répertoires et des fichiers ont un nom écrit complètement en minuscules.

Nous allons à présent nous intéresser à l'organisation du système de fichiers de Linux. Ce système de fichiers contient un certain nombre de répertoires et de fichiers standards, qui ont chacun une fonction bien définie. Cette structure standard permet de gérer tous les systèmes Linux de manière homogène : les programmes savent où trouver ce dont ils ont besoin. Ils sont donc portables d'un système à un autre, et les utilisateurs ne sont pas dépaysés lorsqu'ils changent de machine (en fait, cette structure est à peu près la même pour tous les systèmes Unix, ce qui donne encore plus de poids à l'argument précédent).

Cette organisation standard du système de fichiers a été conçue de telle manière que les données et les programmes sont placés chacun à leur place, et de telle manière qu'ils puissent être facilement intégrés dans un réseau. L'intégration dans un réseau sous-entend que les fichiers des programmes peuvent être partagés par les différentes machines de ce réseau, ce qui permet d'économiser beaucoup de place. De plus, il est possible d'accéder à la plupart des ressources du système grâce à une interface uniforme. En particulier, il est possible d'accéder à tous les périphériques installés sur l'ordinateur par l'intermédiaire de fichiers spéciaux, et on peut récupérer des informations sur le système mises à disposition par le noyau simplement en lisant des fichiers générés par un pseudo système de fichiers. Le tableau suivant décrit les principaux éléments de l'arborescence du système de fichiers de Linux.

Tableau 3-2. Hiérarchie standard du système de fichiers

Répertoire	Signification
`/`	**Répertoire racine. Point de départ de toute la hiérarchie du système de fichiers. Le système de fichiers contenant ce répertoire est monté automatiquement par le noyau pendant l'amorçage du système. Ce système de fichiers est appelé système de fichiers racine (« root » en anglais).**
`/boot/`	**Répertoire contenant le noyau de Linux et ses informations de symboles. Ce répertoire est souvent le point de montage d'un système de fichiers de très petite taille, dédié au noyau. Dans ce cas, il est recommandé que le système de fichiers correspondant soit monté en lecture seule. On notera que sur certains systèmes, le noyau reste placé dans le répertoire racine. Cette technique n'est pas recommandée, car on ne peut pas monter en lecture seule la partition racine en utilisation normale du système.**

Répertoire	Signification
`/boot/vmlinuz`	**Noyau compressé de Linux. Les noyaux compressés se décompressent automatiquement lors de l'amorçage du système. Sur certains systèmes, le noyau est encore placé dans le répertoire racine de la partition root.**
`/boot/System.map`	**Fichier système contenant la liste des symboles du noyau. Ce fichier est utilisé par certains programmes donnant des renseignements sur le système. En particulier, il est utilisé par le programme « top » (programme qui indique la liste des principaux processus actifs) afin de donner le nom de la fonction du noyau dans lequel un processus bloqué se trouve.**
`/dev/`	**Répertoire contenant tous les fichiers spéciaux permettant d'accéder aux périphériques. Sous Linux, la plupart des périphériques sont accessibles au travers de fichiers spéciaux, grâce auxquels l'envoi et la réception des données vers les périphériques peuvent être réalisés de manière uniforme. Il existe un tel fichier pour chaque périphérique, et ils sont tous placés dans ce répertoire. Les distributions installent souvent dans ce répertoire les fichiers spéciaux pour la plupart des périphériques, même s'ils ne sont pas physiquement présents dans la machine. Dans ce cas, les opérations sur les fichiers spéciaux des périphériques non installés seront tout simplement refusées par le noyau. Une autre possibilité est d'utiliser un système de fichiers virtuels, géré directement par le noyau. Le répertoire `/dev/` ne contient dans ce cas que les fichiers spéciaux des périphériques pour lesquels le noyau dispose d'un gestionnaire intégré ou chargé dynamiquement. Quelle que soit la technique utilisée, ce répertoire doit être impérativement placé dans le système de fichier root.**
`/sbin/`	**Répertoire contenant les commandes systèmes nécessaires à l'amorçage et réservées à l'administrateur. Ce répertoire doit être impérativement placé dans le système de fichiers root. En général, seul l'administrateur utilise ces commandes.**
`/bin/`	**Répertoire contenant les commandes systèmes générales nécessaires à l'amorçage. Ce répertoire doit être impérativement placé dans le système de fichiers root. Tous les utilisateurs peuvent utiliser les commandes de ce répertoire.**
`/lib/`	**Répertoire contenant les librairies partagées « DLL » en anglais, pour « Dynamic Link Library ») utilisées par les commandes du système des répertoires `/bin/` et `/sbin/`. Ce répertoire doit être impérativement placé dans le système de fichiers root.**

Répertoire	Signification
`/lib/modules/`	**Répertoire contenant les modules du noyau. Ce répertoire contient les modules additionnels du noyau. Ces modules sont des composants logiciels du noyau, mais ne sont pas chargés immédiatement pendant l'amorçage. Ils peuvent en revanche être chargés et déchargés dynamiquement, lorsque le système est en fonctionnement. Il est fortement recommandé que ce répertoire soit placé dans le système de fichiers root.**
`/etc/`	**Répertoire contenant tous les fichiers de configuration du système. Ce répertoire doit être impérativement placé dans le système de fichiers root.**
`/etc/X11/`	**Dans certaines distributions, les fichiers de configuration de XWindow sont placés directement dans le répertoire `/etc/X11/`.**
`/etc/opt/`	**Répertoire contenant les fichiers de configuration des applications.**
`/tmp/`	**Répertoire permettant de stocker des données temporaires. En général, `/tmp/` ne contient que des données très éphémères. Il est préférable d'utiliser le répertoire `/var/tmp/`. En effet, le répertoire `/tmp/` ne dispose pas nécessairement de beaucoup de place disponible.**
`/usr/`	**Répertoire contenant les fichiers du système partageables en réseau et en lecture seule.**
`/usr/bin/`	**Répertoire contenant la plupart des commandes des utilisateurs.**
`/usr/sbin/`	**Répertoire contenant les commandes systèmes non nécessaires à l'amorçage. Ces commandes ne sont normalement utilisées que par l'administrateur système.**
`/usr/lib/`	**Répertoire contenant les librairies partagées de tous les programmes de `/usr/bin/` et `/usr/sbin/` et les librairies statiques pour la création de programmes.**
`/usr/include/`	**Répertoire contenant les fichiers d'en-têtes du système pour le compilateur C/C++. Les fichiers de ce répertoire sont utilisés pour réaliser des programmes dans les langages de programmation C et C++.**
`/usr/X11R6/`	**Répertoire contenant X11R6 et ses applications. Ce répertoire contient des sous répertoires `bin/`, `lib/` et `include/`, qui contiennent les exécutables de XWindow, les librairies et les fichiers d'en-têtes pour créer des programmes pour XWindow en C et C++.**
`/usr/src/`	**Répertoire contenant les fichiers sources du noyau et des applications de la distribution. Normalement, ce répertoire ne doit contenir que le code source des applications dépendantes de la distribution que vous utilisez.**

Répertoire	Signification
`/usr/src/linux/`	**Sources du noyau de Linux. Il est vivement recommandé de conserver les sources du noyau de Linux sur son disque, afin de pouvoir changer la configuration du système à tout moment.**
`/usr/local/`	**Répertoire contenant les programmes d'extension du système indépendants de la distribution. Ce n'est pas le répertoire d'installation des applications. « local » ne signifie pas ici que les programmes qui se trouvent dans ce répertoire ne peuvent pas être partagés sur le réseau, mais plutôt que ce sont des extensions du système qu'on ne trouve donc que localement sur un site donné. Ce sont donc les extensions qui ne font pas partie de la distribution de Linux utilisée, et qui doivent être conservées lors des mises à jour ultérieures de cette distribution. Ce répertoire contient les sous-répertoires `bin/`, `lib/`, `include/` et `src/`, qui ont la même signification que les répertoires du même nom de `/usr/`, à ceci près qu'ils ne concernent que les extensions locales du système, donc indépendantes de la distribution.**
`/var/`	**Répertoire contenant toutes les données variables du système. Ce répertoire contient les données variables qui ne pouvaient pas être placés dans le répertoire `/usr/`, puisque celui-ci est normalement accessible en lecture seule.**
`/var/tmp/`	**Répertoire contenant les fichiers temporaires. Il est préférable d'utiliser ce répertoire plutôt que le répertoire `/tmp/`.**
`/var/opt/`	**Répertoire contenant les données variables des applications.**
`/var/log/`	**Répertoire contenant les traces de tous les messages systèmes. C'est dans ce répertoire que l'on peut consulter les messages d'erreurs du système et des applications.**
`/var/spool/`	**Répertoire contenant les données en attente de traitement. Les travaux d'impression en cours, les mails et les fax en attente d'émission, les travaux programmés en attente d'exécution sont tous stockés dans ce répertoire.**
`/var/locks/`	**Répertoire contenant les verrous sur les ressources systèmes. Certaines ressources ne peuvent être utilisées que par une seule application (par exemple, un modem). Les applications qui utilisent de telles ressources le signalent en créant un fichier de verrou dans ce répertoire.**
`/var/cache/`	**Répertoire contenant les données de résultats intermédiaires des applications. Les applications qui doivent stocker des résultats intermédiaires doivent les placer dans ce répertoire.**

Répertoire	Signification
`/opt/`	**Répertoire contenant les applications. C'est dans ce répertoire que les applications qui ne font pas réellement partie du système doivent être installées. Les applications graphiques devraient être installées dans ce répertoire. C'est en particulier le cas des gestionnaires de bureau. Les seules applications graphiques considérées comme faisant partie du système sont les applications de X11, qui sont donc stockées dans `/usr/X11R6/`. Il est recommandé que ce répertoire soit placé sur un système de fichiers en lecture seule, et que les applications utilisent le répertoire `/var/opt/` pour travailler.**
`/home/`	**Répertoire contenant les répertoires personnels des utilisateurs. Il est bon de placer ce répertoire dans un système de fichiers indépendant de ceux utilisés par le système.**
`/root/`	**Répertoire contenant le répertoire personnel de l'administrateur. Il est donc recommandé que le répertoire personnel de l'administrateur soit placé en dehors de `/home/` pour éviter qu'un problème sur le système de fichiers des utilisateurs ne l'empêche de travailler. Toutefois, il est important que l'administrateur puisse travailler même si les répertoires `/root/` et `/home/root/` ne sont pas présents. Dans ce cas, son répertoire personnel devra être le répertoire racine.**
`/mnt/`	**Répertoire réservé au montage des systèmes de fichiers non-permanents (CD-ROM, disquettes, etc…).**
`/proc/`	**Répertoire contenant le pseudo système de fichiers du noyau. Ce pseudo système de fichiers contient des fichiers permettant d'accéder aux informations sur le matériel, la configuration du noyau et sur les processus en cours d'exécution.**

> **Note:** Les informations données ici peuvent ne pas être correctes pour votre distribution. En effet, certaines distributions utilisent une structure légèrement différente. Les informations données ici sont conformes à la norme de hiérarchie de systèmes de fichiers version 2.0 (« FHS » en anglais). Vous pouvez consulter ce document pour une description exhaustive du système de fichiers de Linux.

Vous avez pu constater que les répertoires `bin/`, `lib/`, `include/` et `src/` apparaissent régulièrement dans la hiérarchie du système de fichiers. Ceci est normal : les répertoires sont classés par catégories d'applications et par importance. Les répertoires `bin/` contiennent en général les programmes, et les répertoires `lib/` les librairies partagées par ces binaires. Cependant, les répertoires `lib/` peuvent aussi contenir des librairies statiques, qui sont utilisées lors de la création de programmes. En général, tous les systèmes Unix fournissent en standard un compilateur pour réaliser ces programmes. Dans le

cas de Linux, ce compilateur est « gcc » (pour « GNU C Compiler »). La création d'un programme nécessite que l'on dispose des fichiers sources, qui contiennent le programme écrit dans un langage de programmation, des fichiers d'en-têtes, qui contiennent les déclarations de toutes les fonctions utilisables, et des fichiers de librairies statiques, contenant ces fonctions. Ces différents fichiers sont stockés respectivement dans les répertoires `src/`, `include/` et `lib/`. Les notions de sources et de compilation seront décrites en détail dans le Chapitre 8.

Chapitre 4. Installation du système de base

Maintenant que vous connaissez tout des grands principes de Linux (et de la plupart des systèmes Unix), vous devez brûler d'impatience de l'installer et de commencer à exploiter toutes ces possibilités. Cependant, il faut ne pas tomber dans le travers de ceux qui veulent impérativement utiliser une fonctionnalité sous prétexte qu'elle est disponible. Il se peut que vous ayez envie de les essayer, parce que vous en avez été privé depuis longtemps. Ce comportement est tout à fait normal, mais il vous allez devoir refréner vos envies, pour deux raisons. La première est qu'il va falloir installer Linux au préalable, et la deuxième est qu'il ne faut pas faire un sac de n½uds avec vos données. Vous pourrez expérimenter à loisir, mais il faudra rester sérieux dès que vous toucherez au système. Comprenez bien que l'installation et la configuration de Linux doivent se faire pas à pas. Griller les étapes peut se révéler être une erreur, car cela rendrait les choses plus compliquées qu'elles ne le sont.

4.1. Récupération des informations sur le matériel

Avant de commencer quoi que ce soit, vous devriez récupérer le maximum de données concernant votre matériel. En effet, Linux n'étant pas Plug-and-Play, vous devrez certainement spécifier certains paramètres matériels pour la configuration, et dans le pire des cas, pour démarrer Linux la première fois. Les informations dont vous aurez certainement besoin sont les suivantes :

- type de processeur, avec sa marque et éventuellement sa vitesse ;
- type de carte mère, avec sa marque et impérativement son chipset ;
- date et type du BIOS (PnP, PCI, etc…) ;
- type de souris (série, PS/2, interface bus propriétaire) ;
- nombre de ports série, ainsi que leurs paramètres (ports d'entrée/sortie, lignes d'interruption) ;
- nombre de ports parallèle, ainsi que leurs paramètres (ports d'entrée/sortie, lignes d'interruption) ;
- types de disques durs (IDE, ATAPI, SCSI), ainsi que leurs tailles et leurs géométries (têtes, cylindres, secteurs par piste) ;
- si vous disposez d'un contrôleur de disque SCSI, marque et modèle de ce contrôleur (ou, à défaut, de la carte SCSI) ;
- sinon, nom du contrôleur de disque IDE (regardez sur le chipset de la carte mère !) ;
- type de lecteur de CD-ROM (IDE, ATAPI, SCSI, connecté sur carte son) ;
- type de carte son (ISA, PCI, PnP), avec son nom et sa marque ;
- nom, modèle et marque de la carte graphique, taille de sa mémoire vidéo ;
- nom, modèle et marque de l'écran, avec ses fréquences de rafraîchissement horizontales et verticales.

Il se peut que vous n'ayez pas besoin de ces informations et que l'installation se passe sans problème. Cependant, si d'aventure Linux exige que vous les connaissiez, mieux vaut pouvoir lui répondre.

4.2. Sauvegarde des données

L'installation du système de base est l'opération la plus délicate, puisqu'il faut travailler au niveau le plus bas. La moindre erreur peut provoquer une catastrophe, ce qui peut aller jusqu'à recommencer complètement l'installation. Si vous installez Linux pour la première fois sur votre ordinateur, ou s'il n'y a pas d'autres systèmes d'exploitation installé dessus, vous en serez quitte pour quelques heures perdues. Sinon, ce qui est le plus à craindre, c'est la perte totale de vos données, y compris celles des autres systèmes installés !

```
IL FAUT DONC FAIRE UNE SAUVEGARDE DE VOS DONNÉES !
```

4.3. Amorçage

Pour commencer votre installation, vous devez avant tout démarrer votre ordinateur sous Linux (sauf si vous ne disposez par de place libre sur votre disque dur pour créer une partition Linux. Dans ce cas, vous devriez lire tout de suite la section suivante, et ne pas essayer de démarrer Linux immédiatement. Vous reviendrez à cette section lorsque vous aurez réussi à dégager de la place pour les partitions Linux). La méthode la plus simple est certainement de mettre le CD d'amorçage de votre distribution dans le lecteur de CD et de redémarrer l'ordinateur. Il faut que votre BIOS soit capable d'amorcer le système sur le lecteur de CD pour que cette solution fonctionne. Si ce n'est pas le cas, il ne vous reste plus qu'à utiliser l'une des disquettes fournies avec votre distribution. Dans tous les cas, Linux doit se lancer, et le programme d'installation doit démarrer. Suivez les indications fournies avec votre distribution pour cette étape.

Le noyau de Linux qui vous a été fourni avec votre distribution est un noyau qui a été spécialement conçu pour démarrer correctement sur le plus grand nombre de machines possibles. Il contient donc les pilotes (« drivers » en anglais) pour votre disque dur, ainsi que pour d'autres types de disques. Il se peut même que plusieurs noyaux vous aient été livrés avec votre distribution. Chacun de ces noyaux permet de démarrer sur un certain type d'ordinateur. Lorsqu'il se lance, Linux tente de détecter le matériel de votre ordinateur. Seuls les pilotes correspondant à votre matériel s'activent, les autres sont tout simplement ignorés. Il se peut cependant que Linux ne détecte pas vos disques durs. Ceci peut arriver surtout pour les disques SCSI. Dans ce cas, soit il faut changer de noyau, soit il faut utiliser des pilotes additionnels, qui peuvent être chargés à la demande. Ces pilotes sont appelés des « modules » du noyau. Ces modules sont en général accessibles directement sur le CD-ROM ou la disquette de démarrage, ou sur une disquette complémentaire. Si Linux ne détecte pas vos disques durs, vous serez obligé d'indiquer les modules à charger au programme d'installation. Si cela ne fonctionne toujours pas, vous devrez réamorcer votre système avec un noyau plus adapté à votre matériel. Pour cela, vous aurez sans doute besoin de créer une disquette d'amorçage pour ce noyau, et de l'utiliser à la place de votre lecteur de CD. En général, les distributions fournissent un petit utilitaire DOS, nommé

RAWRITE.EXE, qui permet de créer une disquette d'amorçage à partir d'un noyau très simplement. Il s'utilise avec la ligne de commande suivante :

```
rawrite image lecteur :
```

où `image` est l'image d'une des disquettes d'amorçage (que vous trouverez sur le CD d'amorçage de votre distribution), et `lecteur` est le lecteur de disquette utilisé (en général, « `A :` »). Vous pouvez aussi consulter la documentation de votre distribution pour savoir comment indiquer au noyau les modules qu'il doit charger, ou comment créer une disquette de démarrage pour un autre noyau.

Dans la majorité des distributions, les outils nécessaires à l'installation sont placés sur un disque virtuel en mémoire après le démarrage de Linux. C'est en particulier le cas du programme d'installation. Ceci est normal, puisque Linux n'est pas encore installé, d'une part, et que ces outils ne tiennent pas facilement sur une disquette, d'autre part. Ces outils sont donc souvent compressés. Vous devez malgré tout avoir déjà accès au périphériques de votre ordinateur, en particulier aux disques, pour poursuivre votre installation.

4.4. Partitionnement du disque

La deuxième opération après le démarrage du système est de partitionner le disque dur. Cette étape est de loin la plus dangereuse, mais elle est absolument nécessaire. Elle ne pose pas de problème lorsqu'on installe Linux sur une machine neuve, mais elle peut être délicate si un autre système d'exploitation est déjà installé. Le problème le plus courant est l'absence de place disponible sur le disque dur pour placer une partition Linux. Ce problème n'a pas de solution : il faut supprimer une autre partition appartenant à un autre système, ou la réduire. Souvent, réduire une partition revient à sauvegarder les données, la supprimer, la recréer avec une taille inférieure et à restaurer les données. De toutes façons, dans ce type de situation, il faut faire des sauvegardes. Cependant, il existe une possibilité pour les partitions FAT. On peut défragmenter la partition afin de placer toutes les données au début de celle-ci, et modifier la variable qui contient sa taille dans la table des partitions pour la réduire. Cette opération ne reconstruit pas le système de fichiers FAT, ce qui fait que cette dernière est trop grosse pour la taille de la partition qu'elle gère après réduction. Ceci n'est pas gênant, et cette technique a l'avantage d'être rapide. La plupart des distributions de Linux fournissent l'utilitaire fips, qui permet de modifier la taille d'une partition FAT sous DOS. Attention cependant, les FAT32 et FAT32X ne sont gérées qu'à partir de la version 2.0 de fips. De plus, on vérifiera bien les options du défragmenteur de fichiers que l'on utilise : quelques outils consolident bien l'espace libre mais placent les fichiers de certains types à la fin de la partition FAT pour optimiser les temps d'accès (c'est notamment le cas avec le défragmenteur de Norton). Il faut impérativement désactiver ce type d'option avant de réduire la partition.

> **Note:** En fait, il est possible d'installer Linux sur une partition FAT, à l'aide d'une extension de ce système de fichier nommée « UMSDOS ». Cependant, cette opération n'est pas prévue par la plupart des distributions en standard, et on commence là à sortir des sentiers battus. Autant dire que l'installation risque de se compliquer sérieusement. De plus, ce choix est fondamentalement mauvais, parce que :

- le système de fichiers FAT est supporté par Linux, mais n'est pas optimisé. Il est donc environ deux fois plus lent que le système de fichiers natif ext2 ;
- le système de fichier UMSDOS simule un système de fichier Unix sur la FAT en rajoutant des fichiers d'informations complémentaires, ce qui est encore plus lent ;
- le système de fichier FAT n'est intrinsèquement pas sûr, car il ne prévoit aucun moyen de dupliquer les données ;
- le système de fichier ext2 est plus performant qu'un système de fichier FAT32, même s'il est géré par Windows 95.

Ceci dit, il faut se faire une raison, si l'on installe Linux, c'est pour l'utiliser. La solution la plus simple pour ceux qui hésitent encore est peut être de disposer d'un tiroir et d'un deuxième disque dur.

Quoi qu'il en soit, il faut déterminer les tailles de partitions. Il est normal de disposer d'au moins trois partitions :

- une partition contenant le noyau, qui doit se trouver dans les 1024 premiers cylindres pour que le BIOS puisse y accéder au démarrage ;
- une partition de swap, que Linux utilisera pour y stocker temporairement des données lorsqu'il aura besoin de récupérer un peu de place en mémoire ;
- une partition contenant les programmes et les données.

En réalité, il est même recommandé de placer les données des utilisateurs sur une partition séparée, et les programmes sur une partition accessible en lecture seule, pour minimiser les risques de pertes de données. Ceci nécessite de disposer d'encore deux autres partitions, ce qui amène le nombre de partition à cinq. Si l'on sépare également les applications (placées dans `/opt`) des programmes d'extensions du système (placés dans `/usr`), il faut même 6 partitions. Comme les tables de partitions ne contiennent que quatre entrées, il faut utiliser des partitions logiques. Linux sait parfaitement gérer ce type de partitions, de manière complètement transparente. L'avantage de cette organisation est une très grande sûreté du système. Son inconvénient est évidemment qu'il faut connaître à l'avance ses besoins en place disque, aussi bien pour les programmes que pour les données et le swap. Cependant, d'une manière générale, on peut considérer que ce qui est le plus important pour un particulier, ce sont ses données, et que le fait de réinstaller le système en cas de perte totale d'une partition n'est pas le plus gros problème. Cela dit, il est très rare que le système de fichiers ext2 soit corrompu, et quand il l'est, il est peut être réparé facilement.

Un particulier peut installer et essayer des programmes très couramment, il est donc logique de regrouper l'espace de données utilisateur avec l'espace d'installation des programmes. En revanche, il reste prudent de continuer à placer le noyau sur sa propre partition, ne serait-ce que pour simplifier l'installation. Vous devrez également prendre en compte l'éventualité d'une réinitialisation complète du système de fichiers système pour une nouvelle installation ou une mise à jour du système. Il peut être agréable dans ce cas de disposer d'une partition indépendante pour les données utilisateurs, sur

laquelle vous pourrez sauvegarder vos fichiers de configuration… L'une des solutions les plus simple consiste évidemment à avoir un deuxième disque, auquel cas le problème ne se pose plus.

Toutefois, si vous décidez de placer les programmes et les données utilisateurs dans la même partition, vous pouvez envisager les tailles de partitions suivantes :

- partition réservée au noyau pour le démarrage : 1 cylindre du disque (il faut au moins 2 Mo, ce qui est inférieur à la taille d'un cylindre sur la plupart des disques durs récents). Il est inutile de consacrer plus de 4 Mo pour cette partition, elle ne contiendra au plus que deux noyaux et quelques autres fichiers ;
- partition de swap : 2 fois la mémoire vive de l'ordinateur, avec pour limite supérieure 128 Mo. Il est inutile de consacrer plus de place que cela, car il est très rare d'avoir besoin de plus de 192 Mo de mémoire virtuelle (la mémoire virtuelle est la somme de la mémoire libre et de la taille du swap). 128 Mo de swap sont donc largement suffisant pour tout type d'opération. De toutes façons, si un jour il fallait plus de swap, on pourrait créer un fichier de swap temporaire ;
- partition root, contenant le système, les programmes et les données utilisateurs : 1,5 Go au moins. Il est possible de travailler avec 1 Go, mais on n'est vraiment pas à l'aise avec XWindow et si l'on désire compiler des programmes récupérés sur Internet (ce qui est assez courant).
- Il est également recommandé de placer la partition de swap à la fin du disque dur. Si jamais le système se plante, il ne risque pas de déborder et d'écrire accidentellement sur les autres partitions. L'ordre recommandé pour les partitions est donc partition de démarrage, partition utilisateur et partition de swap.

> **Note:** La taille de 128 Mo pour le swap n'est plus une limitation à partir de la version 2.2 du noyau. En effet, la gestion du swap a été modifiée et la nouvelle limite est à présent 2 Go par partition de swap, avec un total de 8 partitions possibles. Si j'indique ici une taille maximale de 128 Mo pour le swap, c'est simplement parce que c'est tout à fait suffisant et que si on a besoin de plus, il vaut mieux redimmensionner sa configuration et envisager de gonfler la machine en mémoire vive.

Le partitionnement peut se faire soit directement à l'aide du **fdisk** de Linux, soit par l'intermédiaire du programme d'installation de la distribution correspondante. Il est recommandé d'utiliser ce programme d'installation, qui vous guidera et vous indiquera comment réaliser cette opération. Si toutefois vous désirez utiliser **fdisk**, il vaut mieux faire attention à ce que vous faites. Pour lancer **fdisk**, il suffit de taper la commande suivante en ligne de commande :

```
fdisk disque
```

où `disque` est le fichier spécial de périphérique représentant le disque que vous désirez partitionner.

Comme on l'a vu précédemment, tous les périphériques sont accessibles par l'intermédiaire de fichiers spéciaux placés dans le répertoire `/dev/`. Le nom des fichiers spéciaux correspondant à vos disques peut varier, selon leur nature. Ainsi, les disques et les lecteurs de CD IDE sont accessibles par l'intermédiaire des fichiers spéciaux `/dev/hda`, `/dev/hdb`, `/dev/hdc`, etc… Ces fichiers permettent d'accéder respectivement au disque maître du premier contrôleur IDE, au disque esclave de

ce même contrôleur, puis au disque maître du deuxième contrôleur IDE puis au disque esclave du deuxième contrôleur IDE, etc... Ainsi, si vous ne disposez que d'un disque, il doit normalement être connecté sur le premier contrôleur IDE et être maître. Dans ce cas, vous y accéderez par le fichier spécial `/dev/hda`. Pour les disques SCSI, les noms légèrement différents : ils sont nommés `/dev/sda0`, `/dev/sdb`, etc...

Donc, si vous voulez partitionner le disque maître du premier contrôleur IDE, vous devrez taper :

```
fdisk /dev/hda
```

Si vous ne spécifiez aucun disque en paramètre à fdisk, il prendra par défaut le disque `/dev/sda`, ou `/dev/hda` si aucun disque SCSI n'est installé.

> **Note:** À partir de la version 2.4.0 du noyau de Linux, le répertoire `/dev/` peut être généré par le noyau, dans un système de fichiers virtuel. L'organisation de ce répertoire est dans ce cas différente de l'organisation classique, et les noms de fichiers spéciaux correspondants aux disques peuvent être différents de ceux indiqués ci-dessus. Par exemple, le chemin du fichier spécial permettant d'accéder au premier disque IDE sera `/dev/ide/host0/bus0/target0/disc`. Comme vous pouvez le constater, la représentation de la machine dans le système de fichiers virtuels est plus structurée, mais également plus compliquée. Afin de simplifier ces mécanismes, il est d'usage de placer des liens symboliques dans le répertoire `/dev/` permettant d'accéder aux fichiers spéciaux de périphériques avec leurs anciens noms. Vous n'aurez donc normalement pas à vous soucier de savoir si votre noyau utilise ou non le système de fichiers virtuels `/dev/`, et les chemins utilisés dans la suite de ce document seront les chemins classiques. Vous devrez faire la traduction vers les chemins du système de fichiers virtuels vous-même si vous ne voulez pas utiliser ces liens symboliques.

fdisk est un programme très peu interactif. Il attend que vous lui communiquiez les commandes à exécuter en tapant sur une lettre. Les différentes commandes possibles peuvent être affichées avec la commande '`m`'.

Lorsque vous créez une partition, vous devez utiliser la commande '`n`', puis indiquer son type. Il existe deux types de partitions. Les partitions primaires sont définies directement dans une petite table enregistrée dans le secteur d'amorçage principal, encore appelé « MBR » (abréviation de « Master Boot Record » en anglais). Outre les informations décrivant les partitions primaires, ce secteur contient un petit programme permettant de charger le système d'une de ces partitions (on appelle ce programme le « bootstrap loader »). Il ne peut y avoir plus de quatre partitions primaires, parce que seulement quatre entrées sont définies dans la table des partitions du MBR. Les partitions étendues permettent de repousser cette limite à 64, en recréant une table des partitions plus grande dans une des partitions primaires. Il ne peut y avoir qu'une seule partition étendue sur un disque donné (mais cela n'empêche pas d'avoir des partitions primaires à côté de celle-ci).

Lorsque vous aurez choisi le type de partition désiré (avec les commandes '`p`' ou '`e`', respectivement pour « primary » et « extended »), vous devrez donner son numéro dans la table des partitions, puis indiquer le début et la fin de la partition. Par défaut, l'unité utilisée par **fdisk** est le cylindre. Il est recommandé de conserver cette unité, surtout si l'on utilise un système qui ne sait manipuler que les cylindres (DOS et Windows par exemple). Toutefois, on peut changer cette unité grâce à la commande

'u' et utiliser le secteur comme unité.

Si vous avez créé une partition étendue, celle-ci sera utilisée pour y stocker des partitions logiques. Pour pouvoir les créer, il faut encore utiliser la commande 'n', et choisir le type de partition logique avec la commande 'l'. Les partitions logiques sont numérotées avec les nombres 5 et suivants. La création des partitions logiques se fait exactement de la même manière que les partitions primaires, en spécifiant leur début et leur fin, soit en cylindre, soit en secteurs selon l'unité courante.

Les partitions primaires et logiques ont toutes un identificateur qui permet d'indiquer à quel système elles appartiennent. Les identificateurs à utiliser pour Linux sont 83 pour les partitions en EXT2, et 82 pour les partitions de swap. Cet identificateur peut être fixé à l'aide de la commande 't' (pour « type »), qui demande successivement le numéro de la partition à modifier et la valeur de son identificateur en hexadécimal. La liste des valeurs admissibles par **fdisk** peut être obtenue à l'aide de la commande 'l'. Par défaut, **fdisk** crée des partitions Linux natives, de code 83.

Lorsque vous aurez créé toutes vos partitions, il ne vous reste plus qu'à activer la partition qui contient le secteur d'amorçage (« boot sector » en anglais). Ceci se fait à l'aide de la commande 'a'. C'est sur cette partition que le chargeur du MBR ira chercher le système à lancer. En général, le lancement du système consiste à exécuter le programme enregistré dans le premier secteur de la partition de ce système. On appelle ce secteur le secteur de boot. On ne peut activer qu'une seule partition, c'est donc sur celle-ci qu'il faudra installer le système principal, ou plus généralement le gestionnaire d'amorçage de ce système. Un gestionnaire d'amorçage est un petit programme capable de lancer des systèmes d'exploitation. Il est en général constitué d'une petite partie placée sur le secteur de boot de la partition du système auquel il appartient, et d'un complément enregistré dans des fichiers de cette partition. Le fait de stocker le gestionnaire d'amorçage dans le secteur de boot n'est pas gênant, parce que les gestionnaires d'amorçages des systèmes sont capables d'amorcer d'autres systèmes que celui avec lequel ils ont été fournis (ce n'est cependant pas le cas pour DOS et Windows 9x). En fait, il est fortement recommandé d'installer les gestionnaires d'amorçage sur la partition du système avec lequel il est fourni, et non pas directement dans le MBR. En effet, certains systèmes d'exploitation (ceux de Microsoft) écrasent systématiquement le MBR lorsqu'ils s'installent, détruisant ainsi les chargeurs des autres systèmes qui y seraient installés. Ceci implique que si l'on désire installer un gestionnaire d'amorçage autre que celui des systèmes Microsoft sur le MBR, il faut le faire après l'installation de ces systèmes. En pratique, cela veut dire que dans ce cas, on doit installer Linux après Windows ou le DOS.

Le gestionnaire d'amorçage de Linux se nomme « LILO » (pour « LInux LOader »). LILO permet de démarrer un grand nombre de systèmes, dont DOS, Windows et OS/2. Comme nous venons de l'expliquer, il est conseillé d'installé LILO sur le secteur de boot de la partition Linux, et c'est cette partition qui doit être marquée comme amorçable. Cependant, il vous est parfaitement possible d'installer LILO directement sur le MBR, même si d'autres systèmes d'exploitation que Linux sont déjà installés. En revanche, on prendra garde dans ce cas à toujours installer les éventuels systèmes d'exploitation Microsoft en premier et à faire une sauvegarde de son MBR. Nous verrons comment installer LILO plus loin.

4.5. Création des systèmes de fichiers

Une fois le disque correctement partitionné, il faut créer les systèmes de fichiers. Cette opération n'est pas nécessaire pour les partitions de swap, cependant il faut le faire pour les autres partitions. La création des systèmes de fichiers ne sera décrite que pour le système de fichiers natif de Linux, à savoir EXT2. Attention ! Dans le monde du DOS et de Windows, la création d'un système de fichiers est l'opération de « formatage » d'une partition. Le terme de formatage est très mal employé dans ce sens, car il n'y a strictement rien à voir entre le formatage d'un disque, qui est l'opération consistant à enregistrer des marques sur le support magnétique pour définir les pistes et les secteurs du disque, et la création du système de fichiers d'une partition, qui consiste à enregistrer les structures de données permettant de retrouver les fichiers dans cette partition. Le formatage normalement est effectué par le fabricant du disque, et a lieu avant le partitionnement. En effet, partitionner un disque suppose qu'il existe déjà des pistes et des secteurs sur le disque. C'est pour cette raison que l'on ne parlera ici que de création de systèmes de fichiers.

La méthode décrite ici permet de créer un système de fichiers EXT2. Cette opération peut être faite à l'aide de la commande **mke2fs**. Afin de pouvoir utiliser **mke2fs** correctement, il est nécessaire de définir quelques termes, et d'expliquer à quels notions d'EXT2 ils se réfèrent.

Premièrement, le système de fichiers EXT2 travaille, comme la plupart des systèmes de fichiers, avec des blocs de taille fixe (« clusters » en anglais). Ceci signifie que l'allocation de l'espace disque se fait par multiple de la taille de ces blocs : il est impossible de demander seulement une partie d'un bloc. Cette technique présente ses avantages et ses inconvénients. Essentiellement, l'avantage est la rapidité engendrée par la simplification des mécanismes d'allocation et de libération d'espace disque. L'inconvénient majeur est évidemment qu'on perd de la place pour tous les fichiers qui ne tiennent pas dans un nombre entier de blocs, puisqu'il faut allouer un bloc supplémentaire qui sera partiellement utilisé. En moyenne, on perd la moitié d'un bloc par fichier, ce qui ne peut être réduit qu'en limitant la taille des blocs à une valeur relativement faible. EXT2 gère l'allocation et la libération des blocs, de telle sorte qu'il peut toujours trouver le meilleur bloc à allouer pour créer un fichier. De cette manière, il limite la fragmentation des fichiers à son strict minimum.

Deuxièmement, EXT2 utilise des structures de données appelées « inode » pour définir les fichiers. Un inode contient la plupart des informations d'un fichier, à savoir :

- son propriétaire et son groupe ;
- ses droits d'accès ;
- ses dates de création, modification, accès ;
- les blocs qu'ils utilise ;
- d'autres informations utilisées par EXT2.

Ces inodes sont stockés dans une table du système de fichiers, ce qui permet d'accéder très rapidement à toutes ces informations et de retrouver également très simplement le ou les blocs contenant les données du fichiers. Le problème est ici que cette table a un nombre d'entrées limité, ce qui implique un nombre limité de fichiers dans le système de fichiers. Plus cette table est grande, plus le nombre de fichiers que l'on pourra créer est grand, et inversement. Il faut donc trouver un compromis entre la taille de cette table et le nombre de fichiers que l'on est susceptible de créer. Il va de soi qu'en général, les grandes partitions contiennent plus de fichiers, mais que la table d'inode peut également

avoir une taille supérieure sans que cela ne soit dérangeant. Par conséquent, il est relativement courant de définir le taux d'inode par bloc, ou autrement dit la proportion d'inode dans la partition par rapport à sa taille.

Toutes ces informations (blocs libres et inodes) sont sauvegardées à plusieurs endroits dans la partition, ce qui permet de disposer en permanence de copies de la structure du système de fichiers. De cette manière, il est relativement simple de réparer un système de fichiers endommagé. Chacune de ces copie s'appelle un groupe de blocs. Chaque groupe de blocs contient un bloc particulier, le « super bloc », qui contient la description de son groupe.

Lors de la création du système de fichiers, il est nécessaire d'indiquer la taille d'un bloc en octets. Cette taille doit impérativement être un multiple de la taille d'un secteur du support physique de données, parce que les blocs ne peuvent contenir qu'un nombre entier de secteurs. Pour un disque dur, la taille des secteurs est fixée à 512 octets, ce qui fait que la taille d'un bloc est au moins de 512 octets. De même, il faut spécifier le nombre d'inode de la partition. Il est possible de spécifier ce nombre soit directement, ou d'indiquer seulement le taux du nombre d'octets par inode de la partition. Il faut savoir que le nombre maximum d'inode possible est bien entendu le nombre total de blocs, puisque tout fichier non vide requiert au moins un bloc. Si vous ne savez pas quelles valeurs prendre, vous pouvez utiliser des blocs de 1024 octets (2 secteurs), et un rapport de 4096 octets par inode (donc de 4 blocs de 1 Ko par inode).

La syntaxe de la commande **mke2fs** est donnée ci-dessous :

```
mke2fs fichier
```

où `fichier` est le nom du fichier spécial de périphérique représentant la partition sur laquelle le système de fichiers doit être créé. **mke2fs** prend alors des valeurs par défaut appropriées pour cette partition, mais vous pouvez également spécifier d'autres valeurs. La taille des blocs peut être indiquée en octets avec l'option suivante :

```
mke2fs -b taille fichier
```

où `taille` représente la taille d'un bloc en octets. De même, le nombre d'octets par inode peut être précisé avec l'une des options `-i` :

```
mke2fs -i octets fichier
```

où `octets` est le rapport de la taille de la partition en octets par le nombre d'inodes à créer. Il est possible d'indiquer directement ce nombre avec la commande suivante :

```
mke2fs -N nombre fichier
```

4.6. Création de la partition de swap

En cas de manque de mémoire vive, Linux peut utiliser une partie du disque dur pour y stocker les données temporairement inutilisées afin de libérer de l'espace mémoire. Cette opération permet de

continuer à travailler, même si la machine ne dispose pas de suffisamment de mémoire vive pour exécuter tous les processus dont elle a la charge. L'inconvénient de cette méthode est évidemment la dégradation des performances, mais c'est un bon compromis si l'on considère le prix du méga-octet de mémoire par rapport à celui des disques dur d'une part, et le fait qu'il vaut mieux parvenir à faire son travail, même lentement, que de ne pas le faire du tout.

L'espace disque consacré par Linux pour ce stockage temporaire est appelé « swap », du terme anglais « to swap » qui fait référence à l'échange des données de la mémoire vers le disque dur (et inversement, lorsqu'elles sont rechargées en mémoire). Linux est capable de gérer plusieurs formes de swap. Il est capable d'utiliser des fichiers de swap, qui sont stockés dans un système de fichiers, ou les partitions de swap. Ces dernières ont l'avantage d'être bien plus rapides, puisque le noyau n'a pas à se préoccuper de la structure du système de fichiers lors des opérations de swap (qui, rappelons-le, constituent déjà un ralentissement notable de la machine). En revanche, elles ont l'inconvénient majeur d'être très contraignantes, puisqu'elles nécessitent de réserver une partition pour le swap de manière permanente. Cependant, il est tout à fait acceptable de nos jours de consacrer 64 ou 128 Mo de disques dur pour une partition de swap. Linux est capable de gérer jusqu'à 8 partitions de swap, plus les fichiers de swap que l'on peut rajouter ultérieurement. Nous ne décrirons que la manière de créer une partition de swap, car les fichiers ne constituent plus le meilleur compromis avec les tailles de disques que l'on rencontre à présent.

Bien entendu, le programme d'installation de votre distribution prend certainement déjà en charge la création des partitions de swap. Il est donc recommandé, encore une fois, d'utiliser ce programme, même si la description qui suit vous permettra de comprendre ce dont il s'agit.

Les partitions de swap peuvent être crées, comme toutes les partitions, à l'aide du programme **fdisk**. En fait, la seule distinction entre une partition de swap et une partition EXT2 est tout simplement son identificateur. Comme on l'a déjà vu lors du partitionnement du disque, l'identificateur utilisé pour les partitions de systèmes de fichiers est 83, et celui pour les partitions de swap est 82. Vous devrez donc affecter cet identificateur à votre partition de swap lorsque vous partitionnerez votre disque dur.

Une fois créée, la partition de swap peut être préparée pour que le noyau puisse l'utiliser. Cette préparation revient à peu près à formater un système de fichiers, à ceci près que les structures de données écrites dans la partition de swap sont beaucoup plus simples car il ne s'agit plus ici de stocker une arborescence complète de fichiers. La commande **mkswap** permet de préparer les partitions pour être utilisées en tant que partition de swap. Elle s'utilise selon la syntaxe suivante :

```
mkswap -c partition
```

où `partition` est la partition à préparer pour le swap. Notez qu'en réalité, **mkswap** peut tout aussi bien travailler sur un fichier que sur une partition.

Lorsque la partition aura été préparée pour le swap, il est possible de demander à Linux de l'utiliser avec la commande suivante :

```
swapon partition
```

où `partition` est la partition de swap à utiliser. Cette zone de swap est alors automatiquement prise en compte par le système. La commande suivante permet d'arrêter le swapping sur une partition :

```
swapoff partition
```

Normalement, vous n'aurez jamais à utiliser ces commandes manuellement. Le programme d'installation de votre distribution configure le swap, et fait en sorte que les partitions de swap soient chargés automatiquement lors du démarrage de la machine. Notez cependant que cette méthode de configuration dynamique permet d'ajouter temporairement un peu plus de swap si les besoins s'en font sentir, sans avoir à redémarrer la machine.

4.7. Installation des composants de base

Si vous êtes arrivés jusqu'ici, vous avez fini les opérations les plus risquées et sans doute les plus difficiles. Dans les premiers jours de Linux, il fallait installer à la main les différents composants du système, voire les recompiler soi-même. Heureusement, toutes les distributions actuelles disposent aujourd'hui de programmes d'installation du système de base, qui simplifient beaucoup le travail. Les opérations que vous allez réaliser à présent sont donc plus simples, et certainement moins dangereuses.

L'ordre logique est évidemment de commencer par les couches les plus basses du système, donc en particulier le noyau. Cependant, il faut également installer le shell et les outils complémentaires, puisque sans eux on ne peut strictement rien faire de ce noyau. Cet ensemble de programme constitue ce que l'on appelle le système de base. Je suggère de configurer correctement le système à ce stade, avant de se lancer dans l'aventure XWindow. C'est évidemment cette partie qui sera la plus difficile à réaliser. L'installation de XWindow ne devrait alors pas causer de problèmes majeurs. Une fois ces opérations réalisées, vous pourrez installer les applications. En général, lorsque le système est bien configuré, l'installation des applications est une tâche relativement facile et se fait rapidement. Ce document ne traitera pas en détail de l'installation des applications, car il y en a trop pour que l'on puisse donner des informations valides pour toutes les applications. Quelques règles générales seront malgré tout données, car elles peuvent s'appliquer pour certaines applications.

Toutes les distributions organisent les différents composants logiciels qu'elles fournissent en paquetages (« package » en anglais). Ainsi, l'installation du système se fait par groupes homogènes de fichiers, et le regroupement dans un paquetage est généralement une dépendance forte (en pratique, ce sont les fichiers d'une même application). En installant un paquetage, on installe finalement un logiciel particulier. Cependant, certains paquetages dépendent d'autres paquetages, par exemple, les paquetages contenant le système de base sont évidemment utilisés par tous les autres paquetages. Les programmes d'installation gèrent relativement bien les dépendances et les conflits entre paquetages, si bien que l'installation peut maintenant se faire sans trop de problèmes.

Afin d'organiser un peu tout ces paquetages, les distributions les trient souvent par « séries ». Une série n'est rien d'autre qu'un ensemble de paquetages regroupés par domaine fonctionnel. Ceci signifie que l'on peut facilement retrouver un paquetage donné, en allant le chercher dans la série contenant tous les paquetages fonctionnellement proches. Le regroupement des paquetages en séries ne signifie absolument pas que tous les paquetages de la série doivent être installés pour obtenir une fonctionnalité donnée, mais que les logiciels qui s'y trouvent ont plus ou moins trait à cette fonctionnalité. En fait, il peut même y avoir redondance ou conflit entre deux paquetages d'une même série. Dans ce

cas, il faut choisir l'un ou l'autre, selon ses besoins personnels.

Certains paquetages sont indispensables pour le systèmes, d'autres sont purement optionnels. Mais la plupart des paquetages sont simplement les paquetages des applications, et vous devrez faire le tri et choisir ceux qui vous intéressent parce qu'il est impensable d'installer tous les paquetages (une distribution de Linux peut faire 5 à 6 CD-ROM, en tenant compte du système, des applications et des sources). Les seuls paquetages qu'il faut impérativement installer, ce sont les paquetages de la série de base. En général, cette série porte le nom A, ou AAA, ou quelque chose de similaire, afin qu'elle puisse être toujours en tête de liste dans les programmes d'installation. Cette série comprend au moins les paquetages des commandes Unix de base, du shell, du programme d'installation et de tous les fichiers nécessaires au fonctionnement du système (fichiers de configuration, scripts et librairies partagées). Tant que les programmes de cette série sont intacts et fonctionnels, le système est utilisable. S'il en manque, les problèmes peuvent survenir à tout moment : de l'absence ou l'indisponibilité de la documentation à l'impossibilité complète de démarrer le système.

Le choix des paquetages à installer est crucial mais non définitif. En effet, si vous avez installé un paquetage dont vous n'avez pas ou plus besoin, rien ne vous empêche de le supprimer par la suite. De même, si vous vous apercevez que vous avez oublié d'installer un paquetage dont vous avez besoin, vous pouvez le réinstaller ultérieurement.

Il n'est pas possible de donner ici la liste des paquetages que vous devez installer, car cette liste dépend beaucoup trop de la distribution que vous possédez. Il est conseillé de lire le manuel de cette distribution ou de bien lire les écrans d'aides du programme d'installation. Cependant, les paquetages dont vous aurez certainement besoin pour poursuivre l'installation sont sans doute les suivants :

- paquetage du système de base ;
- paquetage du compilateur GCC ;
- paquetage du réseau ;
- paquetage de XWindow ;
- paquetage de documentation ;
- paquetages susceptibles de faire fonctionner votre matériel (carte son, serveurs XWindow approprié à votre carte graphique, etc…).

Gardez à l'esprit que dans le monde du logiciel libre, les programmes sont souvent distribués sous la forme de fichiers sources et que vous aurez sans doute besoin des outils de développement pour les compiler et les installer. Veillez donc à inclure d'office tous ces outils, même si vous ne désirez pas programmer personnellement. Bien entendu, vous pourrez revenir ultérieurement dans le programme d'installation et réinstaller un paquetage si vous l'avez oublié pendant la phase d'installation.

Lorsque vous aurez installé votre système de base, vous devrez faire en sorte qu'il puisse démarrer. Pour cela, il existe plusieurs possibilités :

- soit vous démarrez à partir d'une disquette ou d'un CD d'amorçage ;
- soit vous lancez Linux à partir du DOS grâce au programme « LOADLIN.EXE » ;

- soit vous installer le gestionnaire d'amorçage LILO.

Il va de soi que c'est la troisième solution qui est recommandée. Cependant, les deux premières solutions pourront être utiles si d'aventure votre MBR se trouvait écrasé par un système d'exploitation maladroit (comme DOS ou Windows).

L'installation de LILO doit se faire de préférence dans le programme d'installation ou de configuration de votre distribution. Consultez votre documentation pour plus de détails à ce sujet. Cependant, si vous désirez installer LILO vous même, vous devrez modifier le fichier de configuration `/etc/lilo.conf`. Ce type d'opération est décrit en détail dans la Section 6.4.

Chapitre 5. Commandes de base d'Unix

Si vous êtes parvenu à installer les paquetages de la série de base, vous disposez d'un système Linux fonctionnel. Félicitations ! Maintenant, il va falloir le configurer… Autant vous prévenir tout de suite : cette opération demande beaucoup de temps et de patience, à moins d'avoir une machine vraiment standard et une chance phénoménale. Mais pour cela, il va falloir que vous appreniez les commandes de base d'Unix et la manière d'utiliser un système Linux en ligne de commande (c'est à dire en mode texte, sans XWindow).

5.1. Login et déconnexion

Comme il l'a déjà été dit, Linux est un système multi-utilisateur. Il faut donc que chacun s'identifie pour que le système puisse fonctionner correctement. Cette opération est réalisée lors de l'opération dite de *login* (du verbe anglais « to log in », qui signifie « s'enregistrer » dans le système). Le login consiste essentiellement à taper son nom d'utilisateur, valider, et répondre éventuellement à la demande de mot de passe de la part du système.

Le login doit être la première opération à effectuer. Il est impossible d'accéder au système d'une autre manière, et la vérification du mot de passe fournit l'authenticité de l'utilisateur qui se logue. Ainsi, le système sait en permanence au nom de quelle personne il effectue les opérations demandées. Cette opération est à la base des mécanismes de sécurité et de personnalisation du système pour chaque utilisateur.

Il existe deux types de login. Le plus courant est le login en mode texte, qui peut être fait directement sur le poste local, ou à travers un réseau. Le système vous invite à vous identifier avec la ligne suivante :

```
login :
```

D'autres informations peuvent être affichées avant le mot `login`, qui peuvent vous renseigner sur la nature du système. Quoi qu'il en soit, vous devez taper votre nom d'utilisateur (que l'on appelle simplement « login »…), ou « root » si vous désirez vous connecter en tant qu'administrateur. Le système vous demande alors le mot de passe avec la ligne suivante :

```
password :
```

Bien entendu, vous ne devrez jamais oublier votre mot de passe administrateur. Si toutefois cela vous arrive, vous n'aurez plus qu'une seule solution : démarrer l'ordinateur à partir d'une disquette système, et faire le ménage dans le fichier de mots de passe. Cette opération n'est jamais très agréable à réaliser. Conclusion : n'oubliez jamais votre mot de passe.

Le deuxième type de login est le login graphique, sous X11. Ce type de login a en général lieu sur un terminal X (c'est à dire un terminal graphique). La procédure peut varier selon l'environnement

utilisé, mais le principe reste toujours le même : il faut fournir son nom d'utilisateur et son mot de passe.

Si, comme la plupart des gens, vous ne cherchez pas à utiliser pas votre ordinateur à distance à travers un réseau, vous vous connecterez quasiment toujours en local. Linux fournit, pour l'utilisateur local, plusieurs terminaux virtuels. Ceci signifie qu'il est possible de se connecter plusieurs fois dans le système dans des terminaux différents. Pour passer d'un terminal virtuel à un autre, il suffit de taper les combinaisons de touches `ALT+DROITE` ou `ALT+GAUCHE`, où `DROITE` et `GAUCHE` sont respectivement les flèches du curseur droite et gauche. Il est également possible d'accéder à un terminal donné à l'aide de la combinaison de touche `ALT+Fn`, où `Fn` est l'une des touches de fonction `F1`, `F2`, `F3`, etc... La plupart des distributions utilisent au moins quatre terminaux virtuels, plus un terminal X. Le terminal X est le terminal graphique, qui fonctionne sous XWindow. Vous noterez sans doute que lorsqu'on est sous XWindow, les combinaisons `ALT+Fn` ont une autre signification. Elles ne peuvent donc pas être utilisées pour basculer vers les terminaux en mode texte. Pour remédier à ce problème, une autre combinaison de touche a été définie, spécialement pour XWindow : `CTRL+ALT+Fn`. Il suffit donc simplement d'utiliser la touche `CTRL` en plus de la touche `ALT`.

L'utilisation des terminaux virtuels est très pratique, même pour un seul utilisateur. En effet, ceux-ci permettent de lancer plusieurs programmes simplement, à raison d'un par terminal virtuel, et de s'y retrouver ainsi plus facilement. Pour ceux qui ne connaissent pas les systèmes Unix, il est recommandé de jouer un peu avec les terminaux virtuels, afin de simuler la présence de plusieurs utilisateurs. Ils auront ainsi un aperçu de la puissance de ces systèmes.

Lorsque l'on a fini de travailler, il faut se déconnecter. Cette opération est très simple pour les terminaux non graphiques, puisqu'il suffit de taper la commande suivante :

```
logout
```

Si d'aventure cette commande ne fonctionnait pas, vous pourrez utiliser la commande **exit**, qui termine le shell courant (y compris le shell de login).

Pour les terminaux X, le processus de déconnexion dépend de l'environnement utilisé. Il faut tâtonner un peu, et normalement on trouve une option de menu du style « logout » ou « déconnexion ». Vous pouvez par exemple cliquer sur le bouton droit de la souris sur le bureau de l'environnement de travail, afin d'appeler un menu contextuel. Dans bien des cas, ce menu contient une option de déconnexion.

Il est très important de se déconnecter et de ne jamais laisser une session ouverte. En effet, cette négligence peut vous coûter cher, car une personne mal intentionnée pourrait très bien utiliser ce terminal à vos dépend. Il aurait tous vos droits, et effectuerait ses opérations en votre nom. La sécurité du système garantissant que vous seul pouvez vous connecter sous ce nom, grâce au mot de passe, vous seriez donc responsables des agissements de l'intrus. Bien entendu, ce genre de considération n'a pas autant d'importance pour un particulier que dans une entreprise ou une collectivité quelconque.

5.2. Arrêt et redémarrage du système

Il faut bien comprendre que Linux, tout comme la plupart des systèmes d'exploitation modernes, ne peut pas être arrêté en éteignant directement l'ordinateur, comme on le faisait autrefois avec le DOS. En effet, la plupart des systèmes d'exploitation utilisent une partie de la mémoire de l'ordinateur pour y stocker temporairement les données qui ont été lues à partir du disque et celles qui doivent y être écrites. Cette zone de mémoire constitue ce qu'on appelle un *tampon* (« buffer » en anglais), et elle sert à accélérer les accès aux périphériques plus lents, que sont les disques durs et lecteurs de CD-ROM. Il va de soi qu'une requête de lecture sur des données déjà situées en mémoire est infiniment plus rapide que si elles ne s'y trouvaient pas. Il est en revanche plus difficile de comprendre pourquoi les requêtes d'écriture doivent être différées. La raison est la suivante : le système préfère effectuer l'écriture physique sur le disque principalement parce qu'une autre requête d'écriture dans la même zone du disque peut très bien avoir lieu ultérieurement. Si les données qui n'ont pas été écrites sont ainsi modifiées par une requête ultérieure, il n'est plus nécessaire de les écrire, et ainsi le système peut économiser un temps précieux en ne le faisant pas. Si les données à écrire sont contiguës à celles d'une requête précédente, le système peut les écrire en bloc, ce qui est toujours plus rapide que de faire plusieurs écritures partielles (notamment parce que les têtes de lecture du disque n'ont pas à être déplacées). Enfin, si les données qui doivent être écrites font l'objet d'une requête de lecture, il va de soi qu'elles sont immédiatement accessibles. On voit que cette stratégie permet de travailler beaucoup plus vite. De facto, Linux utilise toute la mémoire vive libre pour ses tampons d'entrées / sorties, ce qui en fait un système extrêmement performant. Le gain en performances peut facilement atteindre un facteur 3 ou 4.

Le problème majeur est évidemment que si on éteint l'ordinateur brutalement, les données dont l'écriture a été différée sont perdues. Pire, parmi ces données, il est probable qu'il y ait des informations vitales pour le système de fichiers, ce qui fait qu'il risque fort d'être endommagé. En général, le système effectue un contrôle et une réparation si nécessaire lors du redémarrage suivant, mais il est inutile de prendre des risques. Tout ceci signifie qu'il est impératif de prévenir le système avant de l'arrêter, pour qu'il puisse écrire les données situées dans ses tampons.

L'arrêt du système est une opération qui est du ressort de l'administrateur. On ne peut donc le réaliser que sous le compte root. Plusieurs commandes sont disponibles, les plus simples sont données ci-dessous :

halt

Permet d'arrêter le système.

reboot

Permet de le redémarrer.

Ces commandes sont en fait des scripts permettant d'effectuer les opérations d'arrêt et de redémarrage du système rapidement. Si elles ne sont pas disponibles sur votre distribution, vous devrez sans doute utiliser la commande générique suivante :

```
shutdown [-r] now
```

où l'option `-r` permet de demander un redémarrage et non un arrêt simple.

Il est également possible que votre gestionnaire de bureau vous donne le moyen d'arrêter l'ordinateur par l'intermédiaire de l'interface graphique de X11. La technique à utiliser dépend évidemment de l'environnement que vous aurez installé, et elle ne sera pas décrite ici. Consultez la documentation de votre distribution pour plus de détails à ce sujet. De plus, la plupart des distributions provoquent un redémarrage de la machine lorsque l'on appuie sur les touches `CTRL+ALT+SUPPR` simultanément dans un terminal virtuel.

5.3. Pages de manuel

Maintenant que vous savez l'essentiel pour conserver votre système en bon état, nous allons traiter des autres commandes Unix. Parmi elles, il en est qui sont certainement fondamentales : ce sont les commandes qui permettent d'obtenir de l'aide !

Chaque commande Unix a une page de manuel, qui la décrit. Ces pages sont très souvent écrites en nglais, mais elles sont très précises et fournissent toutes les informations dont on peut avoir besoin. Pour afficher la page de manuel d'une commande, il suffit d'utiliser la commande suivante :

```
man page
```

où `page` est la page de manuel de la commande sur laquelle on cherche des informations. En général, le nom de la page de manuel est le même que celui de la commande. Par exemple, pour afficher l'aide sur la commande **cp**, il suffit de taper :

```
man cp
```

Lorsqu'une page de man est affichée, il est possible de faire défiler son texte à l'aide des touches du curseur. Pour quitter l'aide, il suffit d'appuyer sur la touche `q`.

Les pages de man sont classées en groupes thématiques de pages, chaque groupe portant un numéro d'identification. Si la page de man affichée ne correspond pas à celle désirée, c'est qu'une page homonyme d'un autre groupe a été utilisée. Dans ce cas, il faut préciser le numéro du groupe de pages de manuel avant le nom de la page à afficher :

```
man numéro page
```

où `numéro` est le numéro du groupe auquel la page de manuel appartient. Les principaux groupes sont les suivants :

Tableau 5-1. Groupes de pages de man

Numéro de groupe	Type de pages de manuel
1	**Commandes utilisateur**
2	**Appels systèmes (programmation en C)**
3	**Fonctions des librairies C**

Numéro de groupe	Type de pages de manuel
4	**Description des fichiers spéciaux**
5	**Description des fichiers de configuration**
6	**Jeux et programmes divers**
7	**Programmes systèmes divers**
8	**Administration système**

Si vous ne savez pas dans quel groupe se trouve une page de manuel, vous pouvez utiliser l'option `-k`, qui permet d'afficher l'ensemble des pages disponibles portant ce nom :

```
man -k commande
```

Le numéro du groupe est en général précisé entre parenthèses, à la suite du nom de la page de manuel.

Il se peut également que vous recherchiez de l'aide sur un sujet donné, mais que vous ne connaissiez pas le nom exact de la page de manuel qui en parle. Pour ce genre de recherche, vous pourrez utiliser le programme **apropos**, qui recherchera toutes les pages de manuel qui contiennent un mot-clé particulier. Ce programme s'utilise avec la syntaxe suivante :

```
apropos mot-clé
```

où `mot-clé` est le mot-clé à rechercher dans toutes les pages de manuel.

La commande **man** est la commande d'aide standard sur tous les systèmes Unix. Cependant, Linux utilise un grand nombre de commandes écrites sous la licence GNU, et qui utilisent un format d'aide spécifique à GNU. L'aide pour ces commandes peut être obtenue par la commande suivante :

```
info commande
```

Il se peut que les deux méthodes fonctionnent. Dans ce cas, la page de man sera certainement moins récente que la page d'info, car la commande que vous utilisez est sans aucun doute une commande GNU, qui a été fournie avec sa page d'information. Il est donc recommandé de lire plutôt la page d'information GNU.

Le format d'aide GNU est plus riche que celui de man, puisqu'il permet de naviguer dans le système d'aide à l'aide de liens hypertextes. Ces liens sont organisés hiérarchiquement, avec des chapitres et des sous-chapitres. Chaque chapitre dispose d'une forme de table des matières constituée de menus, qui permettent d'accéder aux sous-chapitres. Les menus se distinguent du texte normal par une astérisque (« * ») en début de ligne dans la table des matières. Les commandes clavier suivantes pourront vous être utiles pour naviguer dans la hiérarchie du système d'aide de GNU :

- la touche de tabulation permet de passer au lien hypertexte suivant ;
- la touche `n` (pour « Next ») permet de passer au chapitre suivant ;
- la touche `p` (pour « Previous ») permet de revenir au chapitre précédent ;

- la touche u (pour « Up ») permet de remonter d'un niveau dans le système d'aide et d'atteindre la table des matières référençant le chapitre courant.

Enfin, la commande q permet de quitter le système d'aide.

5.4. Opération de base sur les répertoires

Ce paragraphe va vous décrire les opérations de base qu'il faut savoir faire pour manipuler les répertoires du système de fichiers.

La première commande est évidemment celle qui permet de lister le contenu d'un répertoire. Elle dispose d'un grand nombre d'options :

```
ls [options] [fichier]
```

où `fichier` est le nom d'un fichier ou d'un répertoire que l'on désire lister. Si ce paramètre est absent, **ls** affichera tous les fichiers du répertoire courant. Les principales options sont `-l`, qui permet d'afficher des informations étendues (notamment les propriétaires, les groupes, les droits, la taille et éventuellement les liens), et `-a`, qui permet d'afficher tous les fichiers, y compris les fichiers cachés (ceux dont le nom commence par un point).

La deuxième commande est celle qui permet de changer de répertoire courant. Sa syntaxe est très simple :

```
cd [chemin]
```

où `chemin` est un chemin de répertoire Unix valide. Ce chemin est constitué des noms des répertoires et sous-répertoires successifs, séparés par des barres obliques « / ». Si aucun chemin n'est spécifié, cette commande change le répertoire courant pour le répertoire personnel de l'utilisateur. Par exemple, pour aller dans le répertoire d'installation de XWindow, il faut taper la commande suivante :

```
cd /usr/local/X11
```

La notion de chemin sera détaillée dans le paragraphe suivant. « cd » est l'abréviation de l'anglais « Change Directory ».

La troisième commande permet de créer un répertoire :

```
mkdir chemin
```

où `chemin` est le chemin spécifiant le répertoire à créer. Si le chemin ne contient que le nom du répertoire à créer, celui-ci est créé dans le répertoire courant et devient donc un sous-répertoire. « mkdir » est l'abréviation de l'anglais « MaKe DIRectory »).

La commande pour supprimer un répertoire est la suivante :

```
rmdir chemin
```

Pour supprimer un répertoire, il faut qu'il soit vide (c'est à dire qu'il ne contienne ni fichier, ni répertoire). « rmdir » est l'abréviation de l'anglais « ReMove DIRectory ».

Enfin, voici une commande dont vous ne vous servirez normalement que très peu, voire pas du tout. Elle permet d'afficher le répertoire courant :

```
pwd
```

Cette commande n'est a priori pas très utile, car le shell affiche toujours le répertoire courant sur la plupart des distributions. « pwd » est l'abréviation de l'anglais « Print Working Directory ».

5.5. Notions de chemins sous Unix

Les chemins Unix permettent de qualifier complètement un répertoire ou un fichier sur le disque. Pour cela, ils utilisent les noms de ces répertoires et de ces fichiers, et ils les combinent pour indiquer comment atteindre la cible dans le système de fichiers. Classiquement, les chemins sont spécifiés par la séquence des répertoires dans lesquels il faut aller pour trouver cette cible. Cette séquence est donnée par la suite des noms des répertoires, séparés par un caractère spécial. Sous Unix, ce caractère est la barre oblique (« `/` »). Le répertoire racine n'a pas de nom, et peut donc être référencé par une barre oblique seule.

Les chemins peuvent être *absolus* (c'est à dire qu'ils peuvent partir du répertoire racine) ou *relatifs* (c'est à dire qu'il peuvent partir du répertoire courant). Si l'on utilise un chemin relatif, il faut savoir que le répertoire courant est désigné par un point (« `.` »), et que le répertoire parent du répertoire courant est désigné par deux points successifs (« `..` »). Ainsi, si l'on est dans le répertoire `/usr/local/bin`, on peut accéder au répertoire `/usr/local/X11/bin` avec les deux chemins suivants :

```
/usr/local/X11/bin
```

ou :

```
../X11/bin
```

Le premier chemin est absolu, parce qu'il part directement du répertoire racine. Le deuxième chemin est relatif, car il part du répertoire courant.

> **Note:** Il va de soi que les chemines relatifs ne sont valides, sauf coup de chance, que dans le répertoire dans lequel ils sont écrits, alors que les chemins absolus sont toujours valables. En revanche, si des répertoires sont déplacés ensembles, les chemins relatifs entre ces répertoires restent valides, mais les chemins absolus deviennent faux. Toutefois, ces considérations ne concernent pas un utilisateur de base.

La plupart des shells sont capables d'effectuer ce que l'on appelle la *complétion automatique* des noms. La complétion automatique permet de n'écrire qu'une partie des noms de fichiers ou de répertoires et de demander au shell de compléter ces noms. Ceci peut se faire de deux manières. La première solution, qui est aussi la plus simple, consiste à taper le début du nom, puis d'utiliser une touche spéciale qui permet de demander au shell de le compléter. Si vous utilisez le shell bash (bash est le shell de prédilection sur les systèmes Linux), cette touche est la touche de tabulation. Ainsi, si vous tapez :

```
cd /ho
```

et que vous appuyez sur la touche de tabulation, bash complétera cette ligne de la manière suivante :

```
cd /home
```

Pour cela, il regarde la liste des fichiers et des répertoires qui commencent par « `ho` » dans le répertoire racine. Normalement, il ne s'y trouve que le répertoire `/home`, et c'est ce nom que bash utilise. Il va de soi qu'il ne faut pas qu'il y ait ambiguïté sur un nom partiel. Par exemple, si vous tapez la commande suivante :

```
cd /usr/l
```

et que vous demandiez au shell de compléter le nom, il ne pourra pas choisir quel répertoire utiliser entre `/usr/lib/` et `/usr/local/`. Dans ce cas, il émettra un petit bip signalant l'erreur. En appuyant une fois de plus sur la touche de tabulation, bash affiche la liste des choix possibles et vous propose de compléter vous-même la ligne de commande ambiguë.

La deuxième solution est d'utiliser les caractères génériques du shell. Ces caractères permettent de désigner n'importe quel caractère, ou n'importe quelle séquence de caractères. Ils sont désignés respectivement par un point d'interrogation (« `?` ») et par une astérisque (« `*` »). Ainsi, si l'on tape la commande suivante :

```
cd /ho*
```

le shell ira directement dans le répertoire `/home/`, car le caractère générique « `*` » peut être remplacé par la séquence de caractères « `me` ». Il est également possible d'écrire :

```
cd /?ome
```

et dans ce cas le caractère générique « `?` » sera remplacé par « `h` ». Encore une fois, il ne faut pas qu'il y ait ambiguïté. Dans le cas contraire, le comportement varie selon le shell. En général, il essaie de résoudre l'ambiguïté au mieux en analysant la suite du chemin, et s'il ne peut pas, il affiche un message d'erreur.

Note: Ces caractères génériques sont interprétés directement par le shell et non par la commande qui les reçoit en paramètres. Tout nom de fichier contenant un caractère générique est remplacé par la liste des fichiers qui correspondent au motif donné. S'il n'existe qu'un seul fichier dans cette liste, il est possible d'utiliser les commandes comme cd, qui ne prennent qu'un seul paramètre. Mais il est possible d'utiliser les commandes acceptant plusieurs paramètres, même s'il y a plusieurs fichiers dans cette liste. Ainsi, la commande suivante :

```
ls *txt
```

permet de lister tous les fichiers dont le nom se termine par « txt ». Il ne peut évidement pas y avoir d'ambiguïté dans ce cas.

Si on doit passer un paramètre comprenant l'un des caractères génériques interprétés par le shell à une commande particulière, on devra préfixer les caractères génériques d'un caractère d'échappement pour signaler au shell qu'il ne doit pas l'interpréter. Ce caractère d'échappement est la barre oblique inverse (« \ »). Il est également possible de passer les paramètres entre guillements « " », car le shell n'interprète pas les caractères génériques dans les chaînes de caractères. Par exemple, pour créer un répertoire `*`, on utilisera la commande suivante :

```
mkdir \*
```

5.6. Opération de base sur les fichiers

Vous aurez sans doute à afficher le contenu d'un fichier. Pour cela, la commande la plus appropriée est certainement la commande **less** :

```
less fichier
```

Cette commande affiche le contenu du fichier et vous permet de le faire défiler avec les flèches du curseur. Lorsque vous désirez terminer la visualisation, il suffit de taper la touche `q` (pour « quitter » **less**). Pour information, le nom de la commande **less** provient d'un trait d'humour sur une commande Unix plus classique, la commande **more**. Cette commande effectue à peu près le même travail que **less**, mais elle n'affiche le texte que page par page. Pour passer à la page suivante, il faut appuyer sur la barre d'espace. Quant à l'origine du nom de la commande **more**, c'est qu'elle afficher le mot « more » au bas de l'écran pour indiquer qu'il y a encore du texte à visualiser, et qu'il faut appuyer sur la barre d'espace pour lire la suite.

La commande **less** permet également d'effectuer une recherche dans le fichier en cours d'édition. Pour cela, il suffit de taper une commande de recherche de less. Cette commande commence par une barre oblique, suivie du texte à chercher. Par exemple, pour rechercher la chaîne de caractère « `local` » dans un fichier en cours de visualisation avec **less**, il suffit de taper :

```
/local
```

Lorsque vous voudrez rechercher l'occurrence suivante du motif de recherche, vous pourrez appuyer sur la touche `n` (pour « Next » en anglais). Pour rechercher l'occurrence précédente, il suffit de taper la touche `N` (en majuscule, cette fois).

Il est encore plus probable que vous aurez à éditer un fichier. Cette opération peut se faire relativement facilement grâce à un éditeur simplifié, **vi**. Cet éditeur n'est pas franchement ce qui se fait de plus convivial, cependant, il existe sur toutes les plates-formes Unix d'une part, et il est suffisamment léger pour pouvoir fonctionner sur un système minimal. Il est donc recommandé de savoir se servir de **vi**, ne serait-ce que dans le cas où votre système ne serait pas complètement fonctionnel. En clair, quand tout va mal, on peut compter sur **vi** ! **vi** sera décrit plus loin dans la Section 5.8, car il dispose d'un grand nombre de commandes et il ne serait pas opportun de les décrire ici.

En général, la création d'un fichier se fait avec **vi**, bien que d'autres commandes puissent créer des fichiers. En revanche, pour supprimer un fichier, il n'existe qu'une seule commande :

```
rm chemin
```

où `chemin` est le chemin complet permettant d'accéder au fichier à supprimer. Il est possible de spécifier plusieurs fichiers à la commande **rm**, dans ce cas, ils seront tous supprimés. rm est également capable de supprimer tous les fichiers d'un répertoire, ainsi que ses sous-répertoires. Dans ce cas, elle détruit toute une branche de l'arborescence du système de fichiers. Pour cela, il suffit d'utiliser l'option `-r` (pour « récursif ») avant le chemin du répertoire à supprimer.

> **Attention !:** La commande **rm** ne demande aucune confirmation avant de supprimer les fichiers ! D'autre part, les fichiers supprimés sont irrémédiablement perdus (il n'y a pas de commande « undelete » ou autre commande similaire). Vérifiez donc bien ce que vous avez tapé avant de valider une commande **rm** (surtout si vous êtes sous le compte root).

La copie d'un fichier se fait avec la commande **cp**, dont la syntaxe est donnée ci-dessous :

```
cp fichiers répertoire
```

où `fichiers` est la liste des fichiers à copier, et `répertoire` est le répertoire destination dans lequel ces fichiers doivent être copiés.

Enfin, le déplacement des fichiers se fait avec la commande **mv**, comme indiqué ci-dessous :

```
mv source destination
```

où `source` est le nom du fichier source et `destination` est le nom du répertoire destination. Notez que **mv** est une commande très puissante, puisqu'elle permet également de déplacer des répertoires et de renommer des fichiers et des répertoires. Pour renommer un fichier ou un répertoire, il suffit d'indiquer le nouveau nom de ce fichier ou de ce répertoire à la place de `destination`.

5.7. Autres commandes utiles

Pour terminer ce petit cours d'Unix, nous allons décrire quelques-unes des autres commandes d'Unix parmi les plus utiles. Elles sont utilisées moins souvent que les commandes vues précédemment, mais vous apprendrez certainement très vite à vous en servir, car elles sont très pratiques.

5.7.1. Passage en mode superviseur

Si vous êtes prudents, vous avez sans doute créé un compte utilisateur juste après avoir installé votre système de base, et vous ne travaillez plus que sous ce compte. Cette technique est prudente, cependant, elle pose un problème évident : vous ne pouvez pas faire votre travail d'administrateur sous ce compte. C'est pour cela que la commande **su** a été créée. Cette commande permet de changer son identité dans le système :

```
su [utilisateur]
```

où `utilisateur` est l'utilisateur dont on veut prendre l'identité. Par défaut, si aucun utilisateur n'est spécifié, le changement d'identité se fait vers l'utilisateur root. Bien entendu, il va de soi que la commande su demande le mot de passe avant d'obtempérer…

5.7.2. Changement des droits des fichiers, du propriétaire et du groupe

La commande permettant de changer les droits d'un fichier ou d'un répertoire est la suivante :

```
chmod droits fichier
```

où `fichier` est le fichier ou le répertoire dont on désire changer les droits, et `droits` est une chaîne de caractères permettant de spécifier les nouveaux droits. Cette chaîne commence par une lettre indiquant le groupe d'utilisateur auquel le droit doit être appliqué, d'un caractère + ou - indiquant si le droit doit être ajouté ou supprimé, et d'une lettre indiquant le droit que l'on est en train de manipuler. La première lettre peut prendre les valeurs suivantes :

- `u` pour le champ « utilisateur », c'est à dire le propriétaire du fichier ;
- `g` pour le champ « groupe », c'est à dire tous les utilisateurs faisant partie du groupe du fichier ;
- `o` pour le champ « other », c'est à dire pour tous les utilisateurs qui ne sont ni propriétaire, ni membre du groupe du fichier ;
- `a` pour tous les champs sans distinction, donc pour tous les utilisateurs.

Les droits sont identifiés par l'un des caractères suivants :

- `r` pour le droit de lecture ;
- `w` pour le droit d'écriture ;
- `x` pour le droit d'exécution ;

- `s` pour les bits setuid et setguid ;
- `t` pour le bit sticky.

Ainsi, la commande suivante :

```
chmod g+w toto
```

permet de donner le droit d'écriture sur le fichier `toto` à tous les membres du groupe auquel ce fichier appartient.

Les droits d'accès ont déjà été décrit en détails ci-dessus dans le chapitre concernant les notions générales sur Unix.

Le changement de propriétaire d'un fichier ne peut être réalisé que par l'administrateur du système. Cette opération se fait à l'aide de la commande suivante :

```
chown utilisateur fichier
```

où `utilisateur` est le nom de l'utilisateur qui doit devenir propriétaire du fichier, et `fichier` est le fichier devant changer de propriétaire.

Le changement de groupe peut être réalisé par n'importe quel utilisateur, mais on ne peut donner un fichier qu'à l'un des groupes dont on est membre. Cette opération se fait à l'aide de la commande suivante :

```
chgrp groupe fichier
```

où `groupe` est le nom du groupe qui doit être affecté au fichier, et `fichier` est le fichier devant changer de groupe. Bien entendu, l'administrateur peut affecter un fichier à n'importe quel groupe d'utilisateur.

5.7.3. Gestion des liens

La commande pour créer un lien est **ln**. Cette commande utilise la syntaxe suivante :

```
ln [-s] source lien
```

où `source` est le nom du fichier ou du répertoire source auquel le lien doit se référer, et `lien` est le nom du lien. L'option `-s` permet de créer un lien symbolique, par défaut, ce sont des liens physiques qui sont créés. Rappelons qu'il est impossible de créer des liens physiques sur des répertoires.

Lorsque l'on liste des fichiers, on peut demander l'affichage d'informations complémentaires sur les liens. Pour cela, il suffit d'utiliser l'option `-l` de la commande **ls**. Ainsi, la commande suivante :

```
ls -l lien
```

permet d'afficher les informations sur le lien `lien`, et en particulier le fichier ou le répertoire cible de ce lien.

La suppression des liens se fait exactement comme celle d'un fichier. La destination n'est pas affectée en générale, sauf si le lien est un lien physique et constitue la dernière référence au fichier pointé par le lien.

Les liens symboliques n'ont pas de droits d'accès ni de propriétaires, les informations de sécurité de la cible sont utilisées lorsque l'on accède au lien.

5.7.4. Montage et démontage d'un système de fichiers

Comme il l'a été vu dans le chapitre expliquant les généralités sur Unix, les systèmes de fichiers ne sont pas accessibles directement. Ils doivent subir une opération que l'on nomme le *montage*. De la même manière, il faut penser à *démonter* les systèmes de fichiers avant d'éteindre la machine ou de retirer le support dans le cas des supports amovibles, faute de quoi les données non écrites seront définitivement perdues.

Heureusement, il y a la possibilité d'enregistrer les systèmes de fichiers les plus utilisés pour que le système les monte et les démonte automatiquement au démarrage et à l'arrêt du système. Cependant, cet automatisme n'est disponible que pour les systèmes de fichiers fixes, et il est nécessaire de monter soi-même les systèmes de fichiers sur disques amovibles (comme, par exemple, les disquettes et les CD-ROM).

L'opération permettant de monter un disque suit la syntaxe suivante :

```
mount [-t type] fichier base
```

où `fichier` est le fichier contenant le système de fichiers à monter (en général, il s'agit d'un fichier spécial de périphérique, mais ce peut également être une image disque), et `base` est le *point de montage*, c'est à dire le répertoire à partir duquel le système de fichiers doit être accédé. L'option `-t` permet d'indiquer le type du système de fichiers, mais en général, il n'est pas nécessaire de le préciser. En effet, le noyau sait reconnaître la plupart des systèmes de fichiers automatiquement. Pour information, les systèmes de fichiers les plus utilisés sont les suivants :

- `ext2` pour les systèmes de fichiers EXT2FS ;
- `iso9660` pour les CD-ROM (qu'ils soient avec extensions Juliette ou Rock Ridge ou en mode ISO 9660 pur) ;
- `msdos` pour les systèmes de fichiers FAT normaux ;
- `vfat` pour les systèmes de fichiers FAT32 ;
- `umsdos` pour les systèmes de fichiers UMSDOS (extension au DOS pour permettre les fonctionnalités des systèmes de fichiers Unix).

Si le répertoire de montage n'est pas vide, les fichiers qui s'y trouvent sont masqués par le système de fichiers monté. Il est donc recommandé de ne monter les systèmes de fichiers que dans des répertoires vides.

La commande permettant de démonter un système de fichiers est beaucoup plus simple :

```
umount fichier | base
```

où `fichier` est le fichier contenant le système de fichiers à démonter, et `base` est le répertoire dans lequel ce système de fichiers est monté. On peut utiliser l'un ou l'autre de ces paramètres, la commande **umount** se débrouillera pour retrouver l'autre automatiquement. On notera qu'il est impossible de démonter un système de fichiers qui est en cours d'utilisation par quelqu'un. En particulier, il ne faut pas être dans le répertoire servant de point de montage pour pouvoir démonter un système de fichiers, car dans ce cas on est en train de l'utiliser.

La commande **mount** peut prendre diverses options pour le montage des systèmes de fichiers. Par exemple, elle permet de monter des systèmes de fichiers en lecture seule, ou de monter des systèmes de fichiers placés dans des images disques. Il est possible d'enregistrer des options par défaut dans le fichier de configuration `/etc/fstab`. Ce fichier contient, entre autre, le répertoire de montage, le type du système de fichiers et le fichier de périphérique à utiliser pour ce système de fichiers. De cette manière, il est possible d'utiliser la commande **mount** de manière simplifiée, en ne précisant que le répertoire servant de point de montage ou le fichier spécial de périphérique. Le fichier `/etc/fstab` sera décrit plus en détail plus loin.

5.7.5. Recherche d'un texte dans un fichier

La recherche d'une chaîne de caractères dans un ou plusieurs fichiers peut se faire à l'aide de la commande **grep**. Cette commande prend en premier paramètre le texte à rechercher, puis la liste des fichiers dans lequel ce texte doit être trouvé :

```
grep texte fichiers
```

Le texte peut être placé entre guillemets si nécessaire (en particulier, s'il contient des espaces ou des caractères interprétés par le shell, comme `*` et `?`). **grep** accepte un grand nombre d'options, qui ne seront pas décrites ici. Consulter les pages de manuel pour plus d'information à ce sujet.

5.7.6. Recherche de fichiers

Il vous sera sans doute nécessaire de rechercher des fichiers selon un critère donné dans toute une arborescence de répertoires. Pour cela, vous utiliserez la commande **find**. Cette commande est très puissante, mais dispose d'une syntaxe assez compliquée :

```
find répertoire -name nom -print
```

où `répertoire` est le répertoire à partir duquel la recherche doit commencer et `nom` est le nom du fichier à rechercher. Ce nom peut contenir des caractères génériques du shell, mais dans ce cas doit être placé entre guillemets afin d'éviter que ce dernier ne les interprète.

find accepte d'autre options de recherche que le nom (partie « `-name` » de la ligne de commande), et peut effectuer d'autres actions que l'affichage du chemin des fichier trouvés (partie « `-print` »). Consultez les pages de manuel pour plus d'information à ce sujet.

5.7.7. Compression et décompression des fichiers

Linux fournit un grand nombre de programme de compressions de fichiers. Le meilleur est sans doute **bzip2**, et le plus compatible sans doute **compress**. Cependant, le plus utilisé et le plus courant, surtout pour la distribution des sources, reste incontestablement **gzip**. Nous allons donc décrire brièvement comment comprimer et décomprimer des fichiers avec **gzip** dans ce paragraphe.

La compression d'un fichier se fait de manière élémentaire :

```
gzip fichier
```

où `fichier` est le fichier à comprimer. Après avoir effectué son travail, **gzip** renomme le fichier comprimé en « `fichier.gz` ».

La décompression d'un fichier gzippé se fait à l'aide de la commande suivante :

```
gunzip fichier.gz
```

où `fichier.gz` est le nom du fichier à décomprimer. Après décompression, l'extension .gz est supprimée du nom de fichier.

5.7.8. Archivage de fichiers

L'archivage de fichiers se fait classiquement sous Unix avec le programme **tar** (abréviation de l'anglais « Tape ARchiver »). Ce programme permet simplement de regrouper tous les fichiers qu'il doit archiver dans un seul fichier structuré en blocs. Il a été évidemment écrit pour permettre des archivages sur bandes ou sur tout autre périphérique de stockage de masse, mais il est également utilisé pour créer des fichiers archives contenant toute une arborescence.

La syntaxe de **tar** est très simple :

```
tar options archive [fichiers]
```

où `options` sont les options qui indiquent l'opération à effectuer et comment elle doit être réalisée, `archive` est le nom de l'archive qui doit être créée ou le nom du fichier de périphérique du périphérique d'archivage, et `fichiers` est la liste des fichiers à archiver. Cette liste ne doit pas être précisée s'il s'agit d'une restauration.

Les options de **tar** que vous utiliserez le plus souvent sont les suivantes :

- `cvf` pour créer une archive ;
- `tvf` pour lister le contenu d'une archive ;

- `xvf` pour restaurer le contenu d'une archive.

Par exemple, pour archiver le contenu du répertoire courant dans le fichier `archive.tar`, vous utiliserez la ligne de commande suivante :

```
tar cvf archive.tar *
```

De plus, pour extraire le contenu de l'archive `archive.tar`, vous utiliserez la commande suivante :

```
tar xvf archive.tar
```

> **Note:** L'option `z` permet d'effectuer une compression des données archivées ou une décompression des données restaurées à la volée. **tar** utilise **gzip** et **gunzip** pour la compression et la décompression.
>
> Si l'on utilise un signe négatif ('`-`') à la place du nom de l'archive, **tar** enverra le résultat de la compression vers la sortie standard. Ceci peut être utilisé pour des opérations avancées. Un exemple sera donné dans la Section 5.9.2.

5.7.9. Gestion des paquetages

La plupart des distributions actuelles utilisent le format de fichier « rpm » (« Redhat Package Manager ») pour leurs distributions. Ce format de fichier a été introduit par la distribution Redhat, mais a été licencié sous la licence GNU, ce qui a permis aux autres distributions de l'utiliser. Ces fichiers encapsulent tous les fichiers des paquetages, ainsi que des informations permettant de gérer les dépendances entre les paquetages, leurs versions et la manière de les installer dans le système et de les y supprimer.

Les fichiers rpm peuvent être manipulés à l'aide du programme **rpm**. Il est probable que le programme d'installation de votre distribution vous évite d'avoir à manipuler cet outil vous-même. Cependant, les principales commandes de **rpm** seront décrites ici, afin que vous puissiez l'utiliser en cas de besoin.

Le programme **rpm** utilise une syntaxe très classique :

```
rpm options [paquetage]
```

Les options indiquent les opérations à effectuer. La première option est bien entendu l'option `-i`, qui permet l'installation d'un paquetage :

```
rpm -i paquetage
```

La mise à jour d'un paquetage déjà installé se fait à l'aide de l'option `-U` :

```
rpm -U paquetage
```

La suppression d'un paquetage se fait à l'aide de l'option `-e` :

```
rpm -e paquetage
```

La commande permettant d'obtenir les informations (auteur, description, version) sur un paquetage contenu dans un fichier rpm est la suivante :

```
rpm -qi -p paquetage
```

Enfin, la commande pour lister tous les fichiers d'un paquetage contenu dans un fichier rpm est la suivante :

```
rpm -ql -p paquetage
```

Cette commande affiche les chemins complets, ce qui permet de savoir dans quel répertoire chaque fichier sera installé.

Il existe beaucoup d'autres option disponibles. Cependant, leur description dépasserait le cadre de ce document. Vous pouvez toujours consulter la page de manuel `rpm` si vous désirez plus d'informations.

5.8. vi, l'éditeur de fichiers de base

Vous serez obligé, lorsque vous effectuerez la configuration de votre système, d'éditer les fichiers de configuration (classiquement, ces fichiers sont placés dans le répertoire `/etc/`). Ces modifications peuvent être réalisées avec n'importe quel éditeur a priori, et il est même conseillé d'utiliser votre éditeur favori. Cependant, il faut savoir se servir de vi, parce que c'est le seul éditeur qui sera toujours installé, et qui fonctionnera en toutes circonstances. Le prix à payer pour cette fiabilité est un nombre restreint de fonctionnalités. En fait, vi est très puissant, mais il ne s'embarrasse pas de superflu, ce qui en fait certainement l'éditeur le moins convivial du monde. Ce paragraphe vous donnera la description des principales commandes de vi, et ne sera pas exhaustive car vous ne l'utiliserez certainement pas dans la vie courante.

Pour éditer un fichier avec vi, il suffit de passer le nom de ce fichier en ligne de commande :

```
vi fichier
```

Il est possible de passer plusieurs fichiers dans la ligne de commande, et vi les éditera les uns après les autres. Cependant, il faut savoir que vi ne permet de travailler que sur deux fichiers à la fois, et qu'il n'est pas facile de passer de l'un à l'autre. Par conséquent, il est conseillé de n'éditer qu'un seul fichier à la fois.

vi est un éditeur qui fonctionne avec plusieurs modes de fonctionnement : le *mode d'édition*, dans lequel le texte peut être modifié, le *mode de commande*, dans lequel des commandes particulières peuvent être données, et le *mode de visualisation*, dans lequel le fichier ne peut être que visualisé. Par défaut, vi est en mode de visualisation, et il faut utiliser une commande d'édition pour passer en mode

d'édition. Quand on est en mode d'édition, on peut revenir au mode de visualisation en appuyant sur la touche `Echap` (ou `Esc`, selon votre clavier). Cette touche a aussi une signification dans le mode de commande : elle annule la saisie de la commande en cours. Par conséquent, lorsqu'on est perdu et que l'on ne sait plus dans quel mode on se trouve (ce qui arrive fatalement à un moment donné), il suffit d'appuyer sur cette touche. On sait alors qu'on se trouve en mode de visualisation.

Le déplacement du curseur en mode de visualisation se fait avec les touches du curseur. Cependant, si votre clavier n'est pas bien configuré, ces touches peuvent ne pas fonctionner. C'est pour cette raison que vi fournit un jeu de touches alternatif :

- la touche `h` permet de déplacer le curseur vers la gauche ;
- la touche `l` permet de déplacer le curseur vers la droite ;
- la touche `j` permet de déplacer le curseur vers le bas ;
- la touche `k` permet de déplacer le curseur vers le haut.

Le curseur est bien entendu déplacé automatiquement lors de la saisie du texte en mode d'édition.

Le passage en mode d'édition peut se faire avec l'une des commandes suivantes :

- la touche `i` permet de passer en mode d'insertion (le texte saisi s'insère avant le caractère sur lequel le curseur est positionné) ;
- la touche `a` permet de passer en mode d'ajout de caractères (le texte saisi s'insère après le caractère sur lequel le curseur est positionné) ;
- la touche `A` permet de placer le curseur en fin de ligne et de passer en mode d'ajout de caractères ;
- la touche `o` permet de créer une nouvelle ligne après la ligne où se trouve le curseur et de passer en mode d'édition sur cette nouvelle ligne ;
- la touche `O` permet de créer une nouvelle ligne avant la ligne où se trouve le curseur et de passer en mode d'édition sur cette nouvelle ligne.

La création d'une nouvelle ligne peut donc être faite avec les commandes `o` et `O`, mais il est possible de couper une ligne en deux, ou de passer à la ligne simplement en tapant sur la touche `Entrée` en mode d'édition. Inversement, la commande `J` permet de supprimer un saut de ligne en fin de ligne et de placer ainsi le texte de la ligne suivante à la suite du texte de la ligne courante.

La suppression d'un caractère se fait avec la touche `Suppr` (ou `Del`, selon le clavier) ou la touche de retour arrière (dite touche `Backspace`). Cependant, encore une fois, vi fournit un jeu de touches alternatif permettant de travailler avec un clavier mal configuré :

- la commande `x` permet d'effacer le caractère situé sous le curseur ;
- la commande `dd` permet d'effacer la ligne où se trouve le curseur ;
- la commande `dw` permet d'effacer le mot où se trouve le curseur.

Le texte qui a été supprimé est placé dans ce que l'on appelle un buffer. Le contenu du buffer peut être inséré à n'importe quel endroit du fichier grâce à la commande `p`. Ainsi, il est possible de faire un couper/coller en effaçant la ligne désirée et en appuyant sur la touche `p` à l'emplacement destination.

La commande `u` permet d'annuler la dernière opération effectuée, et la commande `U` permet de la ré-exécuter.

La commande `yy` permet de copier la ligne courante dans le buffer. Cette commande est donc utilisée pour effectuer des copier/coller, en combinaison avec la commande `p`.

Les commandes de vi peuvent être répétées un certain nombre de fois, en tapant un nombre avant de les taper. Ainsi, pour supprimer 3 lignes, il suffira de taper la commande suivante :

```
3dd
```

Dans ce cas, ces trois lignes sont également placées dans le buffer. La même technique peut être utilisée pour copier/coller plusieurs lignes en une seule opération.

Enfin, vi accepte un certain nombre de commandes générales lorsqu'il est en mode de commande. Ce mode est activé dès que l'on appuie sur la touche deux points (':') dans le mode de visualisation. Les commandes générales les plus utiles sont décrites ci-dessous :

- la commande `:q` permet de quitter vi. Si le fichier en cours d'édition a été modifié, vi refusera de se terminer sans l'enregistrer. Si l'on veut malgré tout sortir sans l'enregistrer, il faudra utiliser la commande `:q!` ;
- la commande `:w` permet d'enregistrer le fichier courant. Pour enregistrer ce fichier et quitter vi, la commande `:wq` peut être utilisée ;
- la commande `:help sujet` permet d'obtenir de l'aide sur le sujet « `sujet` » ;
- la commande `:!commande` permet d'exécuter la commande du shell « `commande` ». Ceci peut être pratique pour effectuer une opération dans le shell sans avoir à quitter vi. Ceci dit, il sera sans doute plus efficace d'utiliser un autre terminal virtuel.

Comme vous l'avez constaté, vi est réellement une horreur à utiliser. Malgré tout, il permet de faire tout ce dont on a besoin pour éditer un fichier. Il dispose même de puissantes fonctionnalités que même les traitements de texte évolués ne sont pas capables de faire. Elles ne seront cependant pas décrites ici, car ceci dépasserait le cadre de la simple installation de Linux. Vous pourrez toujours consulter la page de manuel de vi pour de plus amples informations.

5.9. Utilisation du shell bash

Le shell est l'environnement utilisateur en mode texte sous Linux. C'est le programme qui se charge de lire et d'exécuter les commandes que l'utilisateur saisit. Classiquement, le shell est utilisé de manière interactive, c'est à dire que l'utilisateur dialogue avec le système par l'intermédiaire du shell. Il saisit les commandes, et le shell les exécute et affiche les résultats. Le shell le plus couramment utilisé sous

Linux est sans aucun doute bash. En tout cas, c'est le shell par défaut que la plupart des distributions utilisent. Il est donc conseillé de connaître un petit peu ce que ce shell est capable de réaliser, et comment. Le shell bash une évolution du shell sh, utilisé par quasiment tous les systèmes Unix. Son nom provient de l'abréviation de l'anglais « Bourne Again SHell », ce qui signifie qu'il s'agit effectivement d'une nouvelle variante du shell sh.

Au temps des interfaces graphiques complexes et sophistiquées, il peut paraître archaïque de vouloir encore utiliser des lignes de commandes pour utiliser un ordinateur. C'est en partie vrai, mais il faut savoir que les shells Unix sont extrêmement puissants et que les interfaces graphiques ne permettent toujours pas, même à l'heure actuelle, de réaliser toutes les tâches faisables avec un shell. D'autre part, il est souvent plus efficace de taper une simple commande dans un shell que de rechercher un outil graphique et de parcourir les divers menus, puis de choisir les options de la commande désirée avant de valider. Des ergonomes ont démontré, et des graphistes du monde entier le confirmeront, que la souris n'est pas le périphérique d'entrée le plus précis et le plus facile à utiliser pour manipuler les objets de l'environnement utilisateur. La plupart des programmeurs utilisent encore bon nombre de ce qu'on appelle des « raccourcis claviers » pour exécuter des commandes, même dans les environnements utilisateurs graphiques par excellence.

Quoi qu'il en soit, le shell est bien plus qu'un interpréteur de commande. Il s'agit réellement d'un environnement de programmation, permettant de définir des variables, des fonctions, des instructions complexes et des programmes complets, que l'on appelle des *scripts shell*. Les sections suivantes ont pour objectif de vous montrer les principales caractéristiques du shell, sans pour autant prétendre vous apprendre la programmation des scripts shell. La lecture de cette section pourra donc être différée dans un premier temps. Toutefois, elle pourra être bénéfique à ceux qui désirent comprendre les scripts de configuration utilisés par leur distribution, ou tout simplement à ceux qui sont curieux de nature.

5.9.1. Contrôle des processus

Un des avantages des lignes de commandes par rapport aux environnements graphiques est la facilité avec laquelle elles permettent de contrôler les processus. Ce paragraphe décrit les principales méthodes pour lancer et arrêter un processus, ainsi que pour lui fournir les données sur lesquelles il doit travailler et récupérer ses résultats.

5.9.1.1. Lancement d'un programme en arrière plan

Le lancement normal d'un programme se fait en tapant sa ligne de commande et en appuyant sur la touche de validation. Le shell ne rendra pas la main et ne permettra pas de lancer un autre programme tant que le processus en cours ne sera pas terminé. Cependant, vous pouvez fort bien désirez lancer en arrière plan une commande dont la durée d'exécution peut être très longue et continuer à travailler. Après tout, Linux est multitâche... Eh bien, rien de plus facile !

Pour lancer une commande en arrière plan, il faut :

- s'assurer que la commande aura toutes les informations nécessaires pour travailler sans intervention de l'utilisateur (ou, autrement dit, que la commande ne soit pas interactive) ;

- ajouter une espèrluette (caractère « `&` ») à la fin de la ligne de commande commande.

Par exemple, la commande suivante :

```
cp /cdrom/kernel/linux-2.2.10.tar.gz . &
```

copiera l'archive du noyau 2.2.10 du CD-ROM vers le répertoire courant, et s'exécutera en arrière plan.

Lorsqu'une commande est lancée en arrière plan, le shell affiche deux nombres qui permettront de l'identifier par la suite. Le premier nombre, indiqué entre crochets, est le numéro de « job » du shell. Ce numéro sert à identifier les commandes du shell de manière unique. Un job est donc en réalité une commande du shell, simple ou complexe. Le deuxième numéro est le numéro de processus (« PID », pour « Process IDentifier ») dans le système du processus maître du job. Le PID est un numéro unique dans le système, qui permet d'identifier de manière unique les processus en cours. Ces deux nombres permettront de manipuler les processus, avec les commandes que l'on verra plus tard.

Il ne faut pas confondre les numéros de job avec les numéros de processus. Premièrement, un numéro de job n'est unique que dans un shell donné, et n'a aucune signification au niveau du système complet, alors que le numéro de processus est attribué par le système à chaque programme en cours d'exécution. Ensuite, une même commande du shell peut lancer plusieurs processus conjointement. Dans ce cas, il y a bien évidemment plusieurs numéros de processus, mais un seul et unique job. Ce genre de situation se produit par exemple lors de l'utilisation d'une redirection du flux de sortie standard d'un processus vers le flux d'entrée standard d'un autre processus.

Le numéro de processus affiché par le shell lors du lancement d'une ligne de commande complexe représente le PID du processus maître de la commande, c'est à dire, en pratique, le dernier processus d'une série de redirections ou le processus du shell exécutant les commandes complexes. Ceci signifie que dans tous les cas de configuration, ce PID est celui du processus qui contrôle l'ensemble des opérations effectuées par la ligne de commande. C'est donc par ce processus que l'on peut manipuler la commande complète, par exemple pour l'interrompre.

Il est possible de retrouver le PID du processus maître d'une commande à partir du numéro de job correspondant du shell. Ceci se fait simplement, en utilisant l'expressions suivante :

```
%job
```

où `job` est le numéro du job dont on cherche le PID. Ainsi, dans toutes les commandes décrites ci-dessous, le PID peut être utilisé directement, ou être remplacé par le numéro du job préfixé du caractère de pourcentage.

5.9.1.2. Listing des processus

Il n'est pas nécessaire de retenir tous les numéros de jobs et tous les PID des processus en cours d'exécution. Il existe en effet des commandes permettant d'obtenir la liste des processus et des jobs. La plus simple à utiliser est bien évidemment la commande du shell pour obtenir la liste des jobs avec leurs lignes de commandes. Pour obtenir cette liste, il suffit de taper la commande suivante :

```
jobs
```

qui affiche, dans l'ordre, le numéro de job, l'état du processus correspondant, et la ligne de commande.

Une autre commande, plus bas niveau, permet d'obtenir des informations plus complète directement à partir du système. Il s'agit de la commande **ps**, dont la syntaxe est donnée ci-dessous :

```
ps [options]
```

Les options les plus utiles sont sans doute `x`, qui permet de demander l'affichage de toutes les commandes en cours d'exécution et non pas seulement les processus en cours d'exécution dans le shell où la commande **ps** est exécutée, et `a`, qui permet d'obtenir l'affichage de toutes les commandes, pour tous les utilisateurs connectés. Ces deux options peuvent être cumulées, et la commande suivante :

```
ps ax
```

affiche donc toutes les commandes en cours d'exécution sur le système.

Les informations les plus intéressantes affichées par **ps** sont le PID du processus, qui est donné par le premier nombre affiché, et la ligne de commande, qui est la dernière information affichée. Pour plus de détails sur la commande **ps**, veuillez consulter la page de manuel correspondante.

5.9.1.3. Notion de signal

Dans un système Unix, tous les processus peuvent recevoir des messages, envoyés soit par l'utilisateur, soit par un autre processus, soit par le système. Ces messages sont appelés *signaux*. La plupart des signaux sont envoyés par le système pour indiquer au processus qu'il a fait une faute et qu'il va être terminé. Cependant, ce n'est pas toujours le cas : certains signaux sont envoyés uniquement dans le cadre de la communication entre les processus, et certains autres ne peuvent même pas être captés par le processus et sont traités directement par le système. Nous n'entrerons pas en détail dans la gestion des signaux ici, car cela nous emmènerait trop loin. Cependant, la manière d'envoyer un signal à un processus à partir du shell sera décrite.

L'envoi d'un signal se fait avec la commande **kill**, avec la syntaxe suivante :

```
kill [-signal] PID
```

où `signal` est une option qui permet de préciser le signal qui doit être envoyé, et `PID` est le numéro du processus qui doit le recevoir. Les numéros de signaux les plus importants sont décrits dans le tableau ci-dessous :

Tableau 5-2. Principaux signaux Unix

Numéro de signal	Signification
15	**Signal de terminaison de processus.**
9	**Signal de destruction inconditionnelle de processus.**

Numéro de signal	Signification
19	**Signal de suspension de processus.**
18	**Signal de reprise d'exécution d'un processus suspendu.**

Lorsqu'aucun signal n'est spécifié, le signal 15 de terminaison est utilisé par défaut. Ce signal demande au processus en cours d'exécution de se terminer immédiatement. Il peut être capté par le processus, pour lui donner une chance d'enregistrer les données sur lesquelles il travaillait et de libérer les ressources qu'il utilisait. Pour certains processus, ceci ne fonctionne pas, et il faut utiliser le signal de destruction du processus à l'aide de la commande suivante :

```
kill -9 PID
```

Attention cependant à cette commande : le processus est immédiatement détruit, sans autre forme de procès. Il peut donc s'ensuivre une perte de données, n'en abusez donc pas trop.

5.9.1.4. Arrêt d'un processus

Tout processus lancé en ligne de commande peut être arrêté immédiatement sous Linux. Pour cela, deux méthodes sont disponibles. La première consiste à taper la combinaison de touche `CTRL+C` lorsque le processus est en cours d'exécution interactive (c'est à dire lorsqu'il n'a pas été lancé en arrière plan). S'il a été lancé en arrière plan, on peut soit le ramener en avant plan (avec la commande **fg**, que l'on verra plus loin) avant d'utiliser `CTRL+C`, soit lui envoyer le signal de terminaison à l'aide de la commande **kill** vue précédemment. Dans le cas d'une ligne de commande, le signal de terminaison est transmis au processus maître de la ligne de commande, et est ensuite propagé à l'ensemble des processus fils de ce processus. Ceci signifie que tous les processus invoqués dans le cadre de cette commande sont également arrêtés.

5.9.1.5. Gel d'un processus

Il est possible de « geler » un processus en cours d'exécution, c'est à dire de le suspendre, sans pour autant l'arrêter définitivement. Ceci peut être utilisé pour libérer un peu les capacités de calcul, lorsque ce processus consomme trop de ressources par exemple. Pour cela, deux méthodes sont possibles :

- soit on utilise la combinaison de touches `CTRL+Z`, lorsque le processus est en avant plan ;
- soit on envoie le signal 19 au processus (signal « STOP ») à l'aide de la commande **kill**.

La première méthode est recommandée pour les processus lancés par une ligne de commande complexe, car le signal STOP est envoyé au processus maître de la commande, et est propagé à l'ensemble des processus fils de ce processus. Ceci signifie que tous les processus invoqués dans le cadre de cette commande sont également gelés. La deuxième méthode est plus bas niveau, et permet de geler n'importe quel processus que l'on a lancé.

5.9.1.6. Relancement d'un processus

Un processus suspendu peut être relancé soit en avant plan, soit en arrière plan. Pour relancer un processus en avant plan, il faut utiliser la commande suivante :

```
fg PID
```

où `PID` est le PID du processus à relancer en avant plan. **fg** est l'abréviation de l'anglais « foreground », ce qui signifie « avant plan ». Il faut attendre que ce processus se termine pour entrer de nouvelles commandes. Par conséquent, on ne peut lancer en avant plan qu'un seul processus.

De même, pour lancer un processus en arrière plan, il faut utiliser la commande **bg**, qui est l'abréviation de l'anglais « background ». Cette commande s'utilise de la même manière que la commande **fg** :

```
bg PID
```

Le relancement d'un processus suspendu peut également se faire en lui envoyant le signal 18 à l'aide de la commande **kill**.

5.9.2. Redirections

Pour pouvoir lancer un programme en arrière plan, il est nécessaire qu'il n'ait pas besoin de demander des données à l'utilisateur. En effet, lorsqu'il est en arrière plan, la saisie de ces données ne peut pas se faire, puisque le shell les interpréterait comme une nouvelle commande. De plus, tout affichage en provenance d'une commande en arrière plan apparaît sur la console tel quel, et risque de se mélanger avec l'affichage des autres programmes ou même avec la commande en cours d'édition. C'est pour résoudre ces problèmes que le mécanisme des redirections a été introduit.

5.9.2.1. Principe de base

Le mécanisme des *redirections* a pour but de transférer les données provenant d'un flux vers les données d'un autre flux. Il se base sur la notion de descripteur de fichier, utilisée par la plupart des systèmes Unix. Un *descripteur de fichier* est un numéro utilisé par les programmes pour identifier les fichiers ouverts. Les descripteurs 0, 1 et 2 sont respectivement affectés d'office au flux d'entrée standard (nommé « stdin »), au flux de sortie standard (« stdout ») et au flux d'erreur standard (« stderr »), qui en général apparaît également sur l'écran.

Les descripteurs de fichiers d'un processus sont généralement hérités par tous ses processus fils. Ceci signifie que lors de leur lancement, ces processus peuvent utiliser tous les descripteurs de fichiers mis à leur disposition par leur père. Dans le cas des lignes de commande, les processus fils sont les processus lancés par le shell, et les descripteurs de fichiers hérités sont donc les descripteurs de fichiers du shell. C'est de cette manière que le shell peut manipuler les descripteurs de fichiers des processus qu'il lance : il effectue d'abord les redirections sur ses propres descripteurs de fichiers, puis il lance

le processus fils avec ces redirections actives. Le mécanisme est donc complètement transparent pour les processus fils.

Le mécanisme des redirections permet en fait d'injecter dans un descripteur de fichier des données provenant d'un autre descripteur ou d'un fichier identifié par son nom, et d'envoyer les données provenant d'un descripteur de fichier dans un autre descripteur ou dans un fichier identifié par son nom. Si l'on utilise les descripteurs de fichiers des flux d'entrée / sortie standard, on peut exécuter n'importe quelle commande interactive en arrière plan.

5.9.2.2. Redirections de données en entrée

Pour injecter des données provenant d'un fichier dans le descripteur de fichier `n` d'un processus, il suffit d'ajouter la ligne suivante à la fin de la commande permettant de lancer ce processus :

```
n<fichier
```

où `fichier` est le nom du fichier dont les données doivent être injectées dans le descripteur `n`. Dans cette syntaxe, le descripteur peut ne pas être précisé. Dans ce cas, le shell utilisera le descripteur 0, et les données du fichier seront donc envoyées dans le flux d'entrée standard du processus. Par exemple, supposons que l'on désire utiliser une commande nommée « search », et que cette commande demande un certain nombre d'informations lors de son exécution. Si l'on sait à l'avance les réponses aux questions qui vont être posées, on peut créer un fichier de réponse (nommé par exemple « answer.txt » et alimenter la commande « search » avec ce fichier. Pour cela, on utilisera la ligne de commande suivante :

```
search < answer.txt
```

Il est également possible d'injecter des données provenant d'un autre descripteur de fichiers dans un descripteur de fichiers. On utilisera pour cela la syntaxe suivante :

```
n<&s
```

où `n` est toujours le descripteur de fichier du processus à exécuter dans lequel les données doivent être injectées, et `s` est un descripteur de fichiers contenant les données sources à injecter. Par défaut, si `n` n'est pas précisé, le flux d'entrée standard du processus sera utilisé.

5.9.2.3. Redirection de données en sortie

Inversement, il est possible d'enregistrer les données écrites par un processus dans un de ses descripteurs de fichier dans un fichier. Pour cela, on utilisera l'opérateur '>' avec la syntaxe suivante :

```
n>fichier
```

où `n` est le numéro du descripteur de fichier du processus à enregistrer, et `fichier` est le nom du fichier dans lequel les données doivent être stockées. Par défaut, si `n` n'est pas spécifié, le descrip-

teur du flux de sortie standard sera utilisé (descripteur 1). Par exemple, si la commande précédente affiche des résultats et que l'on désire les stocker dans le fichier « result.txt », on utilisera la ligne de commande suivante :

```
search < answer.txt >result.txt
```

Notez que cette commande détruira systématiquement le contenu du fichier « result.txt » et le remplacera par les informations provenant du flux de sortie standard du processus « search ». Il est possible de ne pas vider le fichier « result.txt », et d'ajouter les informations en fin de fichier, en utilisant l'opérateur '>>' à la place de l'opérateur '>'. Ainsi, la commande suivante :

```
search < answer.txt >>result.txt
```

aura pour effet d'ajouter à la fin du fichier « result.txt » les informations affichées par le processus « search ».

Le flux d'erreur standard, qui correspond normalement à l'écran et qui permet d'afficher les messages d'erreur, peut être redirigé avec l'opérateur '2>', de la même manière que l'opérateur '>' est utilisé pour le flux de sortie standard (puisque c'est le descripteur de fichier utilisé par défaut par l'opérateur '>'). Par exemple, si l'on veut envoyer les messages d'erreurs éventuels de la commande précédente vers le périphérique nul (c'est à dire le périphérique qui n'en fait rien) pour ignorer ces messages, on utilisera la ligne de commande suivante :

```
search <answer.txt >result.txt 2> /dev/null
```

Une telle ligne de commande est complètement autonome, et peut être lancée en arrière plan, sans aucune intervention de l'utilisateur :

```
search <answer.txt >result.txt 2> /dev/null &
```

Il est également possible d'effectuer une redirection des données provenant d'un descripteur de fichier du processus vers un autre descripteur de fichier de ce processus. On utilisera pour cela la syntaxe suivante :

```
n>&d
```

où `n` est le descripteur de fichier dont les données doivent être redirigées, et `d` le descripteur de fichier destination. Cette syntaxe est souvent utilisée pour rediriger le flux d'erreur standard vers le flux d'entrée standard, lorsque l'on veut récupérer les erreurs et les messages d'exécution normale dans un même fichier. Par exemple, si l'on veut rediriger le flux de sortie et le flux d'erreurs de la commande « search » dans un même fichier, on utilisera la ligne de commande suivante :

```
search <answer.txt >result.txt 2>&1
```

Cette ligne de commande utilise deux redirections successives pour les données affichées par la commande « search » : la première redirige le flux de sortie standard vers un fichier, et la deuxième le flux d'erreur standard vers le flux de sortie standard. Notez que l'ordre des redirections est important. Elles sont appliquées de gauche à droite. Ainsi, dans la commande précédente, le flux de sortie standard est redirigé vers le fichier « result.txt », puis le flux d'erreur standard est injecté dans le flux de sortie standard ainsi redirigé.

> **Note:** Il est également possible d'utiliser un autre descripteur de fichier que les descripteurs des flux standards. Cependant, il est nécessaire, dans ce cas, d'ouvrir ce descripteur dans le shell avant de lancer la commande. Ceci peut se faire à l'aide de la syntaxe suivante :
>
> `n<>fichier`
>
> où `n` est un numéro de descripteur de fichier non encore utilisé, et `fichier` est un nom de fichier. Ce nouveau descripteur de fichier pourra être utiisé dans les commandes précédentes, afin de faire manipuler le fichier `fichier` par les processus fils de manière transparente.
>
> Les descripteurs de fichiers ouverts de cette manière le restent d'une commande sur l'autre dans le shell. Ceci implique que toutes les données écrites dans ces descripteurs de fichiers sont ajoutées automatiquement à la fin des fichiers manipulés par ces descripteurs. Ce comportement est différent de la redirection vers un fichier effectuée par l'opérateur '>', qui ouvre à chaque fois le fichier en écriture à son début et qui supprime donc toutes les données déjà esistantes. Il n'y a donc pas d'opérateur '>>&' pour ajouter des données à un descripteur de fichiers, car cela n'a pas de sens.
>
> Les descripteurs de fichiers peuvent également être manipulés directement, par l'intermédiaire de fichiers virtuels du répertoire `/dev/fd/`. À chaque descripteur de fichier (y compris les descripteurs pour les flux d'entrée / sortie standard !) y correspond un fichier dont le nom est le numéro du descripteur. Par exemple, le fichier `/dev/fd/2` correspond au flux d'erreur standard.
>
> En fait, le répertoire `/dev/fd/` est un lien symbolique vers le répertoire `/proc/self/fd/` du système de fichiers virtuels `/proc/`. Ce système de fichiers est géré par le noyau directement, et permet d'accéder aux informations sur le système et les processus. Il contient en particulier un sous-répertoire pourtant le nom du PID de chaque processus existant dans le système, et chacun de ces répertoires contient lui-même un sous-répertoire `fd/` où sont représentés les descripteurs de fichiers ouvert par le processus correspondant. Le système de fichiers `/proc/` contient également un lien symbolique `self/` pointant sur le sous-répertoire du processus qui cherche à l'ouvrir. Ainsi, `/proc/self/fd/` est un chemin permettant à chaque processus d'accéder à ses propres descripteurs de fichiers.
>
> En pratique, la manipulation directe des descripteurs de fichiers n'est réellement intéressante que pour les flux standard, dont les numéros de descripteurs sont fixes et connus de tous les programmes. Pour les autres descripteurs, cette technique est souvent inutilisable ou inutile, sauf lorsqu'on utilise des programmes sachant manipuler des descripteurs de numéros bien déterminés.

5.9.2.4. Insertion de documents

Il existe un dernier opérateur de redirection, qui n'est utilisé en pratique que dans les scripts shell. Cet opérateur permet d'insérer directement un texte complet dans le flux d'entrée standard, sans avoir à placer ce document dans un fichier à part. Cette technique permet donc de stocker des données avec le code des scripts shell, et de n'avoir ainsi qu'un seul fichier contenant à la fois le script et ses données.

Cet opérateur est l'opérateur '<<', il s'utilise selon la syntaxe suivante :

```
<<EOF
texte
⋮
EOF
```

où `texte` est le contenu du texte à insérer, et `EOF` est un marqueur quelconque qui sera utilisé seul sur une ligne afin de signaler la fin du texte.

Par exemple, il est possible de créer un fichier `test.txt` de la manière suivante :

```
cat <<fin >test.txt
Ceci est un fichier texte saisi directement dans le shell.
On peut écrire tout ce que l'on veut, et utiliser les fonctions d'éditions de ligne du s
Pour terminer le fichier, il faut taper le mot "fin" tout seul, au début d'une ligne vid
fin
```

5.9.3. Les pipes

Les redirections sont très pratiques lorsqu'il s'agit d'injecter un fichier dans le flux d'entrée standard d'un processus, ou inversement de rediriger le flux standard d'une commande vers un fichier, mais elles ont justement le défaut de devoir utiliser des fichiers. Il est des situations où l'on désirerait injecter le résultat d'une commande dans le flux d'entrée standard d'une autre commande, sans passer par un fichier intermédiaire. Ceci est heureusement réalisable, grâce à ce que l'on appelle les « pipes ».

5.9.3.1. Syntaxe des pipes

Pour rediriger le résultat d'une commande dans le flux d'entrée d'une autre commande, il faut utiliser l'opérateur '|'. Cet opérateur représente un tuyau canalisant les données issues d'une commande vers le flux d'entrée standard de la commande suivante, d'où le nom de « pipe » (« tuyau » en anglais). L'opérateur pipe s'utilise de la manière suivante :

- on écrit la première commande, qui doit fournir les données à la deuxième commande ;
- on écrit l'opérateur pipe ;
- on écrit la deuxième commande, qui doit lire les données provenant de la première.

La commande se trouvant à la gauche de l'opérateur pipe doit être complète, avec ses autres redirections éventuelles. La redirection dans un pipe s'effectue après les autres types de redirections vues précédemment.

Le système contrôle l'exécution des processus qui se trouvent au deux bouts d'un pipe, de telle sorte que le transfert de données puisse toujours se faire. Si le processus source a trop de données, il est figé par le système d'exploitation en attendant que le processus consommateur ait fini de traiter les données déjà présente. Inversement, si le processus source est trop lent, c'est le processus consommateur qui attendra patiemment que les données soient disponibles.

Les pipes sont utilisés très couramment, ne serait-ce que pour afficher page par page le contenu d'un répertoire. La commande suivante effectue un tel travail :

```
ls | less
```

Ici, le résultat de la commande **ls** est redirigé vers la commande **less**, qui permet d'afficher page par page (et de revenir en arrière dans ces pages) la liste des fichiers du répertoire courant.

Prenons un exemple un peu plus complexe. Supposons que l'on veuille archiver et compresser un répertoire. Il est possible d'archiver ce répertoire avec la commande **tar**, puis de comprimer le fichier archive résultant :

```
tar cvf archive.tar *
gzip archive.tar
```

Cette méthode est correcte, mais souffre d'un défaut : elle utilise un fichier intermédiaire, qui peut prendre beaucoup de place disque. Une méthode plus économe consiste à lancer **tar** et **gzip** en parallèle, et de rediriger la sortie standard de l'un dans le flux d'entrée de l'autre. Ainsi, il n'y a plus de fichier temporaire, et la place consommée sur le disque est minimale :

```
tar cv * | gzip > archive.tar.gz
```

La première commande demande à **tar** d'archiver tous les fichiers du répertoire et d'envoyer le résultat dans le flux standard de sortie. Le pipe redirige ce flux standard vers le flux d'entrée standard de **gzip**. Celui-ci comprime les données et les émet vers son flux standard de sortie, qui est lui-même redirigé vers le fichier `archive.tar.gz`. Aucun fichier temporaire n'a été utilisé, et on a ainsi économisé l'espace disque de l'archive complète non compressée, c'est à dire environ la taille complète du répertoire à archiver. Ce genre de considération peut être très important lorsque le disque dur commence à être plein…

> **Note:** En fait, la commande tar de GNU permet de comprimer à la volée les données à archiver, permettant d'éviter de se prendre la tête comme on vient de le faire. Pour cela, il suffit d'utiliser l'option `z` dans la ligne de commande de **tar**. Ainsi, la ligne de commande suivante fournit le même résultat :
>
> ```
> tar cvfz archive.tar.gz *
> ```

> Mais cette solution ne fonctionne pas avec les versions non GNU de tar, qui ne supportent pas cette option.

Un autre exemple pratique est le déplacement de toute une arborescence de fichiers d'un système de fichiers à un autre. Vous ne pourrez pas y parvenir à l'aide de la commande **mv**, car celle-ci ne fait que modifier la structure du système de fichiers pour déplacer les fichiers et les répertoires, elle ne peut donc pas fonctionner avec deux systèmes de fichiers. Vous ne pouvez pas non plus utiliser la commande **cp**, car celle-ci ne prendra pas en compte les dates des fichiers, leurs propriétaires et leur groupes, ainsi que les liens symboliques et physiques. Il faut donc impérativement utiliser un programme d'archivage. La méthode à suivre est donc de créer une archive temporaire, puis de se déplacer dans le répertoire destination, et enfin d'extraire l'arborescence de l'archive :

```
cd source
tar cvf archive.tar *
cd destination
tar xvf source/archive.tar
rm source/archive.tar
```

Malheureusement, cette technique nécessite beaucoup de place disque, puisque l'archive temporaire est stockée directement sur disque. De plus, elle est assez lente, car toutes les données à copier sont recopiées sur le disque dur, et relues ensuite, pour finalement être détruites... La vraie solution est de réaliser un pipe entre les deux processus **tar** invoqués. Dans ce cas, le transfert se fait simplement via la mémoire vive :

```
cd source
tar cv * | (cd destination; tar xvf -)
```

La commande à utiliser est cette fois un peu plus compliquée, car la commande d'extraction des fichiers nécessite un changement de répertoire. Il faut donc utiliser une commande multiple du shell. Ces commandes sont constituées de plusieurs autres commandes séparées par des points virgules. La première commande effectuée ici est le changement de répertoire, et la deuxième est l'extraction par tar de l'archive qui lui est transférée par le flux d'entrée standard (représenté ici par '`-`'). Ces deux commandes sont mises entre parenthèses, car l'opérateur '`|`' du pipe est prioritaire sur l'opérateur '`;`' de concaténation des commandes du shell. Si vous trouvez que ceci est un peu compliqué, je vous l'accorde. Cependant, la commande qui utilise le pipe consomme deux fois moins d'espace disque et est deux fois plus rapide que la commande qui n'en utilise pas. Je vous invite à mesurer le gain de temps sur un répertoire contenant un grand nombre de données (utilisez la commande **time** !).

5.9.3.2. Les pipes nommés

Les pipes créés par l'opérateur '`|`' constituent ce que l'on appelle des pipes anonymes, car ils sont créés directement par le shell pour une commande donnée. Il est possible de créer manuellement des

pipes en leur donnant un nom, et de les utiliser a posteriori dans plusieurs commandes. Ces pipes constituent ce que l'on appelle des pipes nommés.

En fait, les pipes nommés sont des fichiers spéciaux, que l'on crée dans un système de fichier capable de les gérer. Ces fichiers se comportent différemment des fichiers classiques, en ce sens que les données que l'on ne peut pas se déplacer dans ces fichiers. Les seules opérations réalisables sont l'écriture et la lecture, sachant que les données écrites en premier seront forcément les premiers données lues. C'est ce comportement qui a donné leur nom à ces fichiers, que l'on appelle des « FIFO » (abréviation de l'anglais « First In First Out »). De plus, la quantité de données en transit dans ces fichiers est souvent très réduite, ce qui fait que ces données sont toujours placées dans la mémoire cache du système. Ainsi, bienqu'il s'agissent de fichiers, aucune écriture ou lecture sur disque n'a lieu lors de l'utilisation d'un pipe.

Les pipes nommés sont créés par la commande **mkfifo**, dont la syntaxe est la suivante :

```
mkfifo nom
```

où `nom` est le nom du pipe nommé. Notez que cette commande échouera sur les systèmes de fichiers incapables de gérer les pipes nommés.

Une fois créé, le fichier de pipe peut être utilisé comme n'importe quel fichier dans les redirections que l'on a vu dans la section précédente. Par exemple, la redirection suivante :

```
ls | less
```

peut être réécrite pour utiliser un pipe nommé temporaire de la manière suivante :

```
mkfifo /tmp/tempfifo
ls > /tmp/tempfifo
less < /tmp/tempfifo
```

La destruction d'un pipe nommé se fait comme n'importe quel fichier, à l'aide de la commande **rm**.

5.9.3.3. La commande tee

La commande **tee** est un petit programme permettant d'enregistrer les données qu'il reçoit dans son flux d'entrée standard dans un fichier et de les renvoyer simultanément vers sont flux de sortie standard. Elle est couramment utilisée, en conjonction avec les pipes, pour dupliquer un flux de données. Sa syntaxe est la suivante :

```
tee fichier
```

où `fichier` est le nom du fichier dans lequel le flux d'entrée standard doit être enregistré.

Supposons par exemple que l'on désire rediriger tous les messages (d'erreur ou non) de la commande **ls /proc/1/*** dans un fichier `result.txt`, tout en continuant à les visualiser sur l'écran. Pour cela, on utilisera la commande suivante :

```
ls /proc/1/* 2>&1 | tee result.txt
```

À l'issue de cette commande, le fichier `result.txt` contiendra une copie des données qui ont été émises par la commande **ls /proc/1/* 2>&1**.

5.9.3.4. La commande xargs

La commande **xargs** permet d'appeler une autre commande, en passant en paramètre les données qu'elle reçoit dans le flux d'entrée standard. Sa syntaxe est la suivante :

```
xargs commande
```

où `commande` est la commande que **xargs** doit exécuter. **xargs** construira une ligne de commande complète pour cette commande, en utilisant comme paramètres les données issues du flux d'entrée standard. Une fois cette ligne de commande construite, **xargs** l'exécutera. Par exemple, la commande suivante :

```
ls -l
```

peut être exécutée également de la manière suivante :

```
xargs ls
```

et en tapant la chaîne de caractère « `-l` » suivie du caractère de fin de fichier `CTRL+D`.

La commande **xargs** est une commande extrêmement utile lorsqu'elle est utilisée conjointement avec les pipes, parce qu'elle permet d'utiliser le résultat d'une commande en tant que paramètre pour une autre commande. Ce mécanisme est donc complémentaire de celui des pipes, puisque ceux-ci permettaient d'utiliser le résultat d'une commande pour alimenter le flux d'entrée standard d'une autre commande.

Un exemple plus utile que le précédent permettra de mieux comprendre comment on utilise la commande **xargs**. Supposons que l'on désire trouver tous les fichiers d'une arborescence complète dont l'extension est `.txt` et contenant la chaîne de caractères « test ». La liste des fichiers de l'arborescence peut être déterminée simplement à l'aide de la commande **find**, et la recherche du texte dans les fichiers se fait naturellement à l'aide de la commande **grep**. On utilisera **xargs** pour construire la ligne de commande pour **grep**, à partir du résultat fourni par la commande **find** :

```
find -name "*.txt" | grep -l "test"
```

Cette commande est plus simple et plus efficace que la commande équivalente :

```
find -name "*.txt" -exec grep -l "test" {} \;
```

parce que **grep** n'est exécuté qu'une seule fois (alors que l'option `-exec` de la commande **find** l'exécute pour chaque fichier trouvé).

5.9.4. Manipulation des variables d'environnement

Les systèmes Unix permettent de définir un environnement d'exécution pour chaque programme en cours d'exécution. L'environnement est un ensemble de paramètres, que l'on appelle les *variables d'environnement*, qui permettent de modifier le comportement du programme. Ces variables contiennent une valeur de type chaîne de caractères, dont la signification est propre à chaque variable. Il est d'usage que les noms des variables d'environnement soient écrits complètement en majuscules, mais ce n'est pas une obligation.

Chaque programme est susceptible de reconnaître un certain nombre de variables d'environnement qui lui sont propres, mais il existe également des variables standard que tous les programmes utilisent. C'est notamment le cas de la variable d'environnement PATH, qui contient la liste des répertoires dans lesquels le système doit rechercher les programmes à exécuter. Cette variable permet donc de lancer les programmes en tapant simplement leur nom, et de laisser le système rechercher le fichier de ce programme dans chacun des répertoires indiqués par la variable d'environnement PATH.

Par défaut, les programmes sont lancés avec l'environnement du programme qui les lance, c'est à dire dans la plupart des cas l'environnement d'exécution du shell. Les programmes peuvent également définir de nouvelles variables d'environnement, qui seront ainsi accessibles par les programmes qu'ils lanceront eux-mêmes.

Comme tout programme, le shell dispose d'un environnement, qu'il utilise pour stocker ses propres variables. En effet, comme nous l'avons déjà signalé plus haut, le shell est bien plus qu'un interpréteur de commande : il est complètement programmable. Et en tant qu'interpréteur d'un langage de programmation, il fournit la possibilité de définir des variables de ce langage. Les variables du shell sont donc également des variables d'environnement, mais le shell ne les communique pas par défaut aux programmes qu'il lance. Pour être plus précis, le shell utilise deux environnements différents :

- son propre environnement, qui contient les variables d'environnement normales et ses propres variables ;
- l'environnement d'exécution, qui est l'environnement que le shell utilise pour lancer les programmes.

Il est très facile de définir une variable du shell. Pour cela, il suffit de lui affecter une valeur, à l'aide de la syntaxe suivante :

```
variable=valeur
```

où `variable` est le nom de la variable à définir, et `valeur` est la valeur que l'on désire lui affecter. Notez qu'il n'est pas nécessaire de fournir une valeur. Dans ce cas, la variable ainsi définie sera vide.

Par exemple, la ligne suivante :

```
BONJOUR="Bonjour tout le monde \!"
```

permet de définir la variable BONJOUR. Notez que la valeur est encadrée entre guillemets, car elle contient des espaces. Notez également que le caractère point d'exclamation (' !') est précédé d'un caractère antislash d'échappement ('\'), car il a une signification particulière pour le shell. Ce caractère d'échappement permet simplement de signaler au shell qu'il ne doit pas interpréter le caractère qui le suit, et fait donc en sorte que le point d'exclamation fait partie de la chaînes de caractères à affecter à la variable. Bien entendu, le caractère antislash étant lui-même un caractère spécial pour le shell, et il doit lui-même être préfixé d'un autre antislash si l'on désire l'utiliser dans une chaîne de caractères.

La valeur d'une variable peut être récupérée simplement en préfixant le nom de la variable du symbole dollar ('$'). Ainsi, la commande suivante permet d'afficher le contenu de la variable BONJOUR :

```
echo $BONJOUR
```

Les variables ainsi définies ne font partie que de l'environnement du shell, elles ne sont donc pas accessibles aux programmes que le shell lance. Donc, si l'on relance un nouveau shell avec la commande suivante :

```
bash
```

et que l'on essaie de lire le contenu de la variable BONJOUR avec la commande **echo**, on obtient une chaîne vide. Ceci est normal, puisque le deuxième shell (c'est à dire celui qui est en cours d'exécution) n'utilise pas le même environnement que le premier shell. Vous pouvez quitter le nouveau shell avec la commande suivante :

```
exit
```

Dès lors, vous serez à nouveau dans le shell initial, et la variable BONJOUR sera à nouveau accessible.

Pour rendre une variable du shell accessible aux programmes que celui-ci peut lancer, il faut l'exporter dans l'environnement d'exécution. Ceci peut être réalisé avec la commande **export** :

```
export variable
```

où `variable` est le nom de la variable du shell à exporter dans l'environnement d'exécution.

La syntaxe précédente exporte de manière permanente les variables du shell. Mais il existe également une autre syntaxe, qui permet de ne définir des variables d'environnement que pour l'environnement d'exécution d'une seule commande. Cette syntaxe consiste simplement à préfixer la commande à exécuter par la définition de ladite variable. Par exemple, la commande suivante :

```
BONSOIR="Bonsoir tout le monde \!" bash
```

permet de lancer un shell et de lui communiquer la variable d'environnement BONSOIR. Cette variable ne sera définie que pour ce programme, si l'on quitte ce shell avec un **exit**, la variable BONSOIR ne sera plus définie.

Une variable peut être détruite à tout instant à l'aide de la commande **unset**. Cette commande prend en paramètre le nom de la variable à supprimer. Par exemple, la commande suivante supprime notre variable :

```
unset BONJOUR
```

Vous pouvez à tout moment visualiser l'ensemble des variables définies avec la commande **set**. Le tableau donné ci-dessous vous présentera les variables d'environnement les plus utilisées, et que la plupart des programmes utilisent pour permettre à l'utilisateur de modifier leur comportement :

Tableau 5-3. Variables d'environnements courantes

Nom	Signification
HOME	**Chemin du répertoire personnel de l'utilisateur.**
USER	**Nom de login de l'utilisateur. Cette information est également disponible au travers de la variable d'environnement LOGNAME.**
TERM	**Type de terminal utilisé. La valeur de cette variable sert aux applications pour déterminer les caractérisques du terminal et ses fonctionnalités, afin d'optimiser leur affichage. La valeur de cette variable est souvent `linux` sur les consoles Linux, et `xterm` dans les émulateurs de terminal graphiques sous X11. Nous verrons l'utilité de cette variable plus en détail dans la Section 6.10.7.**
SHELL	**Chemin sur le fichier de programme du shell actuellement utilisé. Sous Linux, il s'agit souvent du shell bash.**

Nom	Signification
PATH	**Liste des répertoires dans lesquels les programmes à exécuter seront recherchés. Cette liste ne doit pas contenir le répertoire courant (.), pour des raisons de sécurité de base (il suffit de placer un cheval de troie portant le nom d'une commande classique dans un répertoire pour que l'utilisateur le lance sans s'en rendre compte).**
LD_LIBRARY_PATH	**Liste des répertoires dans lesquels les librairies dynamiques seront recherchées si elles ne sont pas trouvables dans les répertoires classiques des librairies du système.**
C_INCLUDE_PATH	**Liste des répertoires dans lesquels le compilateur C recherchera les fichiers d'en-tête lors de la compilation des fichiers sources C. Cette liste doit contenir les répertoires additionnels, qui ne sont pas déjà pris en compte automatiquement par le compilateur C.**
CPLUS_INCLUDE_PATH	**Liste des répertoires dans lesquels le compilateur C++ recherchera les fichiers d'en-tête lors de la compilation des fichiers sources C/C++. Cette liste doit contenir les répertoires additionnels, qui ne sont pas déjà pris en compte automatiquement par le compilateur C.**

Nom	Signification
LIBRARY_PATH	**Liste des répertoires dans lesquels les librairies à utiliser lors de l'édition de lien des programmes doivent être recherchées. Cette variable n'est utilisée que par les outils de développement lors de la compilation de fichiers sources, et elle ne doit pas être confondue avec la variable d'environnement LD_LIBRARY_PATH, qui indique la liste des répertoires contenant les librairies dynamiques dans lequel l'éditeur de lien dynamique recherchera les librairies utilisées par les programmes lors de leur chargement. Les notions de fichiers sources et de compilation seront détaillées dans le Chapitre 8.**
TMPDIR	**Répertoire des fichiers temporaires. Par défaut, le répertoire des fichiers temporaires est le répertoire `/tmp/`, mais il est possible d'en changer grâce à cette variable d'environnement.**

Nom	Signification
TZ	**Définition de la zone horaire de l’utilisateur. Le système travaillant exclusivement en temps universel, chaque utilisateur peut définir sa propre zone horaire pour obtenir l’affichage des dates et des heures dans son temps local. Le format de cette variable d’environnement est assez complexe. Il est constitué de plusieurs champs séparés par des espaces, représentant successivement le nom du fuseau horaire (au moins trois caractères), le décalage à ajouter à l’heure universelle pour obtenir l’heure locale, le nom du fuseau horaire pour l’heure d’été, le décalage pour l’heure d’été, et les dates de début et de fin de l’heure d’été. Les décalages horaires doivent être exprimés avec un ’+’ pour les fuseaux horaires placés à l’ouest de Greenwich, ’-’ pour ceux situés à l’est. Les dates de début et de fin de la période d’heure d’été peuvent être exprimés de deux manières différentes. La première méthode est d’indiquer le numéro du jour dans l’année après la lettre ’J’. Ce numéro ne doit pas tenir compte du 29 février, même pour les années bissextiles. La deuxième méthode est d’indiquer le mois de l’année, la semaine du mois et le jour de la semaine, séparés par des ’.’, et après la lettre ’M’. Les mois sont comptés de 1 à 12, les semaines de 1 à 5 et les jours de 0 à 6, 0 étant le dimanche. Seul le premier champ est obligatoire, il est possible d’utiliser les noms de fuseau horaires définis par la librairie C. En France, on utilise normalement le fuseau CES (temps d’Europe centrale).**

Nom	Signification
LANG	**Valeur par défaut à utiliser pour les paramètres d'internationalisation des applications. Cette valeur sera utilisée pour les paramètres qui n'en définissent pas une explicitement. Elle doit être composée de deux codes à deux caractères, le premier indiquant la langue, et le deuxième le pays (car plusieurs pays peuvent parler la même langue, et un pays peut avoir plusieurs langues nationales). Pour la France, on utilise normalement la valeur « `fr_FR` ». Cette valeur peut être redéfinie par l'une des variables d'environnement décrites ci-dessous.**
LC_MESSAGES	**Valeur à utiliser pour déterminer la langue des messages. La valeur par défaut est spécifiée par la variable d'environnement LANG.**
LC_TYPE	**Valeur à utiliser pour déterminer les règles de classification des caractères. La classification des caractères permet de dire si un caractère est chiffre ou non, s'il est en majuscule ou en minuscule, etc… La valeur par défaut est spécifiée par la variable d'environnement LANG.**
LC_COLLATE	**Valeur à utiliser pour déterminer les règles de comparaison des caractères. La comparaison des caractères est utilisée pour les tris lexicographiques (tri par ordre alphabétique par exemple). La valeur par défaut est spécifiée par la variable d'environnement LANG.**
LC_MONETARY	**Valeur à utiliser pour déterminer l'emplacement et le caractère à utiliser pour le symbole monétaire du pays. La valeur par défaut est spécifiée par la variable d'environnement LANG.**

Nom	Signification
LC_NUMERIC	**Valeur à utiliser pour déterminer les conventions locales d'écriture des nombres (séparateurs décimal, format de la virgule, etc…). La valeur par défaut est spécifiée par la variable d'environnement LANG.**

En résumé, le shell utilise les variables d'environnement du système pour gérer ses propres variables, et permet de les exporter vers l'environnement d'exécution qu'il communique aux commandes qu'il lance. Un grand nombre de variables d'environnement classiques sont reconnues par les programmes. Elles servent à paramétrer leur comportement. Nous reverrons ultérieurement quelques unes de ces variables lors de la configuration du système de base.

5.9.5. Caractère d'échappement et chaînes de caractères

Un certain nombre de caractères sont interprétés par le shell d'une manière spéciale. Nous en avons déjà vu quelques uns pour les redirections et les pipes, mais il en existe d'autres. Par conséquent, il faut utiliser une syntaxe particulière lorsque l'on désire utiliser un de ces caractères dans une commande du shell sans qu'ils soient interprétés par le shell. Pour cela, il suffit de faire précéder ces caractères du caractère d'échappement antislash (caractère de la barre oblique inverse, \). Ce caractère permet d'indiquer au shell que le caractère suivant doit être traité tel quel, et ne doit pas être interprété avec son sens habituel. Par exemple, pour créer un répertoire nommé <, on utilisera la commande suivante :

```
mkdir \<
```

Bien entendu, le caractère antislash peut lui-même être précédé d'un autre antislash, lorsque l'on veut l'utiliser en tant que caractère normal.

Le caractère d'échappement antislash permet également, lorsqu'il est placé en fin de ligne, de supprimer le saut de ligne qui le suit. Ceci signifie qu'il permet de répartir une commande trop longue sur plusieurs lignes, à des fins de lisibilité. Vous trouverez quelques exemples de cette notation plus loin dans ce document, pour présenter des commandes trop longues pour tenir sur une page A4.

Il peut être relativement fastidieux de devoir taper des antislashes dans les chaînes de caractères qui contiennent beaucoup de caractères interprétables par le shell. C'est pour cela que le shell permet de définir des chaînes de caractères, dont il ignorera le contenu lors de l'analyse syntaxique. Ces chaînes de caractères sont simplement données entre guillemets simples (caractère `'`). Par exemple, la commande suivante :

```
MESSAGE='La syntaxe est A | B'
```

permet d'affecter la chaîne de caractère `La syntaxe est A | B`, contenant des espaces et le caractère `|` normalement utilisé par le shell pour les pipes, dans la variable d'environnement MESSAGE.

> **Note:** Une chaîne de caractères commence par un guillemet et se termine par un guillemet. Les chaînes de caractères ne peuvent donc pas contenir de guillemet, même précédé d'un caractère d'échappement.
>
> On veillera à ne surtout pas confondre les guillemets simples (caractère `'`) avec les guillemets inverses (caractère `‘`), qui se ressemblent énormément dans certaines polices de caractères. Ces deux caractères ont une signification complètement différente. Le premier sert à définir des chaînes de caractères, et le deuxième à exécuter une commande et à en inclure le résultat dans une autre commande. Nous verrons plus loin comment utiliser ce type de guillemets.

Les guillemets simples sont donc très pratiques pour écrire simplement une chaîne de caractères, mais ne permettent pas de bénéficier des fonctionnalités de substitutions du shell, comme par exemple le remplacement d'une variable par sa valeur dans la chaîne de caractères. De plus, elles ne peuvent pas contenir de guillemets simples, puisque c'est leur caractère de terminaison. C'est pour ces raisons que le shell donne la possibilité de définir des chaînes de caractères plus souples, à l'aide des guillemets doubles (caractère `"`). Dans ces chaînes de caractères, la plupart des caractères normalement interprétés par le shell ne le sont plus, comme pour les chaînes de caractères utilisant les guillemets simples. Cependant, les caractères spéciaux `$`, `‘` et `\` conservent leur signification initiale. Il est donc possible, par exemple, d'utiliser des variables d'environnement dans les chaînes de caractères de ce type :

```
echo "Mon nom est $USER"
```

Le caractère d'échappement antislash peut toujours être utilisé, en particulier pour insérer un caractère de guillemets double dans une chaîne de caractères. En effet, ce caractère marquerait la fin de la chaîne de caractère s'il n'était pas précédé d'un antislash.

> **Note:** Remarquez que les guillemets et les caractères d'échappement ne sont utilisés que pour l'analyse de la ligne de commande. Une fois toutes les chaînes de caractères et toutes les substitutions traitées, les guillemets et les caractères d'échappement inutiles sont supprimés. En pratique, ce sont tous les caractères d'échappement et les guillemets qui restent après traitement de la ligne de commande et qui ne font pas partie du résultat d'une des substitutions. Ainsi, la commande suivante :
>
> ```
> echo "Bonjour tout le monde"
> ```
>
> a pour but de passer la chaîne de caractères `Bonjour tout le monde` en tant que premier (et unique) paramètre de la commande **echo**, puis de l'exécuter. Les guillemets ne font pas partie de la chaîne de caractères, ils ont été supprimés par le shell et seul le contenu de la chaîne sera effectivement affiché.
>
> Notez que la commande précédente est très différente de celle-ci :
>
> ```
> echo Bonjour tout le monde
> ```
>
> même si le résultat est le même. En effet, cette dernière commande passe les chaînes de caractères `Bonjour`, `tout`, `le` et `monde` en tant que paramètres (4 au total) à la commande **echo**, alors que l'utilisation des guillemets permet de passer toute la phrase en un seul paramètre. On peut

voir la différence en utilisant plus d'un espace entre chaque mot : les espaces superflus ne sont conservés que dans la première commande.

5.9.6. Les substitutions

L'une des fonctionnalités les plus puissante du shell est sans doute sa capacité à effectuer des substitutions d'expressions par leur valeurs. L'une des substitutions les plus courante est sans doute le remplacement d'une variable par sa valeur, mais le shell peut faire beaucoup plus que cela. Les lignes de commandes peuvent être écrites en utilisant différents types d'expressions spéciales, qui seront remplacées par leur valeur par le shell avant l'exécution de la commande. Ces expressions permettent de spécifier des motifs de chaîne de caractères, d'exprimer des chemins partiels sur des fichiers ou des répertoires, de récupérer la valeur des variables du shell, de calculer des expressions mathématiques, voire même d'inclure le résultat d'une autre commande dans la ligne de commande en cours.

Les mécanismes de substitution décrits ci-dessous sont présentés par ordre de priorité décroissante. Ceci signifie que si une expression substituable contient elle-même une autre expression substituable de priorité inférieure, cette expression sera remplacée après la substitution de l'expression contenante.

5.9.6.1. Génération de chaînes de caractères selon un motif

Il est possible de demander au shell de générer une série de chaînes de caractères selon un motif simple. Ce motif est toujours constitué d'un préfixe, suivi d'une partie variable, suivie d'un suffixe. La partie variable du motif est celle qui subira les substitutions pour générer une liste de chaîne de caractères commençant par le préfixe suivi du résultat de la substitution et se terminant par le suffixe. Cette partie variable doit être spécifié entre accolades, et prend la forme d'une liste de valeurs possibles pour chaque substitution, séparées par des virgules. Par exemple, la commande suivante :

```
ls test{0,1,2,3,4}
```

sera transformée par le shell en la commande suivante :

```
ls test0 test1 test2 test3 test4
```

Note: Ceux qui se souviennent un peu de leurs mathématiques se diront qu'il s'agit là d'une factorisation. C'est rigoureusement exact.

5.9.6.2. Substitution du nom d'utilisateur

Le caractère tilde ('~') est remplacé par le nom de l'utilisateur courant, ou, à défaut de nom, par le chemin sur le répertoire personnel de cet utilisateur. Il est possible de spécifier un autre utilisateur en

donnant le nom de login de cet autre utilisateur immédiatement après le caractère tilde. Par exemple, la commande suivante :

```
cp *.txt ~jean
```

permet de copier tous les fichiers d'extension `.txt` dans le répertoire personnel de l'utilisateur `jean`.

5.9.6.3. Remplacements de variables

Comme il l'a déjà été indiqué plus haut, la valeur des variables du shell et des variables d'environnement peut être récupérée en préfixant le nom de la variable par le caractère dollar ('$'). En fait, cette écriture est l'une des formes les plus simples que peuvent prendre les substitutions de paramètres. En effet, il est possible de remplacer l'expression par une partie seulement de la valeur de la variable, ou une par une autre valeur calculée à partir de celle de la variable.

En pratique, les expressions utilisées par les substitutions de variables peuvent être relativement compliquées, et il peut être nécessaire de les isoler du reste de la ligne de commande à l'aide d'accolades. La syntaxe exacte complète de ce type de substitution est donc la suivante :

```
${expression}
```

où `expression` est l'expression qui définit la chaîne de remplacement à utiliser.

Si cette expression est un nom de variable, ce sera la le contenu de cette variable qui sera utilisée pour la substitution. Il est possible de fournir une valeur par défaut pour le cas où cette variable ne contient rien ou n'est pas définie. Pour cela, on utilisera la syntaxe suivante :

```
${variable :-valeur}
```

où `valeur` est la valeur par défaut à utiliser dans ce cas. Notez que dans la variable reste indéfinie après la substitution. Pour fixer la valeur de la variable à cette valeur par défaut en plus d'effectuer la substitution, on utilisera plutôt la syntaxe suivante :

```
${variable :=valeur}
```

`valeur` a toujours la même signification dans cette syntaxe.

Il est parfois préférable d'afficher un message d'erreur plutôt que de donner une valeur par défaut lorsqu'une variable n'est pas définie. Ceci peut se faire avec la syntaxe suivante :

```
${variable:?message}
```

où `message` est le message à afficher dans le cas où la variable `variable` serait non définie ou de valeur nulle.

Si l'on veut tester si une variable est non définie et renvoyer une valeur spécifique si elle est définie, on utilisera plutôt la syntaxe suivante :

```
${variable:+valeur}
```

où `valeur` est la valeur à renvoyer si la variable est définie. Si la variable n'est pas définie, la substitution sera faite avec la chaîne de caractères vide (l'expression complète sera donc supprimée).

Le shell permet également de faire la substitution avec une sous-chaîne de la valeur de la variable, à partir d'une position donnée et d'une longueur. La syntaxe à utiliser est donnée ci-dessous :

```
${variable:position:longueur}
```

où `position` est la position à laquelle commence la sous-chaîne à extraire, et `longueur` est le nombre de caractères à extraire. Ce dernier champ est facultatif (on ne mettra pas non plus les deux-points précédents si on décide de ne pas spécifier de longueur). Si on ne le précise pas, la sous-chaîne extraite sera constituée du reste de la valeur de la variable à partir de la position indiquée. La position quant à elle doit être positive ou nulle. Une valeur négative indique un point de départ correspondant au nombre de caractères correspondant à partir de la droite de la valeur de la variable. Si l'on veut obtenir la longueur d'une chaîne de caractères contenue dans une variable, on utilisera cette syntaxe :

```
${#variable}
```

où `variable` est toujours le nom de la variable.

Il est également possible de considérer que la valeur d'une variable est une chaîne de caractère préfixée d'une autre chaîne de caractères particulière. Le shell permet d'extraire la chaîne de caractère principale, en supprimant ce préfixe. Pour réaliser cette opération, on utilisera l'une des syntaxes suivantes :

```
${variable#préfixe}
```

ou :

```
${variable##préfixe}
```

où `variable` est la variable contenant la chaîne de caractères à traiter, et `préfixe` est le préfixe à supprimer.

En fait, le préfixe peut être spécifié à l'aide d'un motif de caractères. Ce motif peut correspondre à une partie plus ou moins grande de la valeur de la variable. Dans ce cas, il y a plusieurs manières d'interpréter ce motif, et donc plusieurs choix de préfixes possibles à supprimer. La première syntaxe devra être utilisée lorsque l'on désire supprimer le plus petit préfixe possible correspondant au motif. La deuxième syntaxe, quant à elle, permettra de supprimer le préfixe le plus long. Par exemple, si la variable VAR contient la chaîne de caractères `abbbc`, la commande suivante :

```
echo ${VAR#a*b}
```

affichera la chaîne de caractères `bbc`, car le plus petit préfixe correspondant au motif `a*b` est `ab`. Inversement, la commande :

```
echo ${VAR##a*b}
```

utilisera le préfixe le plus long, à savoir `abbb`. Le résultat de cette substitution sera donc la chaîne de caractères `c`. La syntaxe des motifs de caractères utilisés ici sera précisée dans la Section 5.9.7.

Le shell fournit une syntaxe similaire pour extraire des suffixes de la valeur des variables. Cette syntaxe utilise simplement le caractère `%` au lieu du caractère #. Comme pour les préfixes, le fait de doubler ce caractère implique que le suffixe le plus long correspondant au motif sera utilisé, alors que l'utilisation d'un seul `%` permet de choisir le suffixe le plus court. Ainsi, la commande :

```
echo ${VAR%b*c}
```

affichera la chaîne de caractères `abb`, alors que la commande :

```
echo ${VAR%%b*c}
```

n'affichera que `a`.

Pour terminer ce tour d'horizon des remplacements de variables, nous allons voir les possibilités de recherche et de remplacement du shell dans les chaînes de caractères contenues dans des variables. La syntaxe suivante :

```
${variable/motif/remplacement}
```

permet de rechercher la plus grande sous-chaîne de caractères correspondant au motif `motif` dans la chaîne contenue dans la variable `variable`, et de remplacer cette sous-chaîne par la chaîne de caractères `remplacement`. Par exemple, si la variable VAR contient la chaîne de caractères `abab`, la commande suivante :

```
echo ${VAR/b/d}
```

affichera la chaîne de caractères `adbc`.

Ce remplacement n'est donc effectué qu'une seule fois. Si l'on veut que toutes les occurences du motif soit remplacées par la chaîne de remplacement, il suffit de doubler les caractères `/` :

```
${variable//motif//remplacement}
```

Dans les deux syntaxes, la présence du champ `remplacement` est facultative. Ceci permet de supprimer purement et simplement les sous-chaînes de caractères qui correspondent au motif.

La syntaxe des motifs sera détaillée dans la Section 5.9.7. Cependant, une précision s'impose : si le motif commence par le caractère #, ce motif devra obligatoirement être placé au début de la chaîne de caractères contenue dans la variable. De même, si le motif commence par le caractère %, il devra obligatoirement se trouver à la fin de cette chaîne. Ces deux notations permettent d'obtenir le même effet que les suppressions de préfixes et de suffixes présentées plus haut.

5.9.6.4. Substitution du résultat d'une commande

Le shell peut évaluer une commande apparaissant dans une expression afin de la remplacer par son résultat. Il existe deux syntaxes pour réaliser ce type de substitutions. La première, et la plus classique (voire historique), utilise des guillemets inverses :

```
‘commande‘
```

où `commande` est la commande devant être remplacée par son résultat (c'est à dire ce qu'elle enverra sur le flux standard de sortie). Pour donner un exemple, la commande suivante :

```
kill ‘cat /var/pid/p.pid‘
```

a pour résultat de lancer un signal SIGTERM au processus dont le PID est stocké dans le fichier `/var/pid/p.pid`. La commande **cat** est utilisée pour afficher le contenu de ce fichier, et elle est substituée par ce contenu. En fin de compte, la commande **kill** est appliqué au PID affiché par **cat**.

La deuxième syntaxe utilisable est la suivante :

```
$(commande)
```

où `commande` est toujours la commande à exécuter et à substituer. La différence entre ces deux syntaxes est que dans le premier cas, les caractères `$`, `‘` et `\` sont toujours interprétés par le shell et doivent être précédés d'un antislash s'ils doivent apparaître tels quels dans la commande à substituer, alors que dans le deuxième cas, on peut utiliser tous les caractères sans protection particulière (sauf, bien entendu, la parenthèse fermante, puisqu'elle marque la fin de la commande).

5.9.6.5. Évaluation d'expressions arithmétiques

En général, le shell ne manipule que des chaînes de caractères. Cependant, il est capable d'évaluer des expressions mathématiques simples faisant intervenir des entiers. Pour cela, il faut utiliser la syntaxe suivante :

```
$((expression))
```

où `expression` est l'expression à évaluer.

Les expressions mathématiques peuvent contenir tous les opérateurs classiques du langage C : addition, soustraction, multiplication et division. Il existe en plus un opérateur d'élévation à la puissance, représenté par une double étoile (`**`). Les opérateurs de comparaison de décalage binaires vers la gauche (`<<`) et la droite (`>>`) sont également utilisables, ainsi que les opérateurs de manipulation de bits `&` (« ET binaire »), `|` (« OU binaire »), `^` (« OU binaire exclusif ») et `~` (« négation binaire »).

Comme en C, les comparaisons logiques peuvent également être évaluées, elles valent la valeur 1 lorsque l'expression qui les utilise est vraie, et 0 dans le cas contraire. Les opérateurs disponibles sont `==`, `<`, `<=`, `>`, `>=` et `!=`. Les tests peuvent être composés à l'aide des opérateurs `&&` (« ET logique ») et `||` (« OU logique »).

Les opérateurs d'affectation diverses du langage C `+=`, `-=`, etc… sont également disponibles.

5.9.6.6. Substitution de commandes

Nous avons déjà vu qu'il était possible de récupérer le résultat d'une commande pour l'insérer dans une ligne de commande. Cette technique s'apparente à ce qu'il est possible de faire avec la commande **xargs** et un pipe. De la même manière, le shell fournit une substitution permettant d'obtenir des fonctionnalités similaires à celles fournies par les pipes nommés. Cette substitution est la substitution de commande.

La syntaxe utilisée par les substitutions de commandes est la suivante est similaire à celle des redirections classiques :

`<(command)`

ou :

`>(command)`

où `command` est la commande à substituer.

La première syntaxe permet de lancer une commande en arrière-plan en redirigeant son flux standard de sortie vers un descripteurs de fichiers du shell. Le résultat de cette substitution est le nom du fichier `/dev/fd/n` permettant de lire les données écrite par la commande dans ce descripteur de fichier. En pratique, on utilise donc cette substitution en lieu et place d'un fichier d'entrée pour une commande normale. La deuxième commande permet de lancer également une commande en arrière-plan, mais en redirigeant le flux d'entrée standard de cette commande cette fois. Il est alors possible de fournir les données nécessaires à cette commande en écrivant dans le fichier `/dev/fd/n` dont le nom est fourni par le résultat de la substitution.

Ces deux commandes permettent donc de simplifier l'usage des pipes nommés, en évitant d'avoir à créer un fichier de pipe manuellement et de lancer les deux commandes devant se servir de ce pipe pour communiquer. Ainsi, la commande suivante :

```
cat <(ls)
```

est fonctionnellement équivalente à la série de commande suivante :

```
mkfifo /tmp/lsfifo
ls > /tmp/lsfifo
cat /tmp/lsfifo
rm /tmp/lsfifo
```

Les substitutions de commandes sont donc nettement plus pratiques et plus sûres, car elles n'imposent pas la création d'un fichier de pipe nommé dont le nom peut être choisi arbitrairement.

5.9.6.7. Découpage en mots

Les résultats provenant des substitutions vues précédemment sont systématiquement décomposés en série de mots par le shell avant de poursuivre le traitement de la ligne de commande. Ceci signifie

que les résultats de substitutions sont analysés pour identifier les mots qu'ils contiennent, en se basant sur la notion de séparateur. Par défaut, les séparateurs utilisés sont l'espace, le caractère de tabulation et le retour de ligne, mais il est possible de spécifier des séparateurs différents à l'aide de la variable d'environnement IFS (abréviation de l'anglais « Internal Field Separator »).

Par exemple, le résultat de la commande **ls** dans la commande suivante :

```
echo ‘ls‘
```

est une chaîne de caractères contenant la liste des fichiers du répertoire courant chacun étant séparé du suivant par un caractère de saut de ligne. La substitution du résultat de cette commande est donc soumise au découpage en mots, et chaque caractère de retour à la ligne est interprété comme un séparateur. Par conséquent, cette chaîne de caractère est transformée en une liste de mots, chacun de ces mots étant un des noms de fichiers renvoyés par la commande **ls**. Au final, la commande **echo** est appelée, avec comme paramètres ces noms de fichiers, à raison d'un par paramètre.

> **Note:** Ce découpage en mot est effectué automatiquement par le shell à la suite des substitutions vues précédemment. Ceci signifie en particulier que s'il n'y a pas de substitution, il n'y a pas de découpage en mots non plus.

5.9.6.8. Remplacement des caractères génériques

Si, après avoir appliqué toutes les formes de substitutions précédentes, le shell trouve des caractères génériques `*` et `?` dans l'expression en cours de traitement, il interprétera la partie de l'expression contenant ces caractères comme un motif représentant des chemins Unix de fichiers. Les caractères `*` et `?` auront donc le comportement que l'on a déjà décrit dans la Section 5.5. Ce motif sera donc remplacé par autant de chemins Unix lui correspondant que possible. Rappelons que le caractère générique `*` représente 0 ou plusieurs caractères quelconques, et que le caractère générique `?` représente un caractère et un seul. Les chemins générés sont classés par ordre alphabétique.

Il est possible également de restreindre le jeu de caractère utilisé par le shell pour rechercher les noms de fichiers correspondants au motif. Pour cela, il faut lui indiquer un ensemble de caractères ou de plages de caractères utilisables, séparés par des virgules, et entre crochets. Les plages de caractères sont spécifiées en indiquant le premier et le dernier caractère, séparés par un tiret. Par exemple, la commande suivante :

```
ls [a-c,m-t]*.txt
```

permet d'afficher tous les fichiers dont le nom commence par les lettres `a`, `b`, `c` et les lettres allant de `m` à `t`, et dont l'extension est `.txt`. Vous trouverez de plus amples renseignements sur la syntaxe de ces motifs dans la Section 5.9.7.

Sauf paramétrage pour indiquer explicitement de faire le contraire, le shell ignore systématiquement les répertoires `.` et `..` dans les substitutions. Ceci est très important. En effet, une commande utilisant le caractère générique `*` ne s'appliquera pas sur le répertoire courant et le répertoire père par défaut.

Paramétrer bash pour qu'il prennent en compte ces répertoires peut être extrêmement dangereux, surtout avec une commande telle que `rm -f *`, qui dans ce cas effacerait également le répertoire parent en plus du contenu du répertoire courant !

5.9.7. Les expressions régulières

Les substitutions de variables et de noms de fichiers utilisent des motifs pour identifier des chaînes de caractères. Notez que pour spécifier le caractère `-` lui-même, il suffit de le placer tout seul, à la fin de la liste. De même, pour spécifier le caractère `]` (normalement utilisé pour marquer la fin du jeu de caractères), il faut le placer au début de la liste, juste après le crochet ouvrant.

5.9.8. Structures de contrôle

Tout langage de programmation digne de ce nom dispose de structures de contrôles évoluées permettant de contrôler l'exécution du programme, de réaliser des boucles et de structurer l'ensemble d'un programme. Le shell n'échappe pas à la règle, et fournit la plupart des constructions classiques. Cette section a pour but d'exposer leurs syntaxes.

5.9.8.1. Les instructions composées

Dans le langage du shell, une instruction se termine soit par un retour à la ligne (non précédé d'un antislash), soit d'un point-virgule. Les instructions peuvent être pourtant très complexes, car elles peuvent contenir des pipes et des redirections. En fait, une instruction peut à peu près être définie comme étant une ligne de commande normale du shell.

Le shell permet bien entendu de réaliser des instructions composées, afin de regrouper plusieurs traitements dans un même bloc d'instructions. La méthode la plus simple pour réaliser un bloc d'instruction est tout simplement de les regrouper sur plusieurs lignes, ou de les séparer par des points-virgules, entre accolades. Par exemple, les instructions suivantes constituent un bloc d'instruction :

```
{
    cd /tmp
    rm *.bak
}
```

Notez que l'accolade fermante est considérée comme une instruction à part entière. Ceci signifie que si l'on ne met pas l'accolade fermante sur une ligne indépendante, il faut faire précéder l'instruction précédente d'un point-virgule. De même, il faut le faire suivre d'un autre point-virgule s'il ne se trouve pas à la fin d'une ligne.

Les instructions d'instruction composée créée à l'aide des accolades sont exécutée au sein du shell courant. Les variables qu'elles définissent, ainsi que les changement de répertoires, sont donc toujours valides à l'issue de l'exécution de ces instructions. Si cela n'est pas désirable, on pourra créer des instructions composées à l'aide de parenthèses. Les instructions seront alors exécutées dans un autre

shell, lancé pour l'occasion, et elles n'auront donc pas d'effets de bords imprévus dans le shell appelant. Par exemple, le répertoire courant à l'issue de l'instruction composée précédente est le répertoire `/tmp/`, alors que l'instruction composée suivante :

```
(
    cd /tmp
    rm *.bak
)
```

ne change pas le répertoire courant.

> **Note:** On ne confondra pas les instructions composées utilisant des parenthèses et les substitutions de résultat de commande. Les instructions composées renvoient le code d'erreur de la dernière instruction exécutée, alors que le résultat des substitutions est ce que la commande a écrit sur son flux de sortie standard.

Le shell permet également de réaliser des instructions composées conditionnelles, où l'exécution de chaque instruction de l'instruction composée est conditionnée par le résultat de l'instruction précédente. Ces instructions composées sont définies à l'aide des opérateurs `||` et `&&`. La syntaxe de ces opérateurs est la même :

```
command1 || command2
command1 && command2
```

où `command1` et `command2` sont deux commandes du shell (composées ou non). Avec l'opérateur `||`, la commande `command2` n'est exécutée que si le code d'erreur de la commande `command1` est non nul, ou, autrement dit, si cette commande ne s'est pas exécutée correctement. Inversement, avec l'opérateur `&&`, la commande `command2` n'est exécutée que si la première commande s'est exécutée correctement (et renvoie donc un code d'erreur nul). Par exemple, la commande suivante :

```
rm *.txt 2> /dev/null || echo "Aucun fichier à supprimer"
```

permet d'effacer tous les fichiers d'extension `.txt`, ou d'afficher le message d'erreur `Aucun fichier à supprimer` s'il n'existe pas de tels fichiers.

Les instructions composées peuvent être utilisées comme n'importe quelle commande normale. En particulier, elles peuvent être utilisées dans des commandes plus complexes, par exemple comme destination d'un pipe. C'est ce que faisait l'exemple de déplacement de toute une arborescence dans la Section 5.9.3.1.

5.9.8.2. Les tests

Sous Unix, chaque processus reçoit plusieurs valeurs en paramètres et renvoie un code de retour. La plupart des paramètres sont passés en ligne de commande, et sont récupérés directement par le processus, mais d'autres paramètres peuvent être fournis par le processus appelant par l'intermédiaire

de variables d'environnement et de descripteurs de fichiers. Le code de retour, quant à lui, est un entier signalant si l'exécution du processus s'est terminée correctement ou si des erreurs ont eu lieu. Si les codes d'erreurs varient grandement d'un programme à un autre, la valeur 0 signifie toujours, et ce quel que soit le programme, que l'exécution s'est déroulée correctement.

Il est possible de tester le code de retour d'une commande avec l'instruction `if`. La syntaxe la plus simple pour un test est la suivante :

```
if commande; then
    action
fi
```

où `commande` est la commande dont on désire tester le code de retour, et `action` est la commande à exécuter si ce code vaut 0 (c'est à dire, si la commande `commande` s'est exécutée correctement).

Il peut paraître réducteur de ne pouvoir tester que le code de retour d'une commande. Mais en fait, c'est une fonctionnalité très puissante du shell, car elle permet de réaliser tous les types de tests imaginables. En effet, il existe une commande spéciale, **[**, qui permet de réaliser divers types de tests sur les paramètres que l'on lui passer, et d'ajuster son code d'erreur en conséquence. Par exemple, pour tester l'égalité d'une variable d'environnement avec une chaîne de caractère, on utilisera la syntaxe suivante :

```
if [ $variable == valeur ]; then
    action
fi
```

Notez que dans cette syntaxe, le test effectué est une commande complète. Ceci implique qu'il faut mettre un espace entre chaque paramètre, et en particulier entre le nom de la commande (`[`), le premier opérande (`$variable`), l'opérateur utilisé (`==`), le deuxième opérande (`valeur`) et le caractère de marque de fin de test (`]`).

La commande **[** est capable d'effectuer tous les tests standards. Par défaut, elle considère que les deux opérandes du test sont des chaînes de caractères, et elle utilise l'ordre lexicographique pour les comparer. Les tests d'égalité et d'inégalités sont effectués respectivement avec les opérateurs `==` et `!=`. Les opérateurs d'antériorité dans l'ordre lexicographique sont $<$ et `<=`, et les opérateurs de postériorité sont $>$ et `>=`. Notez que l'utilisation de ces opérateurs peut être relativement pénible, parce que les caractères $<$ et $>$ sont interprétés par le shell en tant que redirections. Par conséquent, il faut souvent les précéder du caractère d'échappement antislash.

L'ordre lexicographique convient dans la plupart des cas, mais il n'est pas très approprié pour la comparaison de chaînes de caractères. Par exemple, le test suivant :

```
if [ -1 \< -2 ]; then
    echo "-1 est plus petit que -2"
fi
```

est vérifié, car le caractère `1` précède le caractère `2` dans l'ordre lexicographique. La commande **[** fournit donc la possibilité d'utiliser une autre syntaxe pour comparer les entiers. Cette syntaxe utilise les options `lt` et `gt` respectivement pour les tests d'infériorité stricte et de supériorité stricte, et les

options `le` et `ge` respectivement pour les tests d'infériorité et de supériorité ou d'égalité. Ainsi, le test :

```
if [ $i -gt 3 ]; then
    echo "$i est supérieur à 3"
fi
```

permet de comparer la valeur entière de la variable i avec le nombre `3`.

Nous avons vu dans la Section 5.9.8.1 que les opérateurs `||` et `&&` permettent de tester le code de retour d'une commande, et qu'en fonction de la valeur de ce code de retour, d'exécuter ou non la commande suivante. La syntaxe de ces opérateurs provient en fait de la possibilité de les employer pour effectuer des tests complexes avec l'instruction `if`. Par exemple, pour effectuer un ET logique entre deux tests, on utilisera la syntaxe suivante :

```
if [ $i == "A" ] && [ $j -lt 3 ]; then
    echo "i contient la lettre \"A\" et j contient un nombre inférieur à 3"
fi
```

Notez que la deuxième commande **[** n'est exécutée que si le premier test est vérifié. L'utilisation de l'opérateur `||` se fait selon le même principe. Il est bien entendu possible de regrouper plusieurs commandes de test ensemble, à l'aide de parenthèses.

Comme dans la plupart des langages informatiques, l'instruction `if` peut prendre des formes plus complexes pour traiter les cas où le test n'est pas vérifié. Ainsi, pour exécuter une action spécifique pour le cas où le test serait faux, on peut utiliser la syntaxe suivante :

```
if commande; then
    action1
else
    action2
fi
```

où `commande` est toujours la commande dont le code de retour sera testé, `action1` est l'action qui doit être réalisée si cette commande a renvoyé le code de retour 0, et `action2` la commande à exécuter dans le cas contraire. De même, si l'on veut enchaîner des tests, on utilisera le mot-clé `elif`. La syntaxe générale du test est donc la suivante :

```
if commande1; then
    action1
elif commande2; then
    action2
elif commande3; then
    ...
else
    actionn
fi
```

> **Note:** Pour des raisons d'optimisations, le shell peut simuler le comportement du programme **[**, et éviter ainsi de le lancer à chaque fois qu'il a à faire un test. Cependant, le principe originel était bien celui décrit ci-dessus, qui, bien que n'étant plus tout à fait exact, permet de mieux comprendre la syntaxe du shell.
>
> Il est possible de récupérer la valeur du code de retour de la dernière commande exécutée grâce à la variable spéciale `$?`. Cependant, il est très rare d'avoir à manipuler cette valeur directement, car les structures de contrôle du shell telles que `if` permettent d'effectuer les actions qui s'imposent sans avoir à la connaître.
>
> Pour ceux qui savent programmer en C, sachez que le code de retour est la valeur renvoyée par la fonction C `exit` ou par l'instruction `return` de la fonction principale `main`. Les paramètres de la ligne de commande, quant à eux, sont récupérables par l'intermédiaire des paramètres de la fonction principale `main`.

Il ne faut pas oublier que la fonction première du shell est de permettre les manipulations de fichiers. Il n'est donc pas étonnant que la commande **[** permette également de réaliser tous les tests imaginables sur les fichiers. Ces tests vont de l'existence d'un fichier à sa nature et ses attributs, en passant par son propriétaire et son groupe. La syntaxe générale de ces tests est la suivante :

```
if [ option fichier ]; then
    ⋮
fi
```

où `option` est une option de la commande **[** décrivant la propriété testée, et `fichier` est le nom du fichier sur lequel le test doit porter.

Les principales options utilisables dans les tests sur les fichiers sont récapitulées dans le tableau ci-dessous :

Tableau 5-4. Tests sur les fichiers

Option	Signification
`-e`	**Test d'existence d'un fichier ou d'un répertoire.**
`-d`	**Test d'existence d'un répertoire.**
`-f`	**Test d'existence d'un fichier normal.**
`-s`	**Test d'existence d'un fichier et vérification que sa taille est non nulle.**
`-L`	**Test d'existence d'un lien symbolique.**
`-b`	**Test d'existence d'un fichier spécial de périphérique de type block (disque dur, CD-ROM, lecteur de cassettes, etc…).**
`-c`	**Test d'existence d'un fichier spécial de périphérique de type caractère (port série, port parallèle, carte son, etc…).**
`-p`	**Test d'existence d'un pipe.**
`-r`	**Test d'existence du fichier et d'accessibilité en lecture de ce fichier.**
`-w`	**Test d'existence du fichier et d'accessibilité en écriture de ce fichier**

Option	Signification
`-x`	**Test d'existence du fichier et de possibilité d'exécution de ce fichier.**
`-g`	**Test d'existence du fichier et de présence du bit setgid sur ce fichier.**
`-u`	**Test d'existence du fichier et de présence du bit setuid sur ce fichier**
`-k`	**Test d'existence du fichier et de présence du bit sticky sur ce fichier.**
`-O`	**Test d'existence du fichier et d'appartenance de ce fichier à l'utilisateur effectif courant.**
`-G`	**Test d'existence du fichier et d'appartenance de ce fichier au groupe effectif courant.**
`-N`	**Test d'existence du fichier et de modification de ce fichier depuis la dernière fois qu'il a été lu.**

> **Note:** Ce tableau n'est pas exhaustif, mais les options les plus importantes et les plus utilisées s'y trouvent.
>
> Vous pourrez vous rafraîchir la mémoire sur les notions de bit setuid, setgid et sticky, ainsi que sur les notions d'utilisateur et de groupe effectif en relisant la Section 3.2.

La commande **[** accepte également les options `-nt` et `-ot`, qui permettent respectivement de tester si un fichier est plus récent ou plus vieux qu'un autre, en se basant sur les dates de dernière modification de ces fichiers. Ces deux opérateurs s'utilisent avec la syntaxe suivante :

```
if [ fichier1 option fichier2 ]; then
    ⋮
fi
```

où `fichier1` et `fichier2` sont les deux fichiers sur lesquels la comparaison doit porter, et `option` est l'une des options `-nt` ou `-ot`.

5.9.8.3. Le branchement conditionnel

Lorsque l'on veut effectuer différentes opérations selon la valeur d'une variable, l'instruction `if` peut devenir très lourde à utiliser. En effet, si le nombre de valeurs différentes est grand, elle peut conduire à écrire un grand nombre de tests. Le shell fournit donc une instruction de branchement conditionnel, qui permet de spécifier quelle action doit être prise pour chaque valeur de la variable.

Le branchement conditionnel s'utiliser de la manière suivante :

```
case valeur in
    ( motif1 | motif2 | ... ) commande1;;
    ( motifn | motifn+1 | ... ) commande2;;
     ⋮
esac
```

où `motif1`, `motif2`, etc... `motifn+1` sont des motifs spécifiant les valeurs possibles pour la valeur `valeur`, et `commande1`, `commande2`, etc... sont les commandes à exécuter pour les valeurs de ces motifs.

La commande exécutée est la première commande pour laquelle la variable correspond à l'un de ses motifs correspondants. Une fois exécutée, la recherche se termine, et l'exécution reprend à la suite du branchement conditionnel. Par exemple ce branchement conditionnel :

```
case $i in
    ( *.txt ) echo "$i est un fichier texte";;
    ( *.gz ) echo "$i est compressé avec gzip";;
    ( *.tar ) echo "$i est une archive";;
esac
```

affiche la nature du fichier dont le nom est stocké dans la variable i à partir de son extension.

Le code de retour du branchement conditionnel est 0 si la variable ne correspond à aucun des motifs, ou le code de retour de la commande exécutée sinon.

5.9.8.4. Les boucles

Il existe deux types de boucles : le `while` et le `until`. La syntaxe des boucles `while` est la suivante :

```
while commande; do
    action
done
```

où `commande` est une commande dont le code de retour est utilisé comme critère de la fin de la boucle, et `action` est l'instruction (composée ou non) exécutée à chaque itération de la boucle. Comme on le voit, le shell utilise le même principe pour les boucles que pour les tests pour évaluer une condition. Tant que la commande `commande`> renvoie un code de retour égal à 0, l'instruction `action` est exécutée.

L'instruction `while` utilise la même syntaxe que l'instruction `while` :

```
until commande; do
    action
done
```

à ceci près que l'instruction `action` est exécutée tant que la commande `commande` renvoie un code de retour non nul. L'instruction `until` utilise donc simplement le test inverse de celui de l'instruction `while`.

Bien entendu, il est possible d'utiliser la commande **[** pour effectuer des tests plus complexes que le simple test du code de retour d'une commande. Par exemple, la boucle suivante calcule la somme des dix premiers entiers :

```
result=0
```

```
i=0
while [ $i -le 10 ]; do
    result=$(($result + $i))
    i=$(($i + 1))
done
echo $result
```

5.9.8.5. Les itérations

Les itérations sont des boucles qui s'exécutent pour chaque élément d'un ensemble donné. Le shell gère les itérations par l'intermédiaire de l'instruction `for`. La syntaxe de cette instruction est la suivante :

```
for variable [ in ensemble ]; do
    action
done
```

où `variable` est un nom de la variable utilisée pour l'itération, `ensemble` est l'ensemble des valeurs que peut prendre cette variable, et `action` est la commande (simple ou composée) à exécuter pour chaque valeur de cette variable.

Le principe des itérations est très simple. Pour chaque valeur indiquée dans l'ensemble des valeurs, la commande est exécutée, avec la valeur en question accessible dans la variable utilisée pour l'itération. Par exemple, la commande suivante :

```
for i in *.txt; do
    mv $i ${i/%.txt/.doc}
done
```

permet de renommer tous les fichiers portant l'extension `.txt` en fichier du même nom, mais avec l'extension `.doc`.

Il n'est pas nécessaire de préciser l'ensemble des valeurs que peut prendre la variable. Dans ce cas, l'ensemble utilisé sera celui de tous les paramètres du script ou de la fonctions. Nous verrons plus loin comment réaliser des fonctions et des scripts, ainsi que la manière de récupérer leurs paramètres.

5.9.8.6. Les ruptures de séquence

Il est parfois nécessaire de modifier l'ordre d'exécution dans les boucles et les itérations du shell. Par exemple, il est souvent nécessaire de sortir de la boucle courante, soit parce qu'on ne peut plus la continuer dans de bonnes conditions, soit parce que le traitement est terminé. C'est notamment le cas lorsqu'une erreur se produit, ou lorsqu'on recherche une valeur spécifique en itérant sur les valeurs possibles d'un ensemble.

Le shell fournit donc les instructions `break` et `continue`, qui permettent respectivement de sortir de la boucle courante et de passer directement à l'itération suivante. Ces deux commandes peuvent

être utilisées aussi bien à l'intérieur des boucles `while` et `until` que dans les itérations écrites avec l'instruction `for`. Par exemple, le calcul de la somme des dix premiers entiers aurait pu être écrit de la manière suivante :

```
result=0
i=0
while true; do
    result=$(($result + $i))
    i=$(($i + 1))
    if [ $i ==11 ]; then break; fi
done
echo $result
```

Les instructions `break` et `continue` peuvent prendre un paramètre entier indiquant le niveau d'imbrication de la boucle sur laquelle elles s'appliquent. Ce paramètre doit impérativement être supérieur à sa valeur par défaut, c'est à dire `1`. Ainsi, pour sortir directement d'une double boucle lorsque l'on est dans le corps de la boucle la plus imbriquée, on utilisera la commande suivante :

```
break 2
```

5.9.8.7. Les fonctions

Le langage du shell est un langage procédural. Ceci signifie que l'on peut créer des fonctions pour regrouper des séries d'instructions couramment exécutées. La syntaxe permettant d'écrire de telles fonctions est la suivante :

```
function nom () {
    instructions
}
```

où `nom` est le nom de la fonction, et `instructions` est la liste des commandes à exécuter dans cette fonction.

Vous constaterez qu'il n'y a pas de déclaration des paramètres de cette fonction. C'est normal : les paramètres des fonctions sont passés implicitement dans les variables d'environnement $1, $2, $3, etc… En fait, comme nous le verrons plus loin, cette syntaxe est également celle utilisée pour récupérer les paramètres de la ligne de commande des scripts shell. Ceci signifie que les paramètres du scripts ne sont pas accessibles dans le corps d'une fonction, puisqu'ils sont masqués par les paramètres de la fonction.

Les autres variables utilisées dans les fonctions sont des variables globales. Celles qui sont déclarées dans une fonction sont donc également globales, et restent accessibles même après l'exécution de cette fonction. Si l'on veut définir des variables locales, on précédera la définition de la variable du mot-clé `local` :

```
local variable=valeur
```

où `variable` est le nom de la variable locale, et `valeur` est sa valeur.

Les fonctions peuvent retourner une valeur numérique en code de retour. Cette valeur peut être indiquée à l'aide de l'instruction `return`. Par exemple, la fonction suivante calcule la somme des entiers de 0 à la valeur de l'entier qu'elle reçoit en paramètre :

```
function somme () {
    local result=0
    local i=0
    while [ $i -le $1 ]; do
        result=$(($result + $i))
        i=$(($i + 1))
    done
    return $result
}
```

Ce code d'erreur pourra être récupéré par l'appelant dans la variable d'environnement $? :

```
somme 10
echo $?
```

5.9.8.8. Les entrées / sorties de données

Tout langage de programmation qui se respecte dispose de possibilités d'entrée / sortie pour permettre la communication avec l'utilisateur de manière interactive, et le shell n'échappe pas à la règle.

Nous avons déjà vu la commande **echo** dans bon nombre des exemples qui précédaient, et vous avez sans doute deviné qu'il s'agissait là de la commande qui permet d'afficher du texte à l'écran. Son utilisation est des plus simple, puisqu'elle se contente d'envoyer sur le flux de sortie standard une chaîne de caractère contenant tous les paramètres qu'elle reçoit, séparés par des espaces. Nous ne nous attarderons donc pas sur cette commande, qui n'a pas dû vous poser de problèmes jusqu'à présent.

Il ne nous reste donc plus qu'à voir la manière de demander à l'utilisateur de saisir une valeur. Avec bash, la demande de saisie des données se fait classiquement à l'aide de la commande **read**. Cette commande lit une ligne sur le flux d'entrée standard, la découpe en une ou plusieurs données et place les résultats dans les variables d'environnement qu'elle reçoit en paramètre. La syntaxe de **read** est donc la suivante :

```
read variable1 variable2 ... variablen
```

où `variable1`, `variable2`, etc... sont les noms des variables d'environnement dans lesquelles les résultats de la saisie doivent être placés.

La commande **read** utilise les séparateurs indiqués dans la variable d'environnement IFS pour découper la ligne lue dans le flux d'entrée standard. Si le nombre de variables spécifiées est inférieur au nombre de mots de cette ligne après découpage, les premières variables d'environnement reçoivent

les premiers mots, et la dernière reçoit les autres mots, séparés par leurs séparateurs respectifs. Par exemple, la commande suivante :

```
read MOT RESTE
```

permet de lire le premier mot d'une ligne dans la variable d'environnement MOT et de placer le reste dans la variable RESTE.

La commande read dispose d'une syntaxe simplifiée, qui ne prend aucun paramètre. Dans ce cas, la ligne lue dans le flux d'entrée standard est placée telle quelle dans la variable d'environnement REPLY. Il est à la charge du programmeur d'analyser son contenu.

Le shell dispose également d'une instruction évoluée permettant de réaliser des menus simplifiés : l'instruction `select`. Cette instruction construit un menu à partir d'un certain nombre de choix, chaque choix étant précédés par un numéro, et demande à l'utilisateur de taper le numéro de son choix. Elle affecte alors la valeur du choix correspondant à une variable d'environnement, et exécute une commande pour le traitement du choix. La syntaxe générale de l'instruction `select` est donnée ci-dessous :

```
select variable in liste; do
    action
done
```

où `variable` est le nom de la variable devant recevoir la valeur choisie par l'utilisateur, `liste` est la liste des valeurs que cette variable peut prendre, et `action` est la liste des instructions à exécuter pour chaque choix effectué.

Si le choix de l'utilisateur est incorrect, la variable de contrôle reçoit la valeur nulle. Le programmeur peut récupérer la valeur saisie par l'utilisateur dans la variable d'environnement REPLY et effectuer un traitement d'erreur approprié.

L'instruction `select` est une boucle. Le menu est reproposé après chaque exécution de l'action `action`. La sortie de cette boucle ne peut se faire que si un caractère de fin de fichier (`CTRL + D`) est lu sur le flux d'entrée standard, ou si une option de menu spécifique est proposée pour quitter cette boucle. Vous trouverez un exemple de menu simplifié ci-dessous :

```
select LU in A B C D Sortir; do
    case  $LU in
        ("A") echo "Vous avez choisi A";;
        ("B") echo "Vous avez choisi B";;
        ("C") echo "Vous avez choisi C";;
        ("D") echo "Vous avez choisi D";;
        ("Sortir") break;;
    esac
done
```

5.9.9. Les alias

Il est incontestable que certaines commandes peuvent avoir une grande complexité, et il peut être fastidieux de les retaper complètement à chaque fois que l'on en a besoin. D'autre part, la saisie d'une longue ligne de commande multiplie les risques de fautes de frappe et d'avoir à corriger la commande. Ceci peut au mieux faire perdre son temps à l'utilisateur, et au pire l'énerver.

Le shell fournit donc un mécanisme simplifié pour donner un nom simplifié aux commandes complexes : le mécanisme des *alias*. Les alias représentent en fait des chaînes de caractères complexes, et sont remplacés automatiquement par le shell lorsqu'il analyse les lignes de commandes. C'est un mécanisme plus souple que celui des variables d'environnement, et permet de définir des macro commandes plus facilement qu'avec les fonctions du shell.

Pour créer un alias, vous devrez utiliser la syntaxe suivante :

```
alias nom=chaîne
```

où `nom` est le nom de l'alias, et `chaîne` est la chaîne de caractère représentée par cet alias. Par exemple, pour faire un alias nommé `beep` permettant de faire un bip sonore, on pourra utiliser la commande suivante :

```
alias beep="echo $'\a'"
```

Cet alias pourra être simplement utilisé simplement en tapant `beep` en ligne de commande.

Vous pourrez visualiser la liste des alias existant simplement à l'aide de la commande **alias**, appelée sans paramètres. Je vous recommande de consulter cette liste, pour vous donner une idée des alias courant, qui se révèlent généralement très utiles.

La suppression des alias se fait à l'aide de la commande **unalias**. Sa syntaxe est la suivante :

```
unalias nom
```

où `nom` est le nom de l'alias à supprimer.

> **Note:** Les alias ne sont remplacés par la chaîne de caractères qu'ils représentent que lorsque le shell analyse la ligne de commande. Ceci signifie que les définitions d'alias ne sont valides qu'après validation de cette ligne. On évitera donc de définir des alias dans la déclaration d'une instruction composée, car cet alias ne sera pas disponible à l'intérieur de son propre bloc d'instruction.
>
> Par défaut, les alias ne sont disponibles que dans les shells interactifs. Ils ne peuvent donc pas être utilisés dans les scripts shell. La notion de script shell est détaillée dans la Section 5.9.10.

5.9.10. Les scripts shell

Pour l'instant, toutes les fonctionnalités de bash, aussi puissantes soient-elles, ne constituent que l'interface d'un interpréteur de commande puissant. Mais nous avons dit que le shell était véritablement un langage de programmation. Ceci signifie qu'il est possible d'écrire des programmes complexes en

langage shell, simplement en stockant plusieurs commandes dans un fichier. On appelle ces fichiers des *script shell*.

L'écriture d'un script shell n'est pas plus compliquée que de taper les commandes du programme les unes à la suite des autres dans un shell interactif. La seule différence est que les scripts shell peuvent être rejoués plusieurs fois, recevoir des paramètres en ligne de commande et renvoyer un code de retour.

Tout script shell est en fait un fichier texte sur lequel on a mis les droits d'exécutions. Il contient les différentes commandes du shell qu'il doit exécuter. Sa première ligne est très importante, elle permet d'indiquer au shell exécutant quel est la nature du fichier. La syntaxe de cette ligne est la suivante :

```
#!shell
```

où `shell` est le chemin absolu sur le shell ou l'interpréteur de commande capable d'exécuter ce script. En pratique, pour bash, on utilisera toujours la ligne suivante :

```
#!/bin/bash
```

Les paramètres des scripts shells sont accessibles exactement comme des paramètres de fonction. On récupérera donc le premier paramètre avec l'expression `$1`, le deuxième avec l'expression `$2`, le troisième avec l'expression `$3`, etc...

Le code de retour d'un shell pourra être fixé à l'aide de la commande **exit**. Par exemple :

```
exit 0
```

Ce code de retour pourra être récupéré par l'appelant à l'aide de l'expression `$?`.

Nous n'irons pas plus loin dans la description du shell bash, car ce n'est pas le but de ce document. Vous pouvez vous référer à un bon livre d'Unix ou aux pages de manuel si vous désirez approfondir le sujet. Comme vous avez dû vous en rendre compte dans cette section, les shells Unix sont des véritables langages de programmation, qui dépassent très loin les interpréteurs de commandes du type DOS. De plus, il existe plusieurs autres langages dont nous n'avons pas parlé ici, chacun étant conçu souvent pour réaliser un certain type de tâche (administration système, manipulation de fichiers textes, création de pages Web dynamiques, création d'interfaces utilisateurs en mode fenêtré, pilotage d'applications, etc...). Si vous vous y intéressez, vous verrez que le sujet est réellement vaste et passionnant.

Chapitre 6. Configuration du système de base

La configuration du système de base est un peu moins sensible que son installation. En pratique, les opérations de configuration consisteront à manipuler les fichiers de configuration du système. Les seuls risques que l'on encourt sont de les détruire, et donc de devoir les recréer manuellement. C'est pour cette raison que nous allons commencer par les sauvegarder, afin de pouvoir revenir à l'étape de configuration du système de base sans repasser par la case départ.

Ceci dit, il se peut fort bien que le programme d'installation ou le programme de configuration de votre distribution vous permette d'éviter cette tâche. Dans ce cas, il vous sera sans doute demandé de répondre à quelques questions, et ce programme effectuera les modifications pour vous. Il est d'ailleurs recommandé de toujours essayer la configuration du système avec un tel programme, car lui seul connaît les spécificités de votre distribution (puisqu'il est fourni avec !). Malheureusement, ces programmes ne peuvent pas tout prévoir, parce que Linux est un système capable d'effectuer un nombre de tâches très diversifié d'une part, et parce que ce que vous voulez en faire personnellement ne correspond pas forcément à un standard prédéterminé d'autre part.

Cette partie décrira le mécanisme général d'amorçage des systèmes Linux et les commandes d'administration les plus importantes. Elle décrira également comment des périphériques additionnels (clavier, souris, cartes son, vidéo, réseau, imprimante) et les configurer pour obtenir un fonctionnement correcte sous Linux. Cependant, la première étape est bien entendu de faire une sauvegarde de tous les fichiers de configuration, car la moindre erreur peut provoquer de lourdes conséquences (jusqu'à la réinstallation complète du système).

6.1. Sauvegarde de la configuration d'installation

La sauvegarde de la configuration du système est une opération facile à réaliser. En effet, tous les fichiers de configuration sont placés dans le répertoire `/etc/`. Par conséquent, il suffit de faire une archive des fichiers de ce répertoire et de ses sous répertoires. Cette opération peut être réalisée avec la commande suivante :

```
tar cvfz /root/install.conf.tar.gz /etc/*
```

Cette commande créera une archive nommée `install.conf.tar.gz` dans le répertoire personnel de l'administrateur système. On notera que pour certaines distributions, quelques fichiers de configuration sont placés dans le répertoire `/sbin/init.d/`. Pour ces distributions, on utilisera donc plutôt la commande suivante :

```
tar cvfz /root/install.conf.tar.gz /etc/* /sbin/init.d/*
```

De cette manière, si l'on a un gros problème avec la configuration de la machine, on peut revenir simplement à la configuration utilisée juste après l'installation du système avec la simple commande suivante :

```
tar xvfz /root/install.conf.tar.gz
```

que l'on exécutera dans la racine du système de fichiers.

Cette commande écrasera tous les fichiers existants par ceux de la sauvegarde. Les fichiers qui ont été ajoutés depuis cette sauvegarde seront bien entendu conservés.

Il est également recommandé de faire une sauvegarde identique à celle-ci une fois que l'on aura réussi à configurer le système correctement, et que théoriquement, il n'y aura plus à toucher aux fichiers de configuration. Cette sauvegarde devrait être placée sur une disquette que l'on conservera en lieu sûr.

6.2. Notion de niveau d'exécution

La plupart des systèmes Unix utilisent la notion de niveaux d'exécution. Un niveau d'exécution est un mode de fonctionnement dans lequel un certain nombre de services sont accessibles. En général, il existe 7 modes d'exécutions, dont seulement trois fournissent des services bien définis pour quasiment toutes les distributions de Linux.

Le niveau 0 correspond à l'arrêt du système, et aucun service n'est disponible (à part le redémarrage de la machine bien entendu…). Le fait de passer dans le niveau d'exécution 0 correspond donc à arrêter le système. Le niveau 6 correspond au redémarrage de la machine. Le fait de passer dans le niveau d'exécution 6 revient donc à arrêter et à redémarrer la machine. Le niveau d'exécution 1 correspond au mode de fonctionnement mono-utilisateur (encore appelé mode de maintenance). Ce niveau d'exécution fournit les services de base pour un seul utilisateur (normalement l'administrateur du système). Dans ce niveau d'exécution, l'administrateur peut changer la configuration et effectuer les tâches de maintenance les plus critiques (par exemple, contrôler la partition root). La signification des autres niveaux d'exécution dépend de la distribution que vous utilisez, mais en général, le niveau d'exécution 2 correspond au mode multi-utilisateur avec réseau sans XWindow, et le niveau d'exécution 3 ou 4 correspond au mode multi-utilisateur avec login graphique sous XWindow. Les autres niveaux restent à votre disposition.

Le programme en charge de gérer les niveaux d'exécution est le programme « init ». Ce programme est le premier programme lancé par le noyau après qu'il a démarré. Il ne peut pas être détruit, et c'est réellement le processus père de tous les autres dans le système. Le rôle d'init est de récupérer les processus zombies (c'est à dire les processus qui viennent de se terminer et dont le processus père s'est terminé avant eux) et de gérer les changements de niveau d'exécution.

Il suffit d'utiliser la syntaxe suivante pour forcer le changement de niveau d'exécution :

```
init niveau
```

où `niveau` est le niveau d'exécution à atteindre. Ceci dit, cette manière de faire est assez rare, car en général on n'a pas besoin de changer de niveau d'exécution, sauf pour arrêter et redémarrer la machine. Mais pour ces opérations, les commandes **shutdown**, **halt** et **reboot** sont déjà disponibles.

init utilise le fichier de configuration `/etc/inittab`. Ce fichier définit les niveaux d'exécution, le niveau par défaut dans lequel le système se place au démarrage, et les actions qu'init doit prendre lorsque certains événements arrivent. En particulier, il est indiqué quels sont les scripts qui doivent être

exécutés lors du changement de niveau d'exécution. Il est fortement, mais très fortement, déconseillé de toucher au fichier `/etc/inittab` pour des raisons bien évidentes. Vous trouverez de plus amples renseignements dans les pages de manuel d'init et d'inittab.

Lorsque l'on change de niveau d'exécution, ainsi qu'au démarrage du système, des scripts de configuration sont appelés. Comme on l'a vu, ces scripts sont spécifiés dans le fichier `/etc/inittab`. En général, ils sont tous placés dans le répertoire `/etc/rc.d/` (ou `/sbin/init.d/`, selon votre distribution). Ce répertoire contient donc :

- le script exécuté lors du démarrage du système ;
- les scripts exécutés lors de la sortie d'un niveau d'exécution ;
- les scripts exécutés lors de l'entrée dans un niveau d'exécution.

Le script appelé lors du démarrage du système est en général spécifié directement dans `/etc/inittab`. Vous pouvez y ajouter les commandes spécifiques à votre système, comme par exemple les commandes de configuration du matériel. Ce fichier n'est exécuté qu'une seule fois et est placé directement dans `/etc/rc.d/` (ou dans `/sbin/init.d/`).

En revanche, les scripts appelés lors du changement de niveau d'exécution sont souvent placés dans des sous-répertoires du répertoire `rc.d` ou `init.d`. Ils sont classés à raison d'un répertoire par niveau d'exécution. Ces sous-répertoires portent le nom de `rc0.d`, `rc1.d`, `rc2.d`, etc... pour les différents niveaux d'exécution. En fait, un seul script est exécuté par init lorsque l'on change de niveau d'exécution, et ce script se charge d'exécuter les bons scripts dans les sous-répertoires de `rc.d` ou `init.d`. Classiquement, ce script principal est appelé avec le numéro du niveau d'exécution en paramètre, et il commence par appeler les scripts de sortie du niveau d'exécution courant, puis les scripts d'entrée dans le nouveau niveau d'exécution.

La distinction entre les scripts d'entrée et de sortie dans chaque répertoire `rc?.d` se fait par la première lettre du script. Sur certaines distributions, la lettre '`K`' correspond aux scripts de sortie et la lettre '`S`' au script d'entrée (ces deux lettres correspondent respectivement aux mots anglais « Kill » et « Start »). De plus, l'ordre dans lequel ces scripts doivent être exécutés est indiqué par le nombre suivant cette lettre dans le nom du script. Ceci dit, ces conventions peuvent varier selon votre distribution. Consultez votre documentation pour plus de détails à ce sujet.

Il est assez courant que les répertoires `rc?.d` ne contiennent que des liens symboliques vers les fichiers de scripts, et que ceux-ci soient tous placés directement dans le répertoire `/etc/rc.d/` (ou `/sbin/init.d/`). La raison en est qu'un même script peut être utilisé pour différents niveaux d'exécution, et qu'il n'a donc pas de raison d'être dans le répertoire d'un niveau plutôt que celui d'un autre.

De même, il est assez courant que chacun de ces scripts gère à la fois l'entrée et la sortie du niveau d'exécution, selon le paramètre qu'il reçoit lors de son appel. Parmi les paramètres les plus courants, on retrouve les suivants :

- `start`, pour le démarrage du service correspondant ;
- `stop`, pour son arrêt.

Ce sont les deux paramètres que le script de contrôle de changement de niveau d'exécution (celui appelé par init et enregistré dans `/etc/inittab`) utilisera lors de l'entrée et de la sortie du niveau d'exécution. Il existe d'autres paramètres, comme par exemple `restart`, pour redémarrer le service correspondant.

De cette manière, vous pouvez ajouter ou supprimer des services simplement dans chaque niveau d'exécution. Il suffit d'écrire un fichier script capable de prendre en paramètre l'action à réaliser sur le service (`start` ou `stop`), de le placer dans `/etc/rc.d/` (ou `/sbin/init.d/`) et de créer les liens dans les sous-répertoires `/etc/rc.d/rc?.d/` (ou `/sbin/init.d/rc?.d/`). Dès lors, votre service sera arrêté ou redémarré selon le niveau d'exécution dans lequel passera le système.

La rédaction des scripts shells de configuration dépasse largement le cadre de ce document, de même que la configuration du comportement du système à chaque changement de niveau d'exécution. La description qui était donnée ici permet simplement d'avoir une idée plus claire de la manière dont le système se comporte au démarrage et à l'arrêt. Consultez la documentation de votre distribution pour plus de détails à ce sujet.

6.3. Notion de fichiers spéciaux de périphériques

Comme il l'a déjà été expliqué dans les chapitres précédent, Linux gère tous les périphériques comme des fichiers spéciaux. Ceci simplifie énormément leur utilisation par les programmes d'application, puisque la plupart des opérations sur un périphérique revient simplement à réaliser une écriture ou une lecture. Évidemment, l'écriture sur un fichier spécial de disque permet d'enregistrer les données sur ce disque, et la lecture permet de les récupérer. Mais cela ne s'arrête pas là ! Par exemple, la communication avec le modem se fait simplement en écrivant et en lisant les données sur le fichier spécial du port série sur lequel le modem est connecté. De même, jouer un fichier son revient simplement à l'écrire dans le fichier spécial qui gère la carte son. Il est même possible d'accéder à la mémoire vidéo par l'intermédiaire d'un fichier spécial de périphérique…

Bien entendu, certaines fonctionnalités avancées des périphériques ne peuvent pas être utilisées simplement par une une écriture ou une lecture dans un fichier spécial. Le système fournit donc aux applications d'autres moyens d'accéder à ces fonctionnalités, par l'intermédiaire d'appels systèmes spécifiques (pour ceux qui sont intéressés par la programmation système, cet appel système est réalisé par la fonction `ioctl`, dont le nom provient de l'abréviation de l'anglais « Input / Output ConTroL »). Évidemment, cette méthode n'est utilisée que par les programmes qui connaissent bien le fonctionnement du gestionnaire du périphérique, car ils doivent spécifier une requête que seul ce gestionnaire comprend en général. Quoi qu'il en soit, les requêtes de ce type utilisent elles-aussi un descripteur de fichier spécial de périphérique, ce qui fait que tous les accès au matériel ont lieu par l'intermédiaire de ces fichiers.

Il existe deux principaux types de fichiers spéciaux de périphérique. Le premier type correspond aux périphériques de *type bloc*, dont les données ne peuvent être lues que par blocs (c'est le cas des disques durs, des lecteurs de CD et de disquettes en particulier). Le deuxième type correspond aux périphériques de *type caractère*, dont les données peuvent être lues caractère par caractère (cartes son, ports série, etc…). En plus de son type, chaque fichier spécial de périphérique est caractérisé par deux codes permettant d'identifier le type et le modèle du périphérique auquel il donne accès.

Ces deux codes portent le nom de *code majeur* et de *code mineur*. Comme nous le verrons plus tard, c'est par l'intermédiaire de ces codes que le noyau est capable de retrouver le gestionnaire de périphérique à utiliser pour satisfaire aux requêtes des clients qui accèdent à un fichier spécial de périphérique. Il y a donc, en général, une association unique entre ces codes et les gestionnaires de périphériques. La liste des codes majeurs et mineurs déjà affectés à des périphériques est donnée dans le fichier `/usr/src/linux/Documentation/devices.txt` (ce fichier vous donnera donc une idée des périphériques qui peuvent être ou qui seront gérés par Linux).

Les fichiers spéciaux de périphériques sont tous stockés dans le répertoire `/dev/`. En général, ce répertoire contient un grand nombre de fichiers spéciaux de périphériques, même pour des périphériques inexistants sur votre machine. Ces fichiers sont installés automatiquement par les distributions, qui laissent au noyau le soin de signaler que le périphérique est absent lorsqu'un programme tente d'accéder à une fichier spécial auquel aucun matériel ne correspond. Les fichiers fournis par les distributions sont donc, en thérorie, susceptible de couvrir toutes les gammes de périphériques courants, et vous n'aurez pas à toucher au répertoire `/dev/`. Cependant, si vous devez créer un fichier spécial de périphérique vous-même, vous devrez utiliser la commande **mknod**. Sa syntaxe est relativement simple :

```
mknod fichier type majeur mineur
```

où `fichier` est le nom du fichier spécial de périphérique à créer, `type` est une lettre indiquant le type du fichier spécial (`'c'` pour les fichiers de type caractère et `'b'` pour les fichiers de type bloc), et `majeur` et `mineur` sont les codes majeurs et mineurs du périphérique.

Depuis la version 2.4.0 du noyau, il est également possible que le répertoire `/dev/` contienne un système de fichiers virtuel, géré par le noyau. Ce système de fichiers est peuplé automatiquement par le noyau avec les fichiers spéciaux de périphériques de tous les périphériques détectés lors de l'initialisation du système. Cette technique permet de résoudre plusieurs problèmes concernant les fichiers spéciaux de périphériques. Le plus évident pour l'utilisateur est que seuls les fichiers spéciaux des périphériques effectivement présents apparaissent dans le répertoire `/dev/`, ce qui élimine la présence de tous les fichiers spéciaux inutiles.

Le gros avantage du système de fichiers virtuel `/dev/` est que les fichiers spéciaux sont créés par les gestionnaires de périphériques eux-même. Ceci implique que l'association entre les fichiers spéciaux et les gestionnaires de périphériques du noyau qui en ont la charge est directe, et ne nécessite plus d'avoir recours aux codes majeurs et mineurs. Ceci a plusieurs conséquences :

- il n'y a plus besoin d'avoir recours à une autorité centrale pour obtenir les valeurs de codes mineurs et majeurs. Les fichiers spéciaux de périphériques continuent d'avoir des codes majeurs et mineurs bien qu'ils ne soient plus nécessaires, mais les valeurs de ces codes sont attribués dynamiquement de manière unique par le noyau lorsque les gestionnaires de périphériques s'initialisent. Il n'y a donc plus de risques de conflits entre deux périphériques ;
- la limitation du nombre de périphériques due au nombre limité de codes majeurs et mineurs n'existe plus (il suffit de consulter la liste du fichier `/usr/src/linux/Documentation/devices.txt` pour constater qu'il reste peu de codes libres pour les périphériques à venir) ;

- les performances sont supérieures, puisque la recherche du périphérique associé à un fichier spécial de périphérique n'a plus lieu ;
- la configuration du système en est simplifiée.

Par ailleurs, le système de fichiers virtuel est beaucoup plus structuré que le répertoire `/dev/` classique, car les fichiers spéciaux de périphériques sont classés par type et par nature. Ce système de fichiers constitue donc une image logique et cohérente de l'architecture du matériel de l'ordinateur. Ainsi, la recherche d'un périphérique est beaucoup plus facile. En revanche, les chemins sur les fichiers spéciaux de périphériques ne sont sont nettement plus longs que les chemins classiques. Par exemple, le fichier spécial de périphérique `/dev/hda1` permettant d'accéder à la première partition du premier disque IDE sera accessible avec le chemin `/dev/ide/host0/bus0/target0/lun0/part1` dans le système de fichiers virtuel.

Ce changement dans la structure du répertoire `/dev/` est un changement de taille, propre à empêcher la plupart des applications système de fonctionner normalement. C'est pour résoudre ce problème que le démon devfsd a été écrit. Ce démon, s'il est lancé juste après que le système de fichiers virtuel ait été créé (donc, en pratique, dans les premiers scripts de démarrage du système), surveille la création des fichiers spéciaux de périphériques dans le système de fichier virtuels et permet ainsi d'effectuer des actions spécifiques lorsqu'un gestionnaire de périphérique s'enregistre ou se décharge du niveau. Les actions à effectuer sont spécifiées dans le fichier de configuration `/etc/devfsd.conf`. Par défaut, le démon devfsd crée des liens symboliques portant les noms des anciens fichiers spéciaux de périphérique vers les fichiers du système de fichiers virtuel. Ainsi, les anciennes applications peuvent continuer à utiliser les anciens noms de fichiers spéciaux pour accéder aux périphériques. La suite de ce document utilisera donc les noms de fichiers spéciaux de périphériques classiques. Vous devrez faire la transcription pour toutes les commandes manipulant les périphériques si vous utilisez le système de fichier virtuel avec sa nouvelle structure.

> **Note:** Un démon est un processus qui tourne en arrière plan dans le système. Les démons fonctionnent souvent dans le compte root et offrent des services de base que les autres programmes utilisent. Le terme « démon » provient de la traduction littérale « daemon », ce qui signifie « ange » en réalité. Le démon se place donc en tant qu'intermédiaire entre Dieu (c'est à dire le noyau) et les hommes (c'est à dire les applications normales). Le terme « daemon » a été ensuite éclairci et défini comme l'abréviation de l'anglais « Disk And Execution MONitor ».

Même si votre distribution utilise le système de fichier virtuel `/dev/`, il se peut que vous ayez à créer des fichiers spéciaux manuellement. Ceci peut être nécessaire pour créer un fichier spécial de périphérique pour un gestionnaire qui ne gère pas encore le système de fichier virtuel `/dev/`. La syntaxe à utiliser est strictement la même que dans le cas des fichiers spéciaux classiques.

Je vous invite à consulter la page de manuel de la commande **mknod** si vous devez créer vous-même un fichier spécial de périphérique. Ce ne sera normalement pas nécessaire. Vous pouvez également obtenir de plus amples informations sur le démon devfsd et son fichier de configuration dans la page de manuel `devfsd`.

6.4. Configuration de LILO

L'installation de LILO est quasiment un passage obligé sur les ordinateurs utilisant Linux, car c'est l'un des gestionnaires d'amorçage les plus puissants. Il permet de démarrer Linux comme tout autre système d'exploitation très simplement, en donnant le nom du système à lancer lors de l'amorçage. Il est bien entendu possible de lancer un système par défaut, et de donner un temps d'attente avant de choisir cette option si l'utilisateur n'a rien saisi.

LILO est constitué de deux parties. La première partie peut s'installer sur le secteur principal d'amorçage du disque dur (« MBR »), ou sur le secteur de boot de n'importe quelle partition. Comme on l'a déjà indiqué plus haut, il est fortement recommandé d'installer cette partie de LILO sur le secteur de boot de la partition Linux, afin d'éviter qu'elle ne soit écrasée par le DOS ou par Windows. La deuxième partie est enregistrée directement dans la partition Linux. Elle contient les informations nécessaires pour pouvoir charger les différents systèmes d'exploitation gérés. Bien entendu, la première partie est capable de retrouver directement la deuxième sur le disque dur, car lors de l'amorçage les systèmes de fichiers de Linux ne sont pas encore chargés.

LILO utilise le fichier de configuration `/etc/lilo.conf`. Ce fichier contient la description des différents systèmes d'exploitation que LILO doit proposer au démarrage. Les options les plus importantes de ce fichier sont les suivantes :

- l'option `boot`, qui permet d'indiquer sur quel secteur d'amorçage LILO doit s'installer. Cette option suit la syntaxe suivante :

  ```
  boot = destination
  ```

où `destination` est le fichier spécial du périphérique sur lequel LILO va s'installer. Ainsi, si l'on précise `/dev/hda`, LILO va s'installer sur le MBR du premier disque du premier contrôleur de disques IDE. De même, `/dev/hda3` demande à LILO de s'installer sur le secteur de boot de la troisième partition de ce même disque ;

- l'option `read-only` permet de demander au noyau de monter la partition root en lecture seule lors du démarrage. Cette option est nécessaire pour que les scripts de démarrage du système puisse effectuer les vérifications du système de fichier de cette partition si nécessaire. La partition sera remontée en lecture et en écriture une fois ces vérifications réalisées ;
- l'option `prompt` permet de demander à LILO de demander le système à lancer à chaque démarrage. Cette option force donc l'apparition du message d'invite de LILO au démarrage : `LILO boot:` auquel on pourra répondre en tapant le nom de la configuration à lancer ;
- l'option `timeout` permet de fixer un délai au delà duquel LILO lancera la première configuration définie dans le fichier `lilo.conf`. La syntaxe de cette option est la suivante :

  ```
  timeout = dixièmes
  ```

où `dixièmes` est le nombre de dixièmes de secondes à attendre avant le lancement du système.

La suite du fichier `lilo.conf` décrit les différentes configurations que LILO peut lancer. Les sections de configuration permettant de charger Linux ont le format suivant :

```
image = noyau
root = root_device
label = nom
```

où `noyau` est le chemin complet sur le noyau de Linux à charger, `root_device` est le nom complet du fichier spécial de périphérique contenant le système de fichier root et `nom` est le nom de la configuration tel qu'il devra être saisi à l'invite de LILO. L'exemple donné ci-dessous permet de charger le noyau `/boot/vmlinuz` en utilisant la partition `/dev/hda2` comme partition racine :

```
image = /boot/vmlinuz
root = /dev/hda2
label = linux
```

Si vous désirez créer une section de configuration permettant de lancer un système DOS (ou Windows 9x), vous pouvez utiliser le modèle suivant :

```
other = partition
table = disque
label = nom
```

où `partition` est la partition où se trouve le système DOS à lancer, `disque` est le disque contenant la table des partitions utilisée par ce système, et `nom` est le nom de cette configuration. Vous pouvez vous inspirer de l'exemple fourni ci-dessous :

```
other = /dev/hda1
table = /dev/hda
label = dos
```

qui permet de lancer le système situé sur la première partition du premier disque IDE.

L'exemple donné ci-dessous permet de donner la possibilité de charger Linux ou le DOS, en lançant Linux par défaut au bout de 10 secondes. Le DOS est installé sur la première partition, et Linux utilise la deuxième et la troisième partition respectivement pour y stocker son noyau et sa partition racine. LILO est ici installé sur la partition contenant le noyau de Linux :

```
# Exemple de fichier de configuration /etc/lilo.conf :

# Options générales :
boot = /dev/hda2
read-only
prompt
timeout=100

# Première configuration (Linux) :
image = /dev/hda2
root = /dev/hda3
label = linux
```

```
# Deuxième configuration (DOS) :
other = /dev/hda1
table = /dev/hda
label = dos
```

Remarquez que si l'on utilise `/dev/hda` à la place de `/dev/hda2` après l'option `boot`, LILO s'installera sur le MBR du disque et non pas sur le secteur de boot de Linux. Dans ce cas, vous prendrez des risques si vous réinstallez le système DOS, car celui-ci écrasera le MBR de votre disque, supprimant ainsi LILO. Si vous désirez poursuivre dans cette voie (ce qui peut être nécessaire si Linux est installé sur un disque esclave par exemple), vous pouvez réaliser une sauvegarde de votre MBR à l'aide de la commande suivante :

```
dd if=/dev/hda of=fichier bs=512 count=1
```

où `fichier` est le nom du fichier devant recevoir l'image du MBR.

L'installation de LILO est très simple une fois que l'on a écrit le fichier `lilo.conf`. En effet, il suffit tout simplement de taper la commande suivante :

```
lilo
```

Si lilo signale une erreur, il vaut mieux ne pas insister et corriger le fichier `lilo.conf`.

Lorsque la machine démarre, LILO affiche son invite de démarrage : `LILO boot :`

Il attend ici que vous indiquiez le nom du système que vous désirez démarrer. Attention, le clavier est ici en Américain, et la disposition des touches est donc QWERTY. Vous devez ici taper le nom du système à charger et valider : `LILO boot :linux`

Si vous ne tapez rien, et que vous avez donné un délai d'attente dans le fichier de configuration de LILO, la première configuration sera lancée automatiquement après ce délai.

Il est possible de spécifier des paramètres de démarrage à la suite du nom du système utilisé. Ces paramètres servent principalement à renseigner le noyau sur la configuration matérielle (en particulier les ports d'entrée/sortie et les lignes d'interruption des périphériques non Plug and Play), pour le cas où il ne parviendrait pas à la déterminer automatiquement. L'un des paramètres les plus intéressant est sans doute `mem`, qui permet d'indiquer au noyau la taille de la mémoire vive dont dispose l'ordinateur. Ce paramètre peut être nécessaire si vous disposez de plus de 64 Mo de mémoire, parce que les fonctions du BIOS ne permettent pas d'indiquer les tailles de mémoire plus grandes. Par exemple, si votre ordinateur dispose de 128 Mo de mémoire, vous devrez taper la ligne de paramètres suivante au démarrage : `LILO boot :linux mem=128M`

Bien entendu, il est possible d'enregistrer ces paramètres dans le fichier de configuration de LILO. Pour cela, il suffit d'indiquer le paramètre de démarrage du noyau dans une ligne `append` de la section de configuration de Linux :

```
append="paramètre"
```

Ainsi, la section de configuration de Linux du fichier `lilo.conf` exemple donné ci-dessus pourrait être remplacée par celle-ci sur une machine disposant de 128 Mo de mémoire :

```
# Première configuration (Linux) :
image = /dev/hda2
root = /dev/hda3
label = linux
append="mem=128M"
```

Le niveau d'exécution dans lequel le système doit se placer lors de son démarrage peut également être précisé en paramètre du noyau lors du démarrage. Vous devrez donc utiliser une commande semblable à celle-ci : `LILO boot :linux niveau` pour démarrer Linux dans le niveau d'exécution `niveau`. Ainsi, pour passer en mode mono-utilisateur (c'est à dire le mode de maintenance), il suffit de taper la commande suivante à l'amorçage de LILO :

```
LILO boot:linux 1
```

Il est également possible d'utiliser le paramètre `single`, qui est synonyme du niveau d'exécution 1.

6.5. Vérification du disque dur

La vérification du disque dur est une opération que l'on ne devrait jamais avoir à faire soi-même. Il y a plusieurs raisons à cela. Premièrement, si l'on arrête le système correctement avant d'éteindre la machine, les systèmes de fichiers sont démontés et ils sont dans un état correct. Deuxièmement, les systèmes de fichiers Unix sont réputés pour être très fiables. Troisièmement, une vérification périodique est faite par le système au bout d'un certain nombre de démarrages. Enfin, si un système de fichiers n'est pas démonté correctement avant l'arrêt du système, celui-ci effectuera automatiquement une vérification de ce système de fichiers au démarrage suivant, ce qui fait qu'il n'y a pas lieu de le faire soi-même.

Toutefois, si l'on désire effectuer une vérification manuelle, il faut savoir qu'un système de fichiers ne se manipule que lorsqu'il est démonté. Ceci pose évidemment quelques problèmes pour le système de fichiers root, puisqu'on ne peut pas accéder aux outils de vérification sans le monter. Pour ce système de fichiers, il n'y a donc que deux possibilités :

- soit on utilise une disquette de démarrage contenant les outils de vérification et de réparation des systèmes de fichiers ;
- soit on monte le système de fichiers root en lecture seule.

La deuxième solution est la seule réalisable si l'on ne dispose pas de disquette de démarrage. Par conséquent, c'est cette méthode qui sera décrite ici.

La première étape consiste à passer en mode mono-utilisateur, afin de s'assurer que personne ni aucun programme n'accède à la partition root en écriture. Pour cela, il suffit de taper la commande suivante :

```
init 1
```

qui fait passer le système dans le niveau d'exécution 1. On peut également passer le paramètre `single` au noyau lors de l'amorçage du système, comme il l'a été expliqué dans le paragraphe précédent. Ensuite, il faut remonter la partition root en lecture seule, ce qui se fait avec la commande suivante :

```
mount -n -o remount,ro /
```

l'option `remount` permet de démonter et de remonter la partition racine, et l'option `ro` indique qu'elle doit être remontée en lecteur seule (« ro » signifie « Read Only »). Les options sont séparées par des virgules (attention, il ne faut pas insérer d'espaces). De plus, l'option `-n` indique à **mount** qu'il ne doit pas écrire dans le fichier `/etc/mtab` lorsqu'il aura remonté la partition, parce qu'elle sera alors en lecture seule et qu'il ne pourra pas y parvenir. Ce fichier est utilisé par **mount** pour mémoriser les partitions qui sont montées, afin d'éviter de les monter plusieurs fois de suite.

Lorsque la partition root sera montée en lecture seule, on pourra utiliser le programme **fsck** afin de vérifier, et éventuellement réparer, la partition root. En réalité, ce programme ne fait rien d'autre que d'appeler un programme spécifique pour chaque système de fichier. Par exemple, pour les systèmes de fichiers EXT2, **fsck** appelle le programme **e2fsck**, qui est capable de vérifier et de réparer ce type de systèmes de fichiers.

La ligne de commande à utiliser pour vérifier la partition racine avec **fsck** est la suivante :

```
fsck -a périphérique
```

où `périphérique` est le fichier spécial du périphérique contenant le système de fichiers à vérifier. Il faut donc spécifier la partition sur laquelle le système de fichier root se trouve. L'option `-a` demande à **fsck** d'effectuer les éventuelles corrections automatiquement en cas d'erreur sur le système de fichiers ainsi vérifié, sans confirmation de la part de l'utilisateur.

Il est également possible de demander la vérification de tous les systèmes de fichiers enregistrés dans le fichier de configuration `/etc/fstab`. Pour cela, il faut ajouter l'option `-A` :

```
fsck -a -A
```

Il n'est évidemment plus nécessaire de spécifier le fichier spécial du périphérique contenant le système de fichiers à vérifier, puisque tous les systèmes de fichiers enregistrés dans le fichier `/etc/fstab` seront vérifiés.

Une fois que vous aurez terminé la vérification du système de fichiers, vous pourrez le remonter en lecture et écriture avec la commande suivante :

```
mount -n -o remount,rw /
```

Cette commande est similaire à celle que l'on a vue pour monter le système de fichiers en lecture seule, à ceci près que l'option `rw` est utilisée à la place de l'option `ro`. Cette option permet de remonter le système de fichiers en lecture et en écriture (« rw » est l'abréviation de l'anglais « Read Write »).

6.6. Configuration du montage des systèmes de fichiers

Le montage des systèmes de fichiers est une opération assez fastidieuse. Heureusement, elle peut être automatisée pour les systèmes de fichiers situés sur les disques fixes, et simplifiée pour les systèmes de fichiers amovibles. Pour cela, il faut enregistrer ces systèmes de fichiers dans le fichier de configuration `/etc/fstab`.

Ce fichier contient une ligne pour chaque système de fichiers. Ces lignes contiennent plusieurs champs séparés par des espaces. Les informations suivantes sont enregistrées dans ces champs :

- le fichier spécial permettant d'accéder au système de fichiers ;
- le répertoire servant de point de montage par défaut ;
- le type du système de fichiers ;
- les options de montage pour ce système de fichiers ;
- un entier indiquant quels systèmes de fichiers doivent être sauvegardés ;
- un entier indiquant le rang dans lequel les systèmes de fichiers doivent être vérifiés.

Tous les systèmes de fichiers disposant de l'option `auto` seront montés automatiquement au démarrage du système par la commande **mount -a**. Les autres systèmes de fichiers sont montables manuellement, avec les autres options indiquées dans le fichier `/etc/fstab`. Grâce à ces informations, l'emploi de la commande **mount** est plus simple :

```
mount périphérique | répertoire
```

où `périphérique` est le fichier spécial de périphérique contenant le système de fichiers à monter, et `répertoire` est le répertoire servant de point de montage indiqué dans le fichier `/etc/fstab`. Il est possible d'utiliser indifféremment le fichier spécial de périphérique ou le répertoire du point de montage.

Le type du système de fichiers est l'un des types disponibles accepté par la commande **mount**. Consultez la page de manuelle de cette commande pour plus de renseignements à ce sujet. Les principales options disponibles pour le montage sont les suivantes :

- l'option `defaults`, qui permet de choisir les options par défaut pour ce système de fichiers ;
- l'option `auto`, qui permet de faire en sorte que le système de fichiers soit monté automatiquement au démarrage du système ;
- l'option `user`, qui permet d'autoriser le montage de ce système de fichiers par les utilisateurs ;

- l'option `ro`, qui permet de monter le système de fichiers en lecture seule ;
- l'option `rw`, qui permet de monter le système de fichiers en lecture et écriture ;
- l'option `exec`, qui permet d'autoriser l'exécution des fichiers exécutables sur ce système de fichiers ;
- l'option `uid=utilisateur`, qui permet de spécifier le numéro `utilisateur` de l'utilisateur propriétaire du répertoire racine de ce système de fichiers ;
- l'option `gid=groupe`, qui permet de spécifier le numéro `groupe` du groupe d'utilisateur auquel le répertoire racine du système de fichier appartient ;
- l'option `mode=valeur`, qui permet de fixer les droits sur le répertoire racine du système de fichiers à monter. La valeur `valeur` est donnée en octal ;
- l'option `umask=valeur`, qui permet de fixer les droits sur les fichiers qui ne sont pas gérés par le système de fichiers. La valeur `valeur` » est donnée en octal.

Par défaut, les utilisateurs n'ont pas le droit de monter et de démonter les systèmes de fichiers. L'option `user` permet de désactiver cette protection. Elle peut être utile pour permettre le montage et le démontage des disquettes et des CD-ROM. De même, l'exécution des fichiers exécutables n'est par défaut pas autorisée sur les systèmes de fichiers. Cette restriction permet d'éviter l'exécution de programmes placés sur des systèmes de fichiers de systèmes d'exploitation différents. Elle peut être levée grâce à l'option `exec`.

Les options `ro` et `rw` permettent d'indiquer à **mount** si le système de fichiers doit être monté en lecture seule ou en lecture et écriture. Les systèmes de fichiers devant être réparés doivent être montées en lecture seule si l'on ne peut pas ne pas les monter (c'est le cas notamment du système de fichiers root). Il en va de même pour les CD-ROM, car on ne peut bien entendu pas écrire dessus.

Les options `uid` et `gid` permettent de spécifier le propriétaire et le groupe du répertoire racine du système de fichiers à monter. Par défaut, c'est l'utilisateur root qui devient propriétaire de ce système de fichiers.

Enfin, l'option `mode` permet de spécifier les droits d'accès sur tous les fichiers du système de fichiers à monter. L'option `umask` permet quant à elle de fixer les droits qui ne sont pas gérés par le système de fichiers. Ce peut être utile pour les systèmes de fichiers FAT et FAT32. Il est ainsi possible de donner les droits de lecture et d'exécution pour les fichiers de ces systèmes avec une valeur de masque nulle. Ceci permet de monter les systèmes de fichiers FAT et FAT32 de telle sorte que tous les fichiers appartiennent à l'utilisateur root par défaut, et de donner cependant tous les droits à tous les utilisateurs sur ces fichiers. On prendra garde à ces options, car elles permettent à quiconque d'écrire des fichiers sous le nom de root, et donc constituent un grave défaut dans la sécurité du système.

Les deux derniers champs de `/etc/fstab` spécifient des options pour des programmes annexe. L'avant dernier contrôle le comportement du programme de sauvegarde **dump**, et le dernier celui du programme de vérification de système de fichiers **fsck**. Consultez les pages de manuel pour plus de détails à ce sujet.

Si vous disposez de systèmes de fichiers FAT ou FAT32, vous pourrez monter ces partitions automatiquement lors du démarrage du système. Comme les systèmes de fichiers basés sur la FAT ne peuvent

pas gérer les droits des utilisateurs, vous allez devoir faire un choix pour fixer ces droits à une valeur raisonnable. Vous pouvez par exemple donner le droit de lecture à tous les utilisateurs, mais le droit d'écriture uniquement à l'administrateur système. La ligne à ajouter dans le fichier `/etc/fstab` sera alors la suivante :

```
/dev/partition  répertoire    vfat    auto,exec    0    0
```

où `partitition` est la partition contenant le système de fichiers FAT, et `répertoire` est le répertoire servant de point de montage pour cette partition. Cette ligne permettra de monter automatiquement ce système de fichiers en tant que FAT32 au démarrage du système. Les fichiers binaires seront exécutables, bien qu'ils ne soient pas stockés sur un système de fichier EXT2.

Par ailleurs, certaines distributions spécifient des options incorrectes pour le système de fichiers `/dev/pts/` dans le fichier `/etc/fstab`. Veuillez vous assurer que la ligne utilisée pour ce système de fichiers est bien identique à la ligne suivante :

```
none            /dev/pts      devpts  auto,gid=5,mode=620    0    0
```

Si ce n'est pas le cas, certains émulateurs de terminaux développés récemment ne fonctionneront pas correctement. Le système de fichiers `/dev/pts/` est en effet un système de fichiers virtuel, géré directement par le noyau, dans lequel des fichiers spéciaux de périphériques utilisés par les émulateurs de terminaux sont placés. Si les droits ne sont pas correctement fixés sur le répertoire racine de ce système de fichiers, les émulateurs de terminaux utilisant cette fonctionnalité ne pourront se connecter au système que sous le compte root. Il faut donc impérativement corriger cette ligne, si vous voulez que les utilisateurs normaux puissent utiliser ces émulateurs. Notez également qu'il faut que tout le monde ait les droits d'écriture et de lecture sur le fichier spécial de périphérique `/dev/ptmx` pour que les utilisateurs non privilégiés puissent utiliser ce système de fichiers virtuel.

Vous devrez également ajouter le système de fichiers virtuel `/var/shm/` dans votre fichier `/etc/fstab`. Ce système de fichiers permet aux applications qui le désirent d'utiliser des segments de mémoire partagée, afin de réaliser des communications inter-processus. Ce type de communications est très couramment utilisée, aussi faut-il vous assurer que la ligne suivante se trouve bien dans la fichier `fstab` :

```
none            /var/shm         shm  defaults             0    0
```

Bien entendu, vous devrez créer le point de montage `/var/shm/` si celui-ci n'existe pas avant de monter ce système de fichiers virtuels.

Enfin, si vous utilisez des périphériques USB, vous pouvez monter le système de fichiers virtuel `/proc/bus/usb/` en ajoutant la ligne suivante dans le fichier `fstab` :

```
none            /proc/bus/usb   usbdevfs  defaults          0    0
```

Cette ligne doit être placée après celle qui effectue le montage du système de fichiers virtuel `/proc/`.

6.7. Mise à l'heure du système

Les systèmes d'exploitation utilisent l'heure pour un certain nombre de tâches. En particulier, les fichiers disposent de plusieurs dates (date de création, date d'accès et date de dernière modification), qui sont utilisées par différents programmes. Les programmes de sauvegarde en font évidemment partie, parce qu'ils se basent sur les dates de modification des fichiers pour déterminer quels sont les fichiers qui doivent être sauvegardés depuis la dernière sauvegarde (cas de sauvegarde dite « incrémentale ». Les programmes de maintenance sont également lancés à des dates précises, et les applications normales des utilisateurs peuvent utiliser la date système pour l'intégrer dans leurs documents. En clair, il est important que votre système soit à l'heure.

En fait, il existe deux horloges dans votre système. La première horloge, qui est l'horloge de référence pour toutes les opérations effectuées dans le système, est l'horloge dite « système ». Cette horloge est maintenue par le noyau grâce à un compteur, qui est incrémenté régulièrement, sur la base d'une interruption matérielle. La précision de ce compteur est a priori la même que celle de l'interruption du timer matériel. Sur les PC, cette interruption a lieu 18,6 fois par secondes, ce qui donne pour la plus petite unité de temps mesurable environ 1/20ème de secondes. La deuxième horloge est l'horloge matérielle, qui est l'horloge qui maintient l'heure de votre ordinateur pendant qu'il est éteint. Cette horloge est couramment appelée l'horloge CMOS, parce qu'elle est gérée par le composant CMOS, qui stocke toutes les informations permanentes du BIOS.

Pour répondre immédiatement à une question (désormais sans objet), précisons que Linux n'a aucun problème vis à vis des dates critiques du changement de millénaire. En effet, les systèmes Unix n'utilisent qu'un seul format de date au niveau application : le nombre de secondes écoulées depuis le 01/01/1970 à 0 heure. Ce compteur est stocké sur 32 chiffres binaires sur la plupart des machines et passe donc allègrement le cap de l'an 2000. En fait, le débordement de ce compteur est prévu pour 2038, mais n'aura jamais lieu car l'apparition des processeurs 64 bits va porter à 64 bits sa taille. Ceci étant, il est possible que certaines applications mal écrites n'utilise pas ce format de date, et ne soit donc pas compatibles. Heureusement, ce cas de figure est très rare sous Unix. Bien entendu, le problème reste entier si l'horloge matérielle de votre PC n'est pas compatible. Dans ce cas, la solution la plus simple est de régler l'heure système à chaque démarrage, manuellement ou à l'aide de scripts de correction de la date renvoyée par l'horloge matérielle.

La valeur du compteur de l'horloge système est toujours interprétée en temps universel (« UTC » en anglais, abréviation de « Universal Time Coordinated »), c'est à dire le temps de référence valide dans le monde entier. Ce temps ne comprend pas les fuseaux horaires ni les réglementations concernant les heures d'hiver et d'été. Cette convention est utilisée partout dans le système, ce qui est la condition sine qua non pour que tous les ordinateurs du monde utilisent la même date et la même heure. Ainsi, deux ordinateurs connectés à Internet peuvent communiquer sans se poser de questions quant à leurs localisations respectives, ce qui simplifie beaucoup les choses. Notez également que le fait de compter le temps en secondes permet de s'affranchir des conventions de découpage du temps et des calendriers utilisés dans chaque pays.

Bien entendu, les dates présentées à l'utilisateur doivent être traduites en temps local, corrigé des écarts pour l'heure d'été et l'heure d'hiver. Ceci est réalisé grâce par tous les programmes qui doivent afficher ces dates (par exemple, les simples commandes **ls** et **date**). Cette conversion est effectuée par le système en fonction du fuseau horaire et des plages de validité des horaires d'été et d'hiver.

La solution la plus simple pour régler la date et l'heure de votre machine est donc de régler l'horloge

matérielle sur le temps universel, et de définir le fuseau horaire dans lequel elle se trouve, pour que le système puisse calculer l'heure locale. Malheureusement, les systèmes d'exploitation de Microsoft ne voient pas la chose de la même manière. Ils attendent que l'horloge matérielle soit réglée à l'heure locale. Par conséquent, si Linux est installé sur un ordinateur disposant déjà de Windows, vous devrez régler l'heure de votre ordinateur en temps local. A priori, ceci ne fait aucune différence, le système étant également capable de calculer le temps universel à partir de l'heure locale et de la zone horaire. Cependant, ceci a un inconvénient : il est nécessaire de mettre à l'heure l'horloge système en cas de déplacement de la machine, et à chaque changement d'horaire d'été ou d'hiver. Bien sûr, Windows est supposé être capable de mettre à jour l'heure matérielle en « observation avec l'heure d'été / d'hiver ». Mais il utilise pour cela des règles qui sont fixées définitivement dans le système et qui ne peuvent pas être mises à jour avec les réglementations locales (par exemple, la règle de changement d'heure a été modifiée en 1996, si bien que Windows 95 n'a jamais pu fonctionner correctement sur ce point…).

Quoi qu'il en soit, la mise à l'heure d'un système Linux requiert la définition de la zone horaire, la mise à l'heure du système et la mise à l'heure de l'horloge matérielle. La définition de la zone horaire est primordiale et doit avoir lieu avant toute autre opération, car le réglage des horloges dépend évidemment de cette zone.

Les zones horaires sont définies par un ensemble de règles, qui comprennent chacune la période de validité de la règle (en général avec une date de départ et une date de fin) et la différence entre le temps universel et le temps local lorsque cette règle s'applique (gestion des horaires d'été et d'hiver compris). Toutes ces règles portent le nom de la zone géographique dans laquelle elles sont valides. Vous pourrez trouver des exemples de définitions de règles (ainsi que l'historique des conventions concernant le temps) dans le répertoire « timezone » des sources de la librairie C GNU.

Les fichiers de règles des zones horaires doivent être compilées avec le programme **zic** et installés dans le répertoire `/usr/share/zoneinfo`. Normalement, votre système dispose de la totalité des règles, déjà compilées, des différentes zones horaires du monde. Le programme **zic** permet également de définir la zone horaire active. Cette opération se fait dans les fichiers de démarrage de votre système, avec une commande similaire à la suivante :

```
zic -l zone
```

où `zone` est le chemin relatif du fichier de définition des règles de la zone horaire locale, par rapport au répertoire de base `/usr/share/zoneinfo`. Pour les systèmes situés en France métropolitaine, la commande utilisée est donc celle-ci :

```
zic -l Europe/Paris
```

Une fois la zone horaire fixée, il est possible de régler l'horloge système. Il existe deux solutions pour cela. La première solution est d'utiliser la commande système **date**. Cette commande, appelée sans paramètres, permet d'obtenir la date système, exprimée en temps local. Mais elle permet également de modifier la date et l'heure système avec l'option `-s`. La syntaxe complète utilisée est donnée ci-dessous :

```
date -s "MM/JJ/AAAA HH :MM :SS"
```

Il n'est pas nécessaire de préciser l'année si celle-ci ne doit pas être changée. De même, vous pouvez ne donner que l'heure, si la date du jour est correcte. En revanche, vous devez obligatoirement préciser l'heure si vous changez la date. Notez que l'heure doit être donnée en temps local, à moins que l'option `-u` ne soit précisée. Le système réglera son horloge en temps universel automatiquement, selon les règles de zones horaires en vigueur qui ont été indiquées par **zic**. Vous pouvez obtenir l'heure exacte en appelant le 3699 (appel gratuit).

La deuxième solution est celle qui est utilisée au démarrage du système. Elle consiste à initialiser l'horloge système à partir de l'horloge matérielle. Cette opération se fait normalement à l'aide de la commande **clock** (qui en fait est un lien symbolique vers `hwclock`, mais la commande Unix traditionnelle est **clock**). La syntaxe de cette commande est la suivante :

```
clock [-u] -s | -w | -a
```

L'option `-s` permet d'initialiser l'horloge système à partir de la date et de l'heure stockée dans l'horloge matérielle. C'est typiquement cette commande qui est utilisée dans les scripts de démarrage du système. L'option `-w` permet de réaliser l'opération inverse, c'est à dire sauvegarder la date et l'heure de l'horloge système dans l'horloge matérielle. Elle n'est en générale utilisée qu'après avoir remis à l'heure l'horloge système. L'option `-a` permet, quant à elle, de corriger l'avance ou le retard que l'horloge matérielle peut prendre.

Ce dernier point mérite quelques explications complémentaires. En fait, l'horloge matérielle n'est pas extrêmement précise, et peut se décaler petit à petit de l'heure réelle. Heureusement, ce décalage est constant, ce qui fait qu'il est possible de le mesurer et de le prendre en compte. Le programme **clock** utilise le fichier `/etc/adjtime` pour enregistrer de combien est ce décalage afin de pouvoir effectuer les corrections. Le principe de fonctionnement est le suivant :

- lors du premier réglage de l'horloge matérielle (avec l'option `-w`), il enregistre l'instant de ce réglage dans le fichier `/etc/adjtime` ;
- lors des réglages suivants, il calcule le temps qui s'est écoulé depuis le réglage précédent, et le décalage entre l'heure de l'horloge matérielle et l'heure à laquelle celle-ci aurait due se trouver. Il enregistre ce décalage et met à jour la date de mise à l'heure (pour pouvoir refaire ce calcul ultérieurement) ;
- lorsqu'on l'appelle avec l'option `-a`, clock ajuste l'horloge matérielle. Pour cela, il regarde la date courante, calcule le temps écoulé depuis la dernière mise à l'heure ou le dernier ajustement, en déduit l'avance ou le retard de l'horloge matérielle, et la remet à l'heure en conséquence. Il enregistre également la date de cet ajustement comme nouvelle date de mise à l'heure, afin de ne pas faire deux fois l'ajustement pour cette période la prochaine fois.

De cette manière, il est possible de maintenir l'horloge système à une valeur proche de la réalité (sans ce genre de mécanisme, il est courant de prendre 5 minutes d'écart en trois ou quatre mois, ce qui est déjà considérable).

Les scripts d'initialisation de votre système doivent donc certainement contenir au moins les deux lignes suivantes après le réglage de la zone horaire :

```
# Ajuste l'horloge matérielle :
clock -a
# Initialise l'horloge système :
clock -s
```

Dans tous les cas, l'option `-u` permet d'indiquer que l'horloge matérielle est réglée en temps universel. Si votre machine ne dispose pas d'autre systèmes que Linux, il est recommandé de procéder ainsi et d'utiliser systématiquement cette option.

> **Note:** Il est important de définir la zone horaire avec **zic** avant d'utiliser **clock**. En effet, si l'horloge matérielle est réglée en temps local, clock ne pourra pas déterminer l'heure en temps universel. D'autre part, **clock** initialise la structure de zone horaire interne noyau, que celui-ci utilise notamment pour l'écriture des dates en temps local sur les systèmes de fichiers FAT (Eh oui, les dates des fichiers des systèmes de fichiers FAT sont enregistrées en temps local...).
>
> Sachez également que l'horloge système peut également se décaler sensiblement sur de longues périodes. Évidemment, ce phénomène ne peut se détecter que si le système reste actif suffisamment longtemps, ce qui en pratique ne se produit que dans les serveurs (n'oubliez pas que Linux peut fonctionner des mois sans interruption...). Si vous êtes intéressés par la manière de resynchroniser l'horloge système pour de telles configurations, vous devriez vous intéresser à la diffusion du temps sur le réseau Internet avec le protocole NTP (« Network Time Protocol »). En général, la resynchronisation de l'heure système doit se faire progressivement afin de ne pas perturber la ligne du temps pour les applications. Ceci peut être fait avec le programme **adjtimex**.

6.8. Configuration du lancement automatique des tâches

Il est possible de déclencher l'exécution de certaines opérations à intervalle régulier sous Linux. Ces opérations sont définies pour le système et pour chaque utilisateur. Elles sont enregistrées dans des fichiers de configuration indiquant le moment où elles doivent être déclenchées, et quelle action elles doivent réaliser. Les opérations définies pour le système sont stockées dans le fichier de configuration `/etc/crontab`. Des commandes additionnelles peuvent être définies dans les répertoires `/etc/cron.d/`, `/etc/cron.daily/`, `/etc/cron.weekly/` et `/etc/cron.monthly/`. Par ailleurs, les fichiers de configuration des utilisateurs sont stockés dans le répertoire `/var/cron/tab/`, sous le nom de chaque utilisateur. Il est bien entendu possible d'éditer ces fichiers en tant que root, mais ce n'est pas recommandé. En effet, la commande **crontab** permet d'installer, de supprimer et de consulter les fichiers `crontab` de chaque utilisateur, et ce de manière sûre.

La commande **crontab** peut être utilisée pour afficher le contenu du fichier de configuration de l'utilisateur qui l'appelle, à l'aide de l'option `-l` :

```
crontab -l
```

Elle permet également de supprimer ce fichier, à l'aide de l'option `-r` :

```
crontab -r
```

Enfin, l'option `-e` permet d'éditer le fichier `crontab`, à l'aide de l'éditeur spécifié dans la variable d'environnement VISUAL ou EDITOR. Par défaut, l'éditeur vi sera utilisé.

En tant qu'administrateur du système, il est possible de modifier les paramètres pour n'importe quel utilisateur. Pour cela, il faut préciser le login de l'utilisateur avec l'option `-u`. Il est recommandé d'utiliser également l'option `-u` si l'on a effectué un **su**, car la commande **crontab** peut ne pas pouvoir déterminer l'utilisateur qui l'a appelé.

Le format des fichiers `crontab` est suffisamment riche pour permettre de spécifier avec finesse les conditions d'exécution des opérations programmées. En général, le début du fichier contient la définition de variables d'environnement utilisées par `crontab`. La suite du fichier est réservée aux commandes programmées. Chaque programmation est réalisée sur une ligne du fichier `crontab`. Les lignes contiennent 5 champs spécifiant la date et l'heure à laquelle la commande doit être exécutée, un nom d'utilisateur éventuel et la commande elle-même. Le nom d'utilisateur ne doit être spécifié que dans le fichier `/etc/crontab`, qui définit les commandes du système. Il spécifie alors au nom de quel utilisateur la commande doit être exécutée. Pour les fichiers crontab spécifiques à chaque utilisateur, il n'est bien entendu pas nécessaire d'indiquer ce nom.

Les 5 champs de la partie décrivant la date d'exécution de la commande fournissent respectivement les informations suivantes :

- les minutes (comprises entre 0 et 59) ;
- les heures (comprises entre 0 et 23) ;
- le jour dans le mois (compris entre 0 et 31) ;
- le mois (comprise entre 0 et 12, ou indiqué par les trois premières lettres du nom du mois en anglais) ;
- le jour dans la semaine (compris entre 0 et 7, ou indiqué par les trois premières lettres du nom du jour en anglais).

Les numéros de mois 0 et 12 correspondent à Janvier, et les numéros de jours 0 et 7 correspondent au Dimanche.

La commande sera exécutée à chaque fois que le jour, le mois, l'heure et les minutes du système correspondront avec ces 5 champs. Il suffit que l'une des spécifications du jour corresponde pour que la commande soit exécutée (c'est à dire qu'elle est exécutée une fois pour le jour du mois et une fois pour le jour de la semaine si ces deux champs sont spécifiés).

Il est possible d'utiliser un intervalle de valeurs pour chacun de ces champs, en indiquant la première et la deuxième valeur, séparées d'un tiret. Il est également possible de faire une liste de valeurs et d'intervalles, en séparant chaque donnée par une virgule. Si l'on veut spécifier toutes les valeurs possibles pour un champ, on peut utiliser le caractère '`*`'. Enfin, il est possible d'indiquer que la commande doit être exécutée toutes les n valeurs pour chaque champ. Pour cela, il suffit de faire suivre le champ d'une barre de oblique de division ('`/`') et du nombre `n`. Ainsi, si l'on trouve l'expression « */3 » pour les heures, la commande sera exécutée toutes les trois heures.

La spécification de la commande doit être faite sur une seule ligne. Le caractère de pourcentage (’%’) a une signification spéciale, sauf s’il est précédé d’un backslash (’\’). Les données qui suivent le premier pourcentage sont passées telles quelles dans l’entrée standard de la commande. Les caractères pourcentages suivants sont interprétés comme des saut de lignes (donc une validation). Ainsi, la commande suivante :

```
rm -i file.txt%y%
```

permet de supprimer le fichier `file.txt` et de répondre ’`y`’ à la commande **rm**. Le caractère ’`y`’ est passé ici dans le flux d’entrée standard de **rm**.

Comme vous pouvez le voir, le fichier `/etc/crontab` du système permet de programmer des opérations périodiques, comme les sauvegardes, la destruction des fichiers temporaires, ou toute autre tâche de maintenance. Ne vous étonnez donc pas si votre ordinateur semble s’activer tout seul régulièrement, à heure fixe (par exemple, sur le coup de 11 heures ou minuit). C’est le fonctionnement normal de votre système, qui s’occupe de toutes les tâches ménagères qu’il s’est réservé pour une heure où normalement tout le monde dort…

Les utilisateurs peuvent également définir leur propre `crontab` pour effectuer les opérations périodiques qu’il désirent. Par exemple, ils peuvent programmer une commande qui leur rappellera un rendez-vous.

6.9. Configuration des terminaux virtuels

Un terminal est, comme son nom l’indique, un équipement qui se trouve au bout d’une connexion à un ordinateur et qui permet de travailler sur l’ordinateur. Un terminal comprend généralement un clavier et un écran (graphique pour les terminaux X), et parfois une souris. Initialement, les terminaux étaient des périphériques passifs connectés sur un port série de l’ordinateur. Cette architecture permettait de partager une même unité centrale avec plusieurs utilisateurs. De nos jours, la plupart des terminaux sont simplement d’autres ordinateurs du réseau, qui se connectent à l’aide d’un programme que l’on appelle « émulateur de terminal ». L’émulateur de terminal se contentait de simuler un terminal réel et permettait de se connecter à toute machine gérant des terminaux clients. Du point de vue du système, tous les utilisateurs sont connectés via un terminal, qu’ils soient physiques ou émulés.

Les connexions locales sont donc réalisées par l’intermédiaire d’un terminal local, la *console*. La console est tout simplement le périphérique prenant en charge le clavier et l’écran. Ce périphérique est géré directement par le noyau, et émule un modèle de terminal standard, le VT102. En fait, la console n’est utilisée directement en tant que terminal de login qu’en mode mono utilisateur (c’est à dire en mode maintenance). En mode multi-utilisateur, la console est partagée entre plusieurs terminaux dits « virtuels », parce qu’ils simulent la présence de plusieurs terminaux sur la même console. Ces terminaux virtuels se partagent le clavier et l’écran, mais seulement un de ces terminaux peut accéder à la console à chaque instant : celui qui traite les caractères saisis au clavier et qui réalise l’affichage. Pour pouvoir passer d’un terminal virtuel à un autre, il faut utiliser une séquence de touches spéciale, comme par exemple `ALT+Fn` (ou `CTRL+ALT+Fn` si vous êtes sous XWindow), où `Fn` est l’une des

touches de fonction du clavier. Si l'on utilise cette combinaison avec la touche `F1`, on accédera au premier terminal virtuel. Avec la touche `F2`, ce sera le deuxième, avec `F3`, le troisième, etc...

Il est donc possible de simuler la présence de plusieurs écrans et claviers sur une même machine, ce qui est très pratique lorsque l'on commence à lancer plusieurs programmes en même temps (ceci nécessite évidemment de se connecter plusieurs fois sur des terminaux différents). Bien entendu, la configuration du clavier et la police de caractères utilisée sont communs à tous les terminaux virtuels, puisqu'ils utilisent tous la même console.

La plupart des distributions utilisent au moins quatre terminaux virtuels, plus éventuellement un terminal pour le serveur X. Mais il est possible de définir d'autres terminaux, qu'ils soient graphiques ou non. Nous allons maintenant voir comment modifier le nombre de terminaux virtuels disponibles. L'utilisation des terminaux avec XWindow sera traitée dans le chapitre traitant de la configuration de XWindow.

Chaque terminal virtuel utilise un fichier spécial de périphérique du répertoire `/dev/`. Le nom de ce fichier commence toujours par « tty » et est complété par le numéro du terminal. Ainsi, le fichier spécial de périphérique `/dev/tty1` correspond au premier terminal virtuel, accessible avec la combinaison de touches `CTRL+ALT+F1`, le fichier spécial de périphérique `/dev/tty2` correspond au deuxième terminal virtuel, etc... Linux peut gérer jusqu'à 64 terminaux virtuels, cependant, il est nécessaire de définir d'autres combinaisons de touches pour accéder aux terminaux 13 et suivants (puisqu'il n'existe que 12 touches de fonctions). Il serait possible d'utiliser les combinaisons de touches `ALT+DROITE` et `ALT+GAUCHE` pour les atteindre, mais d'une part ce ne serait pas très pratique, et d'autre part, vous ne devriez pas avoir besoin de plus de quatre ou cinq terminaux virtuels. Nous n'utiliserons donc ici que les douze premiers terminaux virtuels.

Les terminaux virtuels sont créés par le noyau à la demande, dès qu'un processus cherche à y accéder. Ainsi, le système n'alloue les ressources utilisées pour la gestion de ces terminaux que lorsque cela est nécessaire. Les terminaux peuvent être créés par différents processus, et leur emploi n'est pas restreint à la simple connexion des utilisateurs. Par exemple, il est possible d'afficher un message sur un terminal simplement en écrivant dans son fichier spécial de périphérique. Ainsi, si vous tapez la commande suivante sous le compte root :

```
echo Coucou > /dev/tty11
```

la chaîne de caractères « Coucou » devrait apparaître sur le terminal virtuel 11.

En général, les terminaux virtuels sont utilisés soit pour afficher les messages du système, soit pour permettre aux utilisateurs de se connecter, soit pour XWindow. Les terminaux peuvent donc être attribués à différents programmes, selon l'emploi qu'on leur réserve. Il faudra cependant bien prendre garde au fait que les terminaux ne sont pas partageables entre tous les processus. Ainsi, on ne devra pas essayer de lancer un serveur X sur un terminal utilisé par un processus de connexion en mode texte.

Pour créer des terminaux de login, il suffit de demander au système de lancer les processus de connexion sur chaque terminal désiré. Ce travail est à la charge du processus fondamental du système : *init*. La définition des terminaux de login se fait donc dans dans le fichier de configuration `/etc/inittab`. Si vous regardez le contenu de ce fichiers, vous trouverez quelques lignes similaires à la suivante :

```
1:2345:respawn:/sbin/getty 9600 tty1 linux
2:2345:respawn:/sbin/getty 9600 tty2 linux
etc...
```

Ces lignes indiquent à init que plusieurs processus *getty* doivent être lancés sur les différents terminaux virtuels. Le programme getty est le programme qui vous demande votre nom d'utilisateur sur les terminaux virtuels. Plus précisément, getty initialise le terminal, demande le nom d'utilisateur et lance le programme *login* en lui fournissant le nom saisi, afin que celui-ci demande le mot de passe de cet utilisateur.

Si vous désirez rajouter des terminaux de login à votre configuration, vous devrez donc rajouter des lignes de ce genre dans le fichier `/etc/inittab`. En fait, ces lignes sont constituées de quatre champs :

- le premier champ est le numéro de la ligne. En pratique, ce numéro doit être celui du terminal virtuel qui sera utilisé par getty ;
- le champs suivant (« `2345` ») contient les numéros des niveaux d'exécution dans lesquels cette ligne est valide. Ces numéros doivent être spécifiés les uns à la suite des autres, sans séparateur. Dans l'exemple donné ci-dessus, ces lignes sont valides dans les niveaux d'exécution 2 à 5 compris, c'est à dire tous les niveaux d'exécution classiques ;
- le troisième champs indique à init des options pour le lancement du programme getty. Dans l'exemple donné ci-dessus, le programme getty doit être relancé immédiatement dès qu'il se termine. Ceci est le comportement désiré, puisque la terminaison de getty correspond à la déconnexion de l'utilisateur courant, et qu'il faut laisser la possibilité de se reconnecter aux suivants ;
- le dernier champ indique le programme à lancer et ses options. Dans notre cas, il s'agit de `/sbin/getty`. Les options indiquées sont la vitesse de la ligne de communication utilisée, le nom du terminal sur lequel getty doit travailler et son type (ici, ce sont des terminaux de type « `linux` »).

Vous pouvez bien entendu vous baser sur les lignes existantes pour en créer de nouvelles. L'opération est très simple : il suffit de renuméroter les lignes et les terminaux virtuels utilisés. Prenez garde cependant à ne pas affecter à getty un terminal utilisé par XWindow. Il est recommandé d'effectuer ces modifications dans le niveau d'exécution 2 pour ne pas être gêné par XWindow.

Une fois les modifications ajoutées, vous pourrez demander à init de relire son fichier de configuration avec la commande suivante :

```
init Q
```

Dès lors, les nouveaux terminaux sont prêts à être utilisés.

6.10. Configuration de la console

Comme nous venons de le voir, tous les terminaux virtuels utilisent la même console. La suite logique des opérations est donc de voir comment on réalise la configuration de celle-ci… Nous allons donc

voir dans ce chapitre la manière de paramétrer le clavier et l'affichage du texte à l'écran.

Pour cela, il est nécessaire de bien comprendre les mécanismes mis en ½uvre pour le traitement des codes émis par le clavier d'une part, et pour l'affichage des symboles des lettres à partir de leur code d'autre part. Ces mécanismes sont relativement évolués et complexes, mais permettent de paramétrer avec précision la disposition des touches du clavier, le comportement des applications et l'allure des symboles affichés à l'écran.

Le plus simple pour comprendre le fonctionnement de la console est encore de voir les différentes étapes entrant en ligne de compte de l'appui sur une touche jusqu'à l'action correspondante à cet appui. Notez bien que cette action n'est pas forcément l'affichage d'une lettre à l'écran : ceci dépend de l'application qui a traité l'événement correspondant à l'appui sur la touche. Il est toutefois nécessaire de présenter auparavant quelques notions sur la manière dont les caractères sont représentés dans les programmes.

6.10.1. Pages de codes et Unicode

De part leur nature de calculateurs, les ordinateurs n'ont jamais été conçus pour manipuler nativement du texte. Ceci signifie qu'ils n'ont aucune notion de caractère ou de symbole : pour eux, tout est numérique. Par conséquent, il a fallu trouver le moyen de représenter les caractères humains sous une forme numérique, afin de pouvoir réaliser des programmes de manipulation de textes. Cette représentation est effectuée en associant à chaque caractère un numéro donné, et en travaillant directement sur ces numéros. Par exemple, le caractère 'A' est classiquement représenté par le nombre 65, la lettre 'B' par le nombre 66, etc… L'opération consistant à effectuer cette association sur chaque caractère d'un texte constitue ce que l'on appelle l'*encodage* de ce texte. Initialement, plusieurs manière de réaliser cet encodage ont été inventées, mais l'un des standard les plus utilisé est l'encodage ASCII. Lorsqu'il a été créé, l'encodage ASCII codait les caractères sur 7 bits. Depuis, il a été étendu pour utiliser 8 bits, ce qui fait que chaque caractère est dorénavant codé sur un octet. Ainsi, il est possible de représenter 256 caractères différents avec un octet, ce qui est suffisant pour toutes les langues occidentales.

Cependant, le standard ASCII initial ne spécifiait que l'encodage des caractères utilisés en anglais, tout simplement parce que les Américains parlent anglais. Évidemment, ceci ne convenait pas pour les pays qui utilisent des lettres accentuées, c'est à dire pour quasiment tout le monde. Il a donc fallu définir d'autres conventions que celle initialement utilisée, afin d'associer des codes numériques aux caractères utilisés par les autres pays. Ces conventions constitue ce que l'on appelle une *page de codes*. Chaque pays est donc susceptible d'utiliser une page de codes qui lui est spécifique. Par exemple, la page de codes 437 représente l'encodage utilisé aux États-Unis, et la 850 celle utilisée en France.

Historiquement, les pages de codes n'ont pas été immédiatement standardisées, ce qui a conduit à la prolifération de pages différentes et parfois incompatibles. Ainsi, les pages de codes utilisées en France sont les pages de codes 850 sous DOS, Windows 1252 sous Windows, ISO 8859-1 et ISO 8859-15 sous Unix. Ces deux dernières constituent la norme actuelle, et sont celles qui doivent être utilisées de préférence. La norme ISO 8859-1 est également connue sous le nom *latin-1*, et la norme ISO 8859-15 sous le nom *latin-0*. La norme latin-0 est une extension de la latin-1, qui ne prévoyait pas le codage de certains caractères européens (comme le o e dans l'o français) et le symbole de l'Euro.

Le défaut majeur de l'ASCII et de ses dérivés est de travailler sur des nombres à 8 bits. Si le nombre de 256 caractères différents convient pour la plupart des pays occidentaux, ce n'est pas le cas de quelques autres pays, qui utilisent un nombre de symboles de très loin supérieur à 256. C'est notamment le cas du Japon et de la Chine, qui ne peuvent pas encoder tous leurs idéogrammes sur des nombres à 8 bits. Il a donc fallu introduire d'autres types d'encodages, plus riches, permettant de satisfaire aux besoins de tout le monde.

Après des tentatives infructueuses d'encodages à taille variable (un ou plusieurs octets selon le caractère codé), Unicode a été introduit et normalisé sous le nom *ISO 10646*. Unicode est une convention de codage universelle des caractères, qui utilise pour cela des nombres 32 bits (il existe également une version plus ancienne qui n'utilise que 16 bits). Chaque caractère est représenté par un nombre et un seul, comme pour l'ASCII. Cependant, avec ses 16 ou 32 bits, le jeu de caractères Unicode est suffisamment large pour coder tous les caractères de toutes les langues du monde. Bien entendu, tous les codes ne sont pas utilisés, et le jeu de caractère Unicode est discontinu. Pour des raisons de compatibilité, les 256 premiers caractères Unicode sont les mêmes que ceux définis dans la norme ISO 8859-1 (ce qui rend malheureusement non conforme la norme ISO 8859-15, plus complète). Les autres caractères sont affectés à d'autres plages de codes, qui sont parfaitement définies. Ainsi, l'utilisation d'Unicode permettra, à terme, de n'avoir plus qu'une seule page de codes pour tous les pays.

Malheureusement, Unicode est une évolution relativement récente, et la plupart des programmes travaillent encore avec des caractères 8 bits, ce qui rend l'utilisation d'Unicode prématurée. Linux, quant à lui, est capable de gérer l'Unicode. Cependant, pour des raisons d'économie de place, il ne l'utilise pas directement. Il préfère en effet utiliser l'encodage *UTF-8* (abréviation de l'anglais « Unicode Tranfer Format »). Cet encodage est un encodage à taille variable, qui permet d'encoder les caractères Unicode avec un, deux ou trois octets selon leur emplacement dans la page de codes Unicode.

6.10.2. Principe de fonctionnement du clavier

En général, les claviers envoient une série de codes à l'unité centrale lorsque l'on appuie sur une touche. Certaines touches génèrent un seul code, d'autres peuvent en produire jusqu'à une dizaine. Ces codes, que l'on appelle *scancode*, sont récupérés par le driver du clavier dans le noyau de Linux, et constituent le début du traitement des saisies clavier. Les scancodes permettent normalement de déterminer avec certitude l'événement qui s'est produit, pour peu que l'on connaisse parfaitement le type de clavier utilisé. Malheureusement, ils sont spécifiques à chaque modèle de clavier, et il est difficilement concevable pour un programme de prendre en charge les particularités de tous les claviers existants. De plus, qui peut prévoir aujourd'hui combien de touches les claviers du futur auront, et quels scancodes ceux-ci utiliseront ?

Linux effectue donc un travail d'uniformisation en interprétant les scancodes et en les traduisant en d'autre codes, les *keycodes*. Ces codes sont toujours les mêmes, quel que soit le clavier utilisé. Les keycodes simplifient le traitement des données provenant du clavier en définissant un code de touche unique à chacune des touches du clavier. Les keycodes sont également souvent appelés les *codes de touches virtuelles*, car ils correspondent aux scancodes d'un clavier virtuel uniforme et commun à toutes les plates-formes. La gestion des événements claviers par l'intermédiaire des keycodes est donc

beaucoup plus aisée, car il n'y a plus à se soucier ici que de ce clavier virtuel. La correspondance entre les scancodes, donc les touches physiques, et les keycodes, ou codes de touches virtuelles, est définie dans les drivers de claviers. La plupart des claviers courants sont gérés et en général peu de personnes ont besoin de modifier ce type d'information. Toutefois, afin de permettre l'intégration des claviers futurs, il est possible de compléter la table de conversion des scancodes en codes de touches virtuelles.

La manipulation des scancodes est une opération relativement compliquée et peu commode, à cause de la complexité des séquences de scancodes envoyées par les claviers. C'est pour cela qu'une autre association, qui permet de définir le comportement à obtenir pour chaque combinaison de codes de touches virtuelles, a été introduite. Cette correspondance est décrite dans un fichier que l'on appelle couramment le plan de clavier (ou *keymap* en anglais). Les keymaps contiennent donc, pour chaque touche ou combinaison de touches virtuelles utilisée, la définition d'une action à effectuer ou d'un code de caractère à renvoyer. Ces codes sont renvoyés en ASCII, codés selon la page de code définie dans le plan de clavier. Cependant, certaines touches ne sont pas associées à une lettre ASCII, et ne peuvent donc pas être représentées par des codes simples. C'est le cas, par exemple, des touches du curseur, de la touche `Entrée` et des touches de suppression du clavier. Ces touches sont donc signalées aux programmes à l'aide de séquences de codes appelées les *codes d'échappement*. Les codes d'échappement sont en réalité des séquences de codes ASCII dont le premier est le code du caractère d'échappement `Escape` (dont le numéro est 27, quelle que soit la page de code utilisée). Ces séquences sont donc utilisées typiquement pour signaler aux programmes qui les lisent qu'un traitement particulier doit être effectué, comme par exemple le déplacement du curseur.

Les programmes peuvent peuvent travailler à n'importe lequel des trois niveaux de traitement que l'on vient de décrire. Les programmes qui récupèrent directement les scancodes travaillent en mode *raw* (ce qui signifie en anglais que les données sont « brutes »). Ils n'utilisent donc pas la traduction en codes de touches virtuelles et doivent nécessairement connaître les caractéristiques physiques des claviers qu'ils veulent supporter. C'est le cas par exemple des serveurs X. Les programmes qui utilisent avec les keycodes travaillent dans le mode nommé *medium raw* ou *keycode*. Cependant, la plupart des programmes travaillent avec les codes issus du plan de clavier, qui sont normalisés et font bénéficier d'une plus grande portabilité (y compris avec d'autres systèmes Unix que Linux). Dans ce cas, on dit qu'ils travaillent en mode ASCII, ou *xlate* (abréviation de l'anglais « translate », ce qui signifie « traduction »).

> **Note:** En fait, la console peut également travailler en Unicode, dans le mode UTF-8. Ce mode permet de travailler directement en Unicode, si vous le désirez. Cependant, vous devrez vous assurer dans ce cas que tous les programmes que vous utilisez sont capables de fonctionner avec un encodage à taille variable.

Dans tous les cas, les programmes lisent les données provenant du clavier sur la console, par l'intermédiaire du fichier spécial de périphérique `/dev/console`. En fait, ces programmes ne savent pas qu'ils lisent les données du clavier local. Pour eux, ces données semblent provenir d'un terminal VT102, et ils traitent ces données comme celles provenant de n'importe quel terminal. Ceci signifie que ces données semblent provenir d'une ligne de communication, à laquelle le terminal local est connecté. Et qui dit ligne de communication, dit paramètres de transmission des informations sur la ligne ! Les caractères issus de la keymaps peuvent donc subir un traitement supplémentaire, en fonction des paramètres de la ligne de communication (virtuelle) utilisée par la console. Cependant, il

n'est en général pas nécessaire de modifier ces paramètres, puisque la ligne de communication utilisée est bien entendue idéale (quitte à être virtuel, autant être idéal, non ?).

Gloups !? Que c'est compliqué ! C'est effectivement la réflexion que vous êtes en droit de vous faire. Cependant, si l'on prend un peu de distance par rapport à ce mécanisme en trois passe, on comprend son intérêt. Premièrement, la contrainte initiale est la complexité des scancodes. Sur ce point, il n'y a rien à dire. Grâce aux keycodes, il n'est plus nécessaire de se soucier du modèle de clavier utilisé. Ensuite, grâce aux keymaps, il est possible de faire l'association entre les keycodes et les lettres écrites sur les touches du clavier. Ainsi, la disposition du clavier, ainsi que les raccourcis claviers, peuvent être complètement paramétrés. Enfin, les applications qui gèrent les touches spéciales du clavier peuvent interpréter les codes d'échappement, qui sont ceux renvoyés par le terminal de la console. Dans tous les cas, les programmes qui lisent les données du clavier considèrent que ces données proviennent d'un terminal classique. Ils n'ont donc pas besoin de faire des hypothèses sur l'origine des données, et leur programmation en est d'autant plus simple.

Passons maintenant aux travaux pratiques. La commande **kbd_mode** permet de déterminer le mode de fonctionnement du clavier. Si vous essayez à partir d'une console, vous verrez que le clavier est en mode ASCII. Inversement, si vous tapez cette commande à partir d'une session X, vous constaterez que XWindow utilise le clavier en mode raw, et gère donc les scancodes lui-même. Vous pouvez également visualiser les codes renvoyés dans les trois modes de fonctionnement du clavier avec la commande **showkey**. Lancée avec l'option `-s`, elle permet d'afficher les scancodes lors de l'appui sur chaque touche. Vous pourrez constater que toutes les touches ne renvoient pas le même nombre de codes. Vous pourrez utiliser l'option `-k` pour visualiser les keycodes renvoyés par le noyau lors de l'appui et du relâchement des touches. Enfin, si vous utilisez l'option `-a`, vous verrez les codes de touches ASCII. Vous pourrez constater que les touches spéciales renvoient des séquences de codes d'échappement.

> **Note:** Notez également que certaines touches renvoient des caractères de contrôle (notés `^A` à `^Z`). Ces caractères correspondent en fait aux codes ASCII allant de 1 à 26 et ne sont pas des séquences d'échappement. Ils peuvent également être générés avec les combinaisons de touches `CTRL+A` à `CTRL+Z`. Si vous ne me croyez pas, tapez la commande **ls** dans une console et tapez la combinaison de touches `CTRL+M` (celle affectée à la touche `Entrée`). Vous verrez, c'est comme si vous aviez tapé `Entrée` !
>
> Le programme showkey ne peut pas être lancé sous XWindow, parce que celui-ci gère lui-même le clavier. showkey se termine de lui-même au bout de dix secondes après la dernière frappe, ou lors de l'appui de la séquence de touches `CTRL+D` s'il est en mode ASCII.

6.10.3. Principe de fonctionnement de l'écran de la console

Il est temps maintenant de passer au mécanisme d'affichage des caractères sur l'écran. Comme vous avez pu le voir dans les paragraphes précédent, le fait d'appuyer sur une touche ne fait rien d'autre que de fournir des codes numériques aux programmes. Contrairement à ce que vous pourriez penser, les codes de ces touches ne sont pas transmis directement à l'écran (heureusement, comment taperiez-vous votre mot de passe à l'abri des regards indiscrets sinon ?). En fait, ce sont les applications qui,

en fonction des codes qu'elles lisent sur la console, vont effectuer une action. Certaines de ces actions peuvent nécessiter l'affichage de résultats sur le moniteur. Pour réaliser ceci, les données à afficher sont écrites dans le fichier spécial de périphérique de la console.

Comme nous l'avons déjà vu, la console de Linux se comporte comme un terminal local. Ceci implique que les applications qui désirent écrire des données doivent les envoyer par la ligne de communication via laquelle elles sont connectées à ce terminal. Tout comme les scancodes du clavier, les caractères envoyés à la console par les applications subissent donc le traitement imposé par les paramètres de la ligne de communication virtuelle utilisée. Bien entendu, ce traitement est encore une fois minimal, puisque cette ligne est idéale Notez que les programmes n'en savent absolument rien, car pour eux, tout se passe comme s'ils écrivaient sur un terminal réel. Quoi qu'il en soit, le flux de caractères arrive au niveau du driver de la console après ce traitement de base.

Comme nous l'avons déjà dit plus haut, la console Linux émule un terminal VT102, et recherche donc les codes de contrôle et les séquences d'échappement de ces consoles. Ces codes d'échappements sont classiquement utilisés pour effectuer des opérations sur la console, comme par exemple le déplacement du curseur, la couleur et les attributs des caractères affichés, l'effacement de l'écran, etc... Pour information, tous les codes de la console sont décrits dans la page de manuel `console_codes`.

Les codes qui ne sont pas reconnus comme étant des codes d'échappement poursuivent leur traitement en tant que caractères normaux. Le driver de la console doit donc déterminer quel caractère doit être imprimé à l'écran pour chaque code fourni par l'application. Ceci n'est pas une opération facile, parce que les polices de caractères, qui contiennent la définition de la représentation graphique des caractères, n'utilisent pas forcément la page de code active dans le système. Ceci signifie que les caractères définis dans la police n'apparaissent pas forcément dans le même ordre que celui de la page de code, et pour cause : une police peut parfaitement être utilisable dans plusieurs pays différents ! Ceci complique donc un peu plus les choses, puisqu'il faut utiliser une table de conversion entre la page de code du texte à afficher et l'encodage propre de la police.

Afin de réaliser cette conversion, le driver de la console Linux comment donc par convertir tous les caractères qu'il reçoit en « Unicode ». Si la console est en mode UTF-8, cette opération est immédiate, car l'UTF-8 code les caractères exactement comme Unicode. En revanche, si la console est en mode ASCII, cette opération nécessite l'emploi d'une table de conversion. Linux dispose de quatre tables de conversions :

- la première table permet de faire la conversion entre la page de code ISO 8859-1 et Unicode (cette conversion est, par définition, immédiate) ;
- la deuxième table permet de faire la conversion entre les codes graphiques des terminaux VT100 et Unicode ;
- la troisième table permet d'effectuer la conversion entre la page de code 437 et Unicode ;
- et la quatrième et dernière table est réservée à l'usage personnel de l'utilisateur.

La dernière table est très utile, car elle permet de définir une nouvelle table de conversion si nécessaire. Par défaut, elle convertit les codes reçus en codes Unicode spéciaux, qui représentent les codes utilisés par la police de caractères chargée. Ainsi, lorsque cette table est utilisée, les codes reçus par la console seront utilisés directement pour localiser les caractères dans la police de caractère. Ceci

ne peut fonctionner que si les applications utilisent le même encodage que la police de caractères courante.

Linux dispose de deux jeux de caractères, nommés respectivement *G0* et *G1*, qui permettent de sélectionner la table de conversion à utiliser. Par défaut, ces jeux de caractères pointent respectivement sur la première et la deuxième table de conversion, mais ceci peut être modifié à l'aide de codes d'échappement. Seul un jeu de caractère est actif à un moment donné. Par défaut, il s'agit du jeu de caractères G0, ce qui implique que la table de conversion utilisée par défaut est la table ISO 8859-1 vers Unicode. Il est également possible de changer de jeu de caractères, à l'aide d'un code de contrôle. Ce mécanisme ne sera pas décrit plus en détail ici, car il ne nous sera pas utile.

> **Note:** En fait, la recherche des codes de contrôle est effectuée a posteriori, après la conversion en Unicode.

Dans tous les cas, les codes envoyés par l'application sont donc convertis en Unicode (16 bits). Ils peuvent donc être reconvertis aisément dans l'encodage utilisé par la police de caractère chargée en mémoire vidéo. Par défaut, cette opération est triviale, et l'encodage cible est donc l'encodage ISO 8859-1, sauf si la quatrième table de conversion a été utilisée. Dans ce cas en effet, l'encodage cible est le même encodage que celui utilisé par l'application (tout se passe donc comme si rien ne s'était passé). Enfin, la dernière opération est tout simplement l'écriture du code dans la mémoire vidéo. Ce code est utilisé par la carte graphique comme index dans la police de caractère, et le caractère ainsi localisé est affiché.

Gloups !? Que c'est compliqué ! Ça ne va pas recommencer ? Eh si ! Mais encore une fois, ce mécanisme permet de rendre indépendant les applications des polices de caractères. Chacun peut utiliser l'encodage qu'il désire, et grâce à un petit passage en Unicode, tous les caractères finissent par être représentés par le bon symbole à l'écran... En fait, le passage par Unicode donne la possibilité de définir les tables de conversion des polices uniquement par rapport à Unicode, et non par rapport à tous les encodages possibles que les applications peuvent utiliser. Ceci permet donc l'utilisation de polices de caractères diverses et variées, sans avoir à modifier les applications.

6.10.4. Configuration du clavier

Bien que les distributions modernes fournissent les outils nécessaires à la configuration correcte du clavier, vous aurez peut-être à intervenir pour le personnaliser selon vos propres désirs. La configuration du clavier comprend la définition des scancodes, la définition du plan de clavier, et le réglage de la vitesse de répétition et de la touche de verrou numérique.

6.10.4.1. Définition de scancodes

La définition des scancodes est une opération qui nécessite une bonne connaissance du fonctionnement du clavier des PC. Heureusement, rares sont les personnes qui disposent de claviers non standard, c'est à dire en pratique de claviers disposant plus de 105 touches. Notez que Linux reconnaît parfaitement les touches « Windows » qui ont été introduites par Microsoft, et il n'y a donc pas lieu de définir leurs scancodes. Cependant, les claviers les plus récents disposent de touches « Internet » ou

« Multimedia », qui ne sont pas encore reconnues par Linux en standard. L'utilisation de ces touches se traduit donc simplement par un message d'erreur dans les traces systèmes. Si vous disposez d'un tel clavier, et que vous désirez utiliser ces touches, vous allez devoir les définir dans la table des scancodes du noyau.

Pour réaliser cette opération, il faut avant tout déterminer les scancodes envoyés par le clavier lors de l'appui sur la touche à définir. Comme nous l'avons indiqué plus haut, les scancodes peuvent être visualisés avec l'option `-s` de **showkey** :

```
showkey -s
```

Les scancodes sont exprimés en hexadécimal, c'est à dire en base 16. Dans cette base, les lettres A à F représentent les chiffres manquant à la base 10, soit les chiffres représentant les valeurs 10, 11, 12, 13, 14 et 15.

En général, le clavier envoie un scancode lors de l'appui sur une touche, puis le même scancode augmenté de 128 lors de son relâchement. Le nombre de 128 provient du fait que le bit de poids fort du scancode est mis à un lors du relâchement de la touche. Ainsi, la touche 'A' des claviers français renvoie le scancode 16 lors de l'appui, soit 0x10 en hexadécimal, et le scancode 144, soit 0x90, lors du relâchement (notez que la valeur 128 se note 0x80 en hexadécimal, et que l'on a bien 0x90 = 0x10 + 0x80).

Cependant, certaines touches renvoient des scancodes plus complexes. Ces touches sont généralement des touches qui ont été ajoutées après la définition des premiers claviers PC, et qui ont sensiblement le même rôle qu'une touche déjà présente. C'est exactement le cas de la touche `CTRL` de droite par rapport à la touche `CTRL` de gauche. Pour cette raison, les scancodes de ces touches sont les mêmes que ceux de certaines touches « classiques », mais ils sont précédés du préfixe 0xE0, qui indique qu'il s'agit d'une touche étendue. Par exemple, la touche `CTRL` de gauche (c'est à dire la première touche de contrôle qui ait été utilisée) utilise le scancode 0x1D à l'appui, et le scancode 0x9D au relâchement. La touche `CTRL` de droite renvoie donc la séquence suivante à l'appui :

```
0xE0 0x1D
```

et la séquence suivante au relâchement :

```
0xE0 0x9D
```

L'intérêt de procéder ainsi est que les vieux programmes, incapables de gérer les codes de touches étendues, ignoraient purement et simplement le code 0xE0. Ainsi, ils confondaient les deux touches `CTRL`, ce qui est le comportement désiré.

D'autres touches étendues utilisent des séquences de scancodes variables. Par exemple, les touches du curseur (entre la touche `Entrée` et le pavé numérique) ont été ajoutées pour simplifier le déplacement du curseur à l'écran. Initialement, on n'utilisait que les touches du pavé numérique, en combinaison de la touche `Majuscule` de droite si le verrou numérique était enclenché. Par conséquent, la séquence de scancodes générée lors de l'utilisation de la touche `Flèche Haute` du pavé numérique était : `0x48`

si le verrou numérique n'était pas enclenché, et : `0x2A 0x48` sinon (soit le scancode de la touche `Majuscule` de gauche suivi du scancode de la touche `8` du pavé numérique). Lors du relâchement, la séquence était inversée (et augmentée de 128) : `0xC8` ou : `0xC8 0xAA` selon l'état du verrou numérique. La touche `Flèche Haute` étendue renvoie donc deux séquences de scancodes différents selon l'état du verrou numérique : `0xE0 0x48` ou : `0xE0 0x2A 0xE0 0x48`

Lors du relâchement, la séquence suivante complexe suivante est émise : `0xE0 0xC8` ou : `0xE0 0xC8 0xE0 0xAA` selon l'état du verrou numérique. Notez que l'état du verrou numérique est maintenu en interne par l'électronique du clavier, indépendamment de l'état de la diode lumineuse du verrou numérique. Celui-ci peut en effet être enclenché sans que la diode soit allumée, si le programme de gestion du clavier ne synchronise pas le clavier et sa diode.

Enfin, pour couronner le tout, certaines touches spéciales utilisent une séquence de scancodes spécifiques. Par exemple, la touche `Pause` ne revoie que la séquence 0xE1 0x1D 0x45 0xE1 0x9D 0xC5 à l'appui. Elle utilise donc le préfixe 0xE1 au lieu de 0xE0, et simule l'appui et le relâchement immédiat des touches `CTRL + Verrou Numérique` (ce qui était la manière de provoquer la pause avant l'introduction de la touche `Pause`).

Comme vous pouvez le constater, tout ceci est très compliqué. Heureusement, le noyau fait le ménage pour vous et élimine les touches suivantes :

- les touches non étendues sont traitées immédiatement (scancodes 0x01 à 0x58) ;
- la touche `Pause` est gérée automatiquement ;
- les codes 0x2A et 0xAA de la touche `Majuscule` de droite sont automatiquement éliminés.

Grâce à ce traitement, les seuls scancodes que vous manipulerez sont les scancodes simples non connus, et les scancodes précédés du préfixe 0xE0.

L'association d'un scancode inconnu à un keycode se fait à l'aide de l'utilitaire **setkeycodes**. Sa syntaxe est la suivante :

```
setkeycodes scancode keycode
```

où `scancode` est le numéro du scancode, exprimé en hexadécimal et sans le préfixe « 0x », et `keycode` est bien entendu le numéro de keycode à attribuer à cette touche. Les numéros de scancodes peuvent être simples (par exemple 48) ou précédés de leur préfixe (par exemple E048). Les numéros de keycodes doivent être exprimés en décimal, et doivent être compris entre 1 et 127.

Par exemple, mon clavier dispose de 12 touches « Multimédia » supplémentaires, dont les scancodes ne sont pas reconnues par le noyau. La touche `Arrêt du son` générant les scancodes suivants à l'appui :

```
0xE0 0x20
```

je peux lui associer le keycode 119 avec la commande suivante :

```
setkeycodes E020 119
```

Il va de soi qu'il faut s'assurer que chaque keycode n'est utilisé qu'une fois (à moins, bien entendu, de vouloir que plusieurs touches aient le même comportement). Pour cela, vous aurez sans doute besoin de voir les correspondances entre scancodes et keycodes. La commande suivante vous donnera la liste de ces correspondances :

```
getkeycodes
```

Les keycodes sont présentés à raison de huit par ligne. Le scancode de la touche décrite par le premier élément de chaque ligne est affiché en tête de la ligne. Les autres scancodes peuvent être déduits du numéro de la colonne dans laquelle se trouve chaque keycode.

6.10.4.2. Définition d'un plan de clavier

Comme nous l'avons expliqué dans les paragraphes précédents, les plans de clavier définissent les associations entre les keycodes et les caractères ou les séquences d'échappement renvoyés par le clavier. Les plans de claviers permettent également de définir des séquences de composition, afin d'obtenir de nouveaux caractères en composant les actions de plusieurs touches.

La définition d'un plan de clavier est donc constituée de trois parties. La première partie décrit les symboles accessibles pour certaines touches du clavier. Par exemple, les symboles accessibles pour la touche `A` peuvent être les caractères 'a', 'A','æ' et 'Æ', à l'aide des touches de majuscule et `AltGr`. La deuxième partie permet d'affecter des chaînes de caractères à certaines touches particulières. Cette partie peut contenir par exemple la définition de touches de raccourci pour les commandes les plus utilisées. En pratique cependant, elles sont souvent utilisées pour définir des séquences d'échappement pour les touches de fonction et les touches du curseur, séquences qui seront interprétées ensuite par les applications. Enfin, la dernière partie permet de définir les compositions de touches pour obtenir des caractères qui n'auraient été accessibles autrement que par des combinaisons de touches compliquées.

Pour chaque touche normale du clavier, il est possible de définir jusqu'à 256 symboles différents. Ces symboles sont sélectionnés grâce à des touches de modification du comportement de base des autres touches, comme par exemple la touche `Ctrl` ou la touche `Alt Gr`. Linux est capable de gérer des combinaisons de touches faisant intervenir jusqu'à huit touches de modification différentes en plus de la touche affectée (remarquez qu'avec dix doigts, on peut encore y parvenir !). Ces touches sont récapitulées dans le tableau suivant :

Touche de modification	Valeur	Description
Shift	1	Touches de passage en majuscule (situées au dessus des touches `Ctrl sur les claviers français).`
altgr	2	Touche `Alt Gr, située à droite de la barre d'espacement.`
control	4	Touches `Ctrl, situées aux extrémités inférieures du clavier.`

Touche de modification	Valeur	Description
alt	8	Touche `Alt, située à gauche de la barre d'espacement.`
shiftl	16	Touche de majuscule de gauche.
shiftr	32	Touche de majuscule de droite.
ctrll	64	Touche `Ctrl de gauche.`
ctrlr	128	Touche `Ctrl de droite.`

Chaque touche dispose d'un code numérique qui est une puissance de deux. Lorsqu'elles sont utilisées dans une combinaison avec une touche donnée, la somme de ces valeurs est calculée pour déterminer le numéro du symbole dans la liste des symboles associés à la touche. Comme on peut le voir, toutes les valeurs possibles allant de 0 à 255 sont réalisables selon la combinaison de touches utilisée, ce qui permet donc de définir effectivement 256 symboles différents pour chaque touche du clavier.

Cependant, les touches `Shift` et `Ctrl` sont utilisées plusieurs fois dans le tableau, précisément trois fois, la première ne faisant pas de distinction entre la touche de droite et la touche de gauche. En pratique donc, toutes les combinaisons ne sont pas réalisables. Mais en réalité, les touches de contrôle du clavier sont des touches comme les autres, et peuvent être placées n'importe où dans le plan de clavier. Il est donc parfaitement possible de réaliser une distinction entre les touches `Majuscule`, `Majuscule Droite` et `Majuscule Gauche` (et de même pour les touches `Ctrl`). Par exemple, on peut associer la touche `Majuscule` à la touche `Echap`, ce qui permet de faire la distinction entre les trois variantes de touches de majuscules… Heureusement, il n'est pas nécessaire d'aller jusque là. Les plans de claviers n'utilisent en pratique que les quatre premières touches de contrôle, qui sont celles que vous connaissez.

La définition des symboles accessibles pour une touche utilise la syntaxe suivant :

```
keycode = symbole symbole symbole ...
```

où `keycode` est le keycode qui identifie la touche en question, et `symbole` est un des symboles acceptés dans les plans de claviers. Vous pourrez obtenir la liste des symboles utilisés grâce à la commande suivante :

```
dumpkeys --long-info
```

Vous pourrez voir par exemple les symboles `A`, `B`, `C`, etc… qui représentent les lettres classiques, ainsi que des symboles spéciaux comme `exclamdown`, `hyphen`, `cedilla`, etc… pour les lettres non alphabétiques. Il existe également des symboles pour les touches spéciales comme `Backspace`, `Delete`, `Escape`. Enfin, le symbole `VoidSymbol` permet de signaler l'absence de symbole pour la combinaison de touches considérée.

En théorie, il faut définir la liste des 256 symboles accessibles pour toutes les combinaisons de touches imaginables. Le premier symbole est donc le symbole obtenu par appui direct de la touche, le deuxième est celui obtenu par la combinaison `Majuscule + Touche`, le troisième celui de la combi-

naison `AltGr + Touche`, le quatrième par `Majuscule + AltGr + Touche`, etc... Évidemment, il est très fastidieux de définir ces 256 possibilités. Pour simplifier le format des plans de clavier, il est possible de ne spécifier que les combinaisons utilisées, ce qui réduit en pratique à quatre colonnes de symboles un plan de clavier français. De plus, les derniers symboles sont facultatifs s'ils sont tous `VoidSymbol`. Vous pouvez indiquer au début du plan de clavier les valeurs des combinaisons de touches de modifications qui seront effectivement utilisées avec la syntaxe suivante :

```
keymaps valeurs
```

où `valeurs` est la liste des valeurs ou des plages de valeurs des combinaisons de touches utilisées. Les éléments de cette liste sont séparés par des virgules, et les plages de valeurs sont indiquées par leurs premières et dernières valeurs, séparées par un tiret.

Vous pouvez également utiliser une autre syntaxe, qui permet de ne modifier que les symboles associés à certaines combinaison de touches. Cette syntaxe est très utile lorsque vous ne désirez modifier que quelques affectations de touches :

```
modificateur keycode = symbole
```

Dans cette syntaxe, `modificateur` est la liste des modificateurs devant intervenir dans la combinaison de touches, et `keycode` et `symbole` sont toujours le keycode de la touche et le symbole à générer. Les modificateurs autorisés sont les noms des touches de modification indiquées dans le tableau ci-dessus, plus le modificateur `plain`, qui signifie qu'aucune touche de modification n'est utilisée. Par exemple, la ligne suivante :

```
plain keycode 16 = q
```

permet d'affecter le symbole `q` à la touche `A` de votre clavier, et la ligne :

```
alt keycode 30 = a
```

permet d'affecter le symbole `a` à la combinaison de touches `Alt + Q` (n'essayez surtout pas ces deux exemples, vous deviendriez fou).

> **Note:** Par défaut, les symboles utilisables dans les plans de clavier sont les symboles du jeu de carctères ISO 8859-1. D'autres encodages sont utilisables, mais celui-ci convient parfaitement pour un clavier français.

Comme nous l'avons dit plus haut, la deuxième partie d'un plan de clavier permet d'affecter des chaînes de caractères à certaines touches. Ceci est facilement réalisable, avec la syntaxe suivante :

```
string symbole = "chaîne"
```

où `symbole` est le nom d'un des symboles affecté précédemment à une touche, et `chaîne` est une chaîne de caractères. Les chaînes de caractères étant délimitées par des guillemets anglais (« " »),

ceux-ci ne peuvent pas être utilisés en tant que caractères de ces chaînes. Pour résoudre ce problème, on peut utiliser un antislash comme caractère d'échappement :

```
\"
```

Vous pouvez également spécifier des caractères directement à l'aide de leur valeur en base huit, à l'aide de la syntaxe suivante :

```
\0valeur
```

où `valeur` est la valeur du caractère. Bien entendu, l'antislash étant utilisé comme caractère d'échappement, il doit lui-même être précédé d'un caractère d'échappement si l'on désire l'utiliser dans une chaîne de caractère :

```
\\
```

Ainsi, si l'on veut faire en sorte que la touche `F4` affiche la chaîne de caractère « `Coucou` », il suffit d'utiliser la ligne suivante dans le plan de clavier :

```
string F4 = "Coucou"
```

Bien entendu, les chaînes de caractères les plus utiles sont celles qui définissent les séquences d'échappement pour les touches spéciales. Par exemple, la définition de la touche `Page Haut` est la suivante :

```
string PageUp = "\033[5~"
```

Vous pouvez reconnaître ces séquences d'échappement à la présence du caractère octal `033`, soit `27` en décimal, qui n'est rien d'autre que le caractère `Escape` dans le jeu de caractères ISO 8859-1. Ne vous inquiétez par pour l'instant de la forme apparemment compliquée de ces séquences d'échappement, nous verrons plus loin comment elles sont utilisées.

Enfin, les plans de claviers contiennent la définition des compositions. Les compositions de touches sont accessibles à l'aide d'une touche spéciale dite de *composition*, dont le symbole est `Compose` dans le plan de clavier (il est donc nécessaire que ce symbole soit affecté à une des touches du clavier pour utiliser les compositions). Lorsque l'on appuie sur la touche de composition, le clavier attend deux autres touches, dont les caractères serviront de base au caractère composé. Par exemple, les caractères '`^`' et '`a`' donne le caractère composé '`â`'.

Les compositions utilisent la syntaxe suivante :

```
compose 'caractère' 'caractère' to 'résultat'
```

où `caractère` est un des deux caractères à composer et `résultat` est le caractère résultant de la composition. Notez bien que les compositions se font sur les caractères, et pas sur les touches. Elles

sont donc actives quel que soit la méthode obtenue pour générer ces caractères (combinaison de touches ou touches simples).

> **Note:** Le noyau définit la plupart des compositions nécessaires pour le jeu de caractères ISO 8859-1. Ne les cherchez donc pas en vain dans votre plan de clavier, vous ne les trouverez pas forcément. . .
>
> D'une manière générale, vous pourrez trouver de plus amples renseignements concernant les plans de clavier dans la page de man `keymaps`.

Il est long et difficile de créer un plan de clavier de toute pièce. Heureusement, encore une fois, cette description n'était que didactique. Vous n'aurez certainement même pas à modifier votre plan de clavier (sauf si vous voulez faire des expériences), car des plans de claviers prédéfinis sont fournis avec toutes les distributions. Ces plans de claviers sont en général stockés dans le répertoire `/usr/lib/kbd/keymap`. Pour les claviers de PC français, je vous conseille tout particulièrement le plan de clavier `fr-latin0.map.gz` du sous-répertoire `i386/azerty`. Ce plan de clavier permet d'accéder à toutes les touches utilisées par le français, plus la plupart des touches utilisées en Europe. Il utilise l'encodage ISO 8859-15, afin d'avoir accès aux rares caractères manquants dans l'encodage ISO 8859-1.

Le chargement d'un plan de clavier est très simple. Il suffit de taper la commande suivante sous le compte root :

```
loadkeys keymap
```

où `keymap` est le nom du fichier contenant le plan de clavier à utiliser. Si l'on ne spécifie pas de fichier, **loadkey** attend que vous tapiez la spécification des touches directement. Vous pourrez valider en générant le caractère `EOF` (abréviation de « End Of File », ce qui signifie « Fin de fichier ») à l'aide de la combinaison de touche `CTRL + D`, ou abandonner avec le classique `CTRL + C`.

En général, les distributions utilisent la commande **loadkey** dans les fichiers d'initialisation du système, pour charger le plan de clavier que vous avez choisi à l'installation dès le démarrage. Il est recommandé de ne modifier le plan de clavier courant que par l'intermédiaire du programme de configuration du système fourni avec votre distribution.

6.10.4.3. Modification des paramètres du clavier

Il nous faut encore voir deux petits utilitaires permettant de fixer quelques paramètres du clavier pour finir notre tour d'horizon de ce périphérique. Le premier de ces paramètres permet de manipuler l'état des diodes lumineuses qui indiquent si le verrou numérique, le verrou des majuscules ou l'arrêt défilement sont actifs. Cet utilitaire se nomme logiquement **setleds**, et il permet non seulement de fixer l'état de ces diodes, mais également l'état du clavier. Rappelons en effet que les diodes peuvent être désynchronisées par rapport à l'état réel du clavier. Ce cas de figure peut se produire si l'on passe d'un terminal X à un terminal en mode console, tout simplement parce que les serveurs X ne savent pas comment déterminer l'état du clavier sur une console afin de le restaurer.

La syntaxe de **setleds** est la suivante :

```
setleds -D num | caps | scroll
```

où `num`, `caps` et `scroll` sont des options permettant de préciser respectivement l'état des verrous numériques, majuscules et défilement. Ces options sont de la forme `+num`, `-num` pour le verrou numérique, et de forme similaire pour les autres verrous. Comme leur syntaxe l'indique, elles permettent d'enclencher ou de désenclencher l'état des verrous correspondants du clavier.

L'état des verrous du clavier est conservé par chaque terminal virtuel, indépendamment les uns des autres. Cette commande devra donc être répétée pour tous les terminaux virtuels utilisés. L'option `-D` permet de rendre les changements permanents pour le terminal sélectionné. Ainsi, si ce terminal est réinitialisé, la modification sera conservée. C'est en général l'effet désiré. Sachez qu'il est également possible d'effectuer une modification temporaire, et de ne modifier que l'affichage des diodes (sans changer l'état des verrous).

Il est recommandé de placer les quelques lignes suivantes dans les fichiers d'initialisation de votre système afin d'activer le verrou numérique pour tous les terminaux, si votre distribution ne vous permet pas de le faire par l'intermédiaire de son programme de configuration :

```
INITTTY=/dev/tty[1-12]
for tty in $INITTTY; do
setleds -D +num < $tty
done
```

Ces lignes appellent la commande **setleds** sur les terminaux virtuels 1 à 12.

Le deuxième utilitaire est **kbdrate**. Sa fonction est de fixer les paramètres de délai d'attente avant répétition lorsque l'on maintient une touche enfoncée, ainsi que la vitesse de répétition utilisée une fois que ce délai d'attente est dépassé. Sa syntaxe est elle aussi très simple :

```
kbdrate -s -r taux -d délai
```

où `taux` est le taux de répétition, et `délai` est le délai d'attente avant le début des répétitions de la touche enfoncée. L'option `-s` indique à **kbdrate** de ne pas afficher de messages pendant l'exécution de la commande. Les délais d'attentes spécifiés peuvent être de 250, 500, 750 ou 1000 millisecondes. Les taux de répétition utilisables vont de 2 à 30 caractères par secondes. Les claviers ne peuvent pas accepter toutes les valeurs intermédiaires pour le taux de répétition. Vous trouverez la liste exhaustive des valeurs admissibles dans la page de manuel `kbdrate`.

6.10.5. Choix de la police de caractères

Par défaut, la police de caractères utilisée par Linux est la police enregistrée dans la carte graphique. Cette police convient pour l'affichage des textes anglais, car elle utilise la page de code 437. Il est donc souhaitable, sinon recommandé, de changer de police de caractères à la fois pour bénéficier d'un encodage adapté à la langue française, et pour bénéficier d'améliorations esthétiques éventuelles.

Linux est fourni avec un certain nombre de fichiers de polices, qui sont normalement installées dans le répertoire `/usr/lib/kbd/consolefonts`. Comme vous pourrez le constater si vous y jetez un coup d'½il, ce répertoire contient un grand nombre de polices, utilisant divers encodages. Les polices les plus utiles pour un ordinateur situé en France sont sans doute les polices `iso01*`, `lat1-*` et `lat0-*`. Cette dernière police utilise l'encodage ISO 8859-15, et contient donc tous les symboles utilisés en Europe. Il est recommandé, mais non nécessaire, d'utiliser une des polices encodées en ISO 8859-1 ou en ISO 8859-15, avec un plan de clavier approprié. Par exemple, le plan de clavier `fr-latin0.map.gz` pourra être utilisé avec la police de caractères `lat0-16.psfu.gz`.

Les polices de caractères sont chargées aisément avec l'utilitaire **setfont**. La syntaxe de ce programme est très simple :

```
setfont police
```

où `police` est le fichier de police à charger. Si vous ne spécifiez aucun fichier de police, la police par défaut sera chargée.

Notez que les fichiers de polices peuvent utiliser un encodage particulier spécifique, qui ne correspond à aucune des tables de conversion prédéfinies du noyau. Pour ces polices, il est donc nécessaire de spécifier des tables de conversion qui leur sont propres. Ces tables permettent de convertir les codes des caractères reçus par la console en codes Unicode spéciaux. Tous ces codes appartiennent à une plage de codes Unicode réservée au système, et que Linux utilise pour définir les codes qui permettront d'accéder directement aux caractères de la police. L'association entre les codes de caractères de l'application et leur description dans la police de caractères est donc réalisée grâce à ces tables de conversion.

Vous pouvez charger une table de conversion spécifique à l'aide de l'utilitaire **mapscrn**. Cet utilitaire prend en ligne de commande un fichier contenant la table de conversion, et charge cette dernière dans la quatrième table de conversion du noyau (c'est à dire la table réservée à l'utilisateur) :

```
mapscrn fichier
```

(où `fichier` est le fichier contenant la description de la table de conversion).

Notez que l'utilitaire **mapscrn** charge la table de conversion, mais ne s'assure pas que le jeu de caractères courant utilise cette table. Par conséquent, vous aurez sans doute à utiliser les codes d'échappement `\033(K` ou `\033)K` pour faire pointer respectivement les jeux de caractères G0 et G1 sur la quatrième table. Ces codes d'échappement peuvent être saisis à l'aide de la commande **cat** :

```
cat
```

Il vous faudra alors taper sur la touche `Echap`, puis saisir « `(K` » ou « `)K` », et enfin signaler la fin de fichier avec un `CTRL + D`. Faites bien attention à ce que vous faîtes : en cas d'erreur, le jeu de caractères utilisé peut rendre l'écran totalement illisible. Si cela devait se produire, vous devrez taper à l'aveugle la commande **reset**. Cette table s'assure que le jeu de caractères actif est le jeu de caractères G0, et que celui-ci utilise la première table de conversion. Vous pourrez également recharger la police par défaut à l'aide de **setfont**.

Heureusement, la plupart des polices de caractères contiennent également la table de conversion à utiliser, et **setfont** effectue tout le travaille en une seule passe. Il charge la police de caractères dans la carte graphique, puis sa table de conversion dans le noyau, et s'assure enfin que le jeux de caractères G0 soit actif et utilise cette table. Ainsi, l'utilisation de **mapscrn** est devenue facultative.

La dernière étape dans la configuration de la police de caractères est le chargement de la table de conversion Unicode vers les indices des caractères dans la police. Cette table est essentielle, puisque c'est elle qui indique l'emplacement des caractères de la police à utiliser pour chaque code Unicode. Ces tables dépendent donc de l'encodage utilisé par la police de caractères.

Le chargement des tables de conversion est réalisée par le programme **loadunimap**. La syntaxe de ce dernier est la même que celle de **mapscrn** :

```
loadunimap fichier
```

où `fichier` est un fichier de conversion approprié. Vous trouverez de tels fichiers dans le répertoire `/usr/lib/kbd/consoletrans`. Les fichiers proposés permettent d'utiliser les polices de caractères encodées selon les encodages les plus courants.

Bien entendu, certaines polices disposent de leur propre table d'encodage. Encore une fois, l'utilisation de **loadunimap** est rendue facultative par **setfont**, qui se charge d'effectuer tout le travail. C'est notamment le cas pour les polices `fr-latin0.psfu.gz` et `fr-latin1.psfu.gz`, dont le « u » en fin d'extension indique la présence de la table de conversion Unicode.

6.10.6. Configuration de la ligne de communication

La configuration du driver de console ne concerne à proprement parler que les plans de clavier et les polices de caractères. Pour les applications, tout se passe comme si elles accédaient à une console comme les autres, donc par l'intermédiaire d'une ligne de communication. En général, les lignes de communication utilisées pour les terminaux sont des lignes série, c'est à dire des lignes sur lesquelles les données sont envoyés sous la forme de petits paquets de 5 à 8 bits, et contrôlés par d'autres bits éventuellement facultatifs (bit de stop et bit de parité). Il va de soi que toutes les lignes de communications ne sont pas identiques, et qu'un certain nombre de paramètres doivent être fixés.

Ces paramètres peuvent être fixés avec la commande **stty**. Leur nombre interdit ici une description exhaustive, d'autant plus que pour un terminal local, ils sont tous initialisés à des valeurs par défaut correctes, et vous n'aurez pas à les modifier. Vous pouvez toutefois consulter la page de manuel de **stty** pour de plus amples renseignements à ce sujet.

Si vous êtes curieux, vous pourrez obtenir à tout moment la liste des paramètres de la ligne de communication d'un terminal avec l'option `-a` de **stty** :

```
stty -a
```

Comme vous pourrez le constater, le nombre des options utilisées est assez impressionnant. Sont définis, entre autres, les paramètres de vitesse de communication de la ligne (option `speed`), la géométrie du terminal (options `rows` et `columns`), les caractères de contrôle affectés à un certain nombre d'ac-

tion relatifs à la ligne (comme par exemple, le caractère « ^Z », accessible par la combinaison de touches CTRL + Z, qui permet de suspendre l'exécution du programme utilisant la ligne, et le caractère « ^C », qui permet de le tuer), le format des données transférées sur la ligne (options parodd» pour la parité impaire, option cs8 pour des paquets 8 bits, etc...) et des options de gestion des caractères transférés (par exemple, echo fait en sorte que tout caractère entrant est immédiatement ré-émis sur la console, ce qui permet de voir ce que l'on tape).

Parmi ces paramètres, les plus intéressants sont sans doute ceux définissant les actions associées aux différentes touches de contrôle. Vous pourrez modifier ces paramètres avec la syntaxe suivante :

```
stty action caractère
```

où action est l'une des actions gérées par stty, comme par exemple susp pour suspendre le processus en cours, intr pour le tuer, etc... Remarquez que l'action kill n'a rien à voir avec la gestion des signaux des processus. Elle permet simplement d'effacer la ligne courante.

6.10.7. Description des terminaux

Le dernier maillon de la chaîne de gestion des caractères est bien entendu les applications. La plupart des applications sont capables de traiter les caractères simples comme les lettres de l'alphabet, les chiffres et la ponctuation. Cependant, il n'en va pas de même pour les codes d'échappement des terminaux. Comme nous l'avons vu, certaines touches sont programmées pour renvoyer des codes d'échappement dans le plan de clavier. Mais le driver de la console est également capable d'interpréter certains codes d'échappements, afin d'effectuer des actions spécifiques. Par exemple, les touches du curseur émettent des codes d'échappement spéciaux, et il est du ressort de chaque programme de reconnaître ces codes et d'agir en conséquence. De même, les programmes peuvent envoyer des codes d'échappement à la console pour déplacer le curseur affiché, effacer l'écran, effectuer un scrolling, etc...

Le malheur, c'est que les codes d'échappement utilisables différent selon les différents terminaux utilisés. Les terminaux les plus standards sont les terminaux VT100 et VT102 (la console Linux en faut partie). Cependant, les programmes ne peuvent pas savoir a priori quels codes d'échappement doivent être utilisés pour effectuer telle ou telle action.

C'est pour ces diverses raisons que les systèmes Unix disposent d'une base de donnée de définition des terminaux. Cette base de donnée contient tous les codes d'échappements que chaque type de terminal est capable de gérer, décrits d'une manière uniforme. Ainsi, les programmes désirant gérer correctement les terminaux n'ont qu'à consulter cette base de données pour interpréter et récupérer les codes d'échappement utilisés par le terminal. Historiquement, la définition des terminaux était réalisée dans le fichier de configuration /etc/termcap. Ce fichier est obsolète et a été remplacé par la base de données terminfo, que tous les programmes modernes doivent à présent utiliser. Cependant, le fichier de configuration /etc/termcap a été conservé par compatibilité avec les vieux programmes qui l'utilisent encore.

Le fichier termcap comprend une ligne pour chaque type de terminal décrit. Cette ligne est constituée d'un certain nombre de champs, séparés par des caractères deux points (':'). Le premier champ de

chaque ligne contient la description du terminal que cette ligne décrit. Les champs suivants sont des définitions des variables contenant les séquences d'échappement du terminal.

Les informations descriptive du terminal sont les suivantes :

- le nom abrégé du terminal ;
- le nom complet du terminal ;
- les autres noms sous lesquels le terminal est également connu ;
- la description détaillée du terminal (toujours en dernier).

Ces informations sont séparées les unes des autres par une barre verticale ('`|`').

Viennent ensuite les définitions des variables. Chaque variable est définie selon la syntaxe suivante :

```
variable=séquence
```

où `variable` est le nom de la variable, et `séquence` est la séquence d'échappement associée à cette variable. Notez que certaines variables ne prennent pas de paramètres, et que leur présence dans la description du terminal signale simplement un comportement particulier de celui-ci. Notez également que pour les variables numériques, le caractère d'égalité est remplacé par un caractère dièse ('`#`').

Un certain nombre de variables peuvent être définies pour chaque terminal. Ce sont ces variables qui sont utilisées par les programmes pour retrouver les séquences d'échappement du terminal. Par conséquent, les noms de ces variables sont fixés une fois pour toutes, et elles représentent toujours la même fonctionnalité. La liste des fonctionnalités est, encore une fois, très grande, et les variables utilisables sont listées exhaustivement dans la page de manuel `termcap`.

> **Note:** Il est évident que les lignes du fichier `/etc/termcap` peuvent devenir très longues. Il est donc possible de les étaler sur plusieurs lignes physiques, en insérant le caractère de continuation de ligne antislash ('`\`'). Après chaque retour à la ligne, il faut utiliser une indentation à l'aide du caractère de tabulation.

Vous trouverez ci-dessous un exemple de définition de terminal :

```
lx|linux|Console Linux:\
  :do=^J:co#80:li#25:cl=\E[H\E[J:sf=\ED:sb=\EM:\
  :le=^H:bs:am:cm=\E[%i%d;%dH:nd=\E[C:up=\E[A:\
  :ce=\E[K:cd=\E[J:so=\E[7m:se=\E[27m:us=\E[36m:ue=\E[m:\
  :md=\E[1m:mr=\E[7m:mb=\E[5m:me=\E[m:is=\E[1;25r\E[25;1H:\
  :ll=\E[1;25r\E[25;1H:al=\E[L:dc=\E[P:dl=\E[M:\
  :it#8:ku=\E[A:kd=\E[B:kr=\E[C:kl=\E[D:kb=^H:ti=\E[r\E[H:\
  :ho=\E[H:kP=\E[5~:kN=\E[6~:kH=\E[4~:kh=\E[1~:kD=\E[3~:kI=\E[2~:\
  :k1=\E[[A:k2=\E[[B:k3=\E[[C:k4=\E[[D:k5=\E[[E:k6=\E[17~:\
  :k7=\E[18~:k8=\E[19~:k9=\E[20~:k0=\E[21~:K1=\E[1~:K2=\E[5~:\
  :K4=\E[4~:K5=\E[6~:\
  :pt:sr=\EM:vt#3:xn:km:bl=^G:vi=\E[?25l:ve=\E[?25h:vs=\E[?25h:\
  :sc=\E7:rc=\E8:cs=\E[%i%d;%dr:\
```

```
:r1=\Ec:r2=\Ec:r3=\Ec:
```

Cette ligne permet de définir le terminal associé à la console Linux. Vous pourrez par exemple reconnaître le nombre de colonnes et de lignes (variables `co` et `li`), ainsi que les codes d'échappements associés aux touches du curseur (variable `ku`, `kd`, `kr` et `kl` respectivement pour les touches haut, bas, droite et gauche). La variable `cl` donne par exemple de séquence d'échappement utilisable pour effacer l'écran et faire revenir le curseur en haut à gauche (séquence d'échappement « `Esc [ H Esc [ J` »). Comme il l'a déjà été dit plus haut, la liste complète des variables peut être obtenue en consultant la page de manuel `termcap`, et ce fichier ne sera pas décrit plus en détail ici.

La base de données `terminfo` a été introduite pour combler certaines limitations du fichier `termcap`. Si le principe de fonctionnement est presque le même, les informations fournies tiennent compte des terminaux plus récents et de nouvelles fonctionnalités. Ceci signifie en pratique que de nouvelles variables ont été définies pour décrire les nouveaux terminaux. Inversement, certaines variables de termcap ont disparu parce qu'elles devenaient obsolètes, et ont été remplacées par des variables équivalentes de `terminfo`.

La principale différence entre `terminfo` et `termcap` est que la description des terminaux n'est plus stockée dans un fichier de configuration en mode texte. Toutes les données sont désormais stockées dans des fichiers binaires, qui peuvent être générés à l'aide du programme **tic**. Ces fichiers binaires sont usuellement placés dans les sous-répertoires du répertoire `/usr/share/terminfo/`. En fait, le nombre de fichiers de description est tellement grand qu'ils ont été regroupés par ordre alphabétique. Ainsi, le répertoire `/usr/share/terminfo` contient des sous-répertoires dont les noms sont les premières lettres des fichiers de description, et chaque fichier de description est situé dans le répertoire correspondant. Par exemple, le fichier de description des terminaux Linux se nomme tout simplement `linux`. Comme la première lettre de son nom est 'l', il est stocké dans le répertoire `/usr/share/terminfo/l/`, avec les descriptions de tous les autres terminaux dont le nom commence par 'l'.

> **Note:** En fait, les fichiers de description de terminaux peuvent être placés à un autre emplacement que l'emplacement par défaut. Les librairies utilisant les informations de `terminfo` cherchent en effet en premier dans le chemin référencé par la variable d'environnement TERMINFO, puis dans le répertoire `.terminfo` du répertoire de l'utilisateur. Ce n'est que si ces deux recherches échouent qu'elles utilisent les informations du répertoire par défaut.

Comme nous l'avons dit, les fichiers de description sont des fichiers binaires, qui ont été compilés à l'aide du compilateur **tic**. Cependant, vous pouvez parfaitement visualiser le contenu de ces fichiers ou comparer deux fichiers à l'aide de l'utilitaire **infocmp**. Par exemple, vous pouvez visualiser sous une forme lisible les informations du fichier de description des terminaux Linux avec la commande suivante :

```
infocmp linux
```

Vous obtiendrez certainement un résultat semblable à ceci :

```
#Reconstructed via infocmp from file: /usr/lib/terminfo/l/linux
linux|linux console,
        am, bce, eo, mir, msgr, xenl, xon,
        colors#8, it#8, pairs#64,
        acsc=+\020\,\021-\030.^Y0\333‘\004a\261f\370g\361h\260i\316j\\
331k\277l\3
        bel=^G, blink=\E[5m, bold=\E[1m, civis=\E[?25l,
        clear=\E[H\E[J, cnorm=\E[?25h, cr=^M,
        csr=\E[%i%p1%d;%p2%dr, cub1=^H, cud1=^J, cuf1=\E[C,
        cup=\E[%i%p1%d;%p2%dH, cuu1=\E[A, cvvis=\E[?25h,
        dch=\E[%p1%dP, dch1=\E[P, dim=\E[2m, dl=\E[%p1%dM,
        dl1=\E[M, ech=\E[%p1%dX, ed=\E[J, el=\E[K, el1=\E[1K,
        flash=\E[?5h\E[?5l$<200/>, home=\E[H, hpa=\E[%i%p1%dG,
        ht=^I, hts=\EH, ich=\E[%p1%d@, ich1=\E[@, il=\E[%p1%dL,
        il1=\E[L, ind=^J, invis=\E[8m, kb2=\E[G, kbs=\177, kcbt=\E[Z,
        kcub1=\E[D, kcud1=\E[B, kcuf1=\E[C, kcuu1=\E[A,
        kdch1=\E[3~, kend=\E[4~, kf1=\E[[A, kf10=\E[21~,
        kf11=\E[23~, kf12=\E[24~, kf13=\E[25~, kf14=\E[26~,
        kf15=\E[28~, kf16=\E[29~, kf17=\E[31~, kf18=\E[32~,
        kf19=\E[33~, kf2=\E[[B, kf20=\E[34~, kf3=\E[[C, kf4=\E[[D,
        kf5=\E[[E, kf6=\E[17~, kf7=\E[18~, kf8=\E[19~, kf9=\E[20~,
        khome=\E[1~, kich1=\E[2~, knp=\E[6~, kpp=\E[5~, kspd=^Z,
        nel=^M^J, op=\E[39;49m, rc=\E8, rev=\E[7m, ri=\EM,
        rmacs=\E[10m, rmir=\E[4l, rmpch=\E[10m, rmso=\E[27m,
        rmul=\E[24m, rs1=\Ec, sc=\E7, setab=\E[4%p1%dm,
        setaf=\E[3%p1%dm,
        sgr=\E[0;10%?%p1%t;7%;%?%p2%t;4%;%?%p3%t;7%;%?%p4%t;5%;\
%?%p5%t;2%;%?%p6%t;1%;%?%p7%t;8%;%?%p9%t;11%;m,
        sgr0=\E[m, smacs=\E[11m, smir=\E[4h, smpch=\E[11m,
        smso=\E[7m, smul=\E[4m, tbc=\E[3g, u6=\E[%i%d;%dR,
        u7=\E[6n, u8=\E[?6c, u9=\E[c, vpa=\E[%i%p1%dd,
```

Comme vous pouvez le constater, le format de ces informations est similaire à celui de celles qui sont enregistrées dans le fichier `/etc/termcap`. Les principales différences sont que les différents champs sont séparés par des virgules (’,’) au lieu du caractère deux points (’:’), et qu’il est possible de les répartir sur plusieurs lignes physiques (il est donc inutile d’utiliser le caractère antislash en fin de ligne physique pour indiquer la continuation de la ligne logique). De plus, le nom des variables utilisées dans les fichiers terminfo n’est pas le même a priori. Cependant, le principe d’utilisation de ces variables reste le même, chacune d’entre elle permet de définir une des fonctionnalité gérée par le terminal et de définir la séquence d’échappement nécessaire à l’obtention de cette fonctionnalité si nécessaire. Vous pourrez trouver la liste exhaustive des variables utilisables dans la page de manuel `terminfo`. Le format des fichiers de configuration de terminfo ne sera donc pas décrit plus en détail dans ce document.

Qu’ils utilisent des librairies basées sur `termcap` ou `terminfo`, les programmes doivent tous connaître le nom du terminal courant pour récupérer les informations qui permettent de l’utiliser dans ces bases de données. C’est précisément ce à quoi sert la variable d’environnement TERM. Cette variable contient en permanence le nom du terminal courant, que les programmes peuvent utiliser comme

index dans les bases de données `termcap` ou `terminfo`. Ainsi, si vous désirez travailler sur un terminal Linux, vous pourrez fixer le type de terminal correct avec la commande suivante :

```
export TERM=linux
```

La liste des noms de terminaux utilisables peut être obtenue qu'en lisant directement le fichier `termcap`. En revanche, cette liste peut être obtenue plus facilement pour `terminfo`, à l'aide de l'utilitaire **toe** (abréviation de l'anglais « Table Of terminfo Entry »).

Bien entendu, vous n'aurez à définir la valeur de TERM que si cette variable d'environnement n'est pas correctement définie, ce qui est très rare. En fait, ceci ne peut se produire que lors d'une connexion à distance sur un autre ordinateur, dont les terminaux ne sont pas de type Linux.

6.10.8. Paramétrage des applications

En théorie, la description du terminal courant fournie par les bases de données `termcap` ou `terminfo` est suffisante pour faire fonctionner correctement la plupart des applications. En particulier, les programmes qui utilisent les librairies de manipulation des terminaux sont capables d'utiliser ces informations. C'est le cas par exemple de tous les programmes qui utilisent les librairies « curses » ou « ncurses », car ces librairies utilisent la base de données `terminfo`.

Cependant, certaines applications utilisent un mécanisme plus simple (mais moins souple) pour gérer les terminaux. Pour ces applications, une configuration spécifique doit être effectuée, souvent pour chaque terminal, en précisant les codes d'échappement à utiliser dans leurs fichiers de configuration. On remarquera la présence du shell « bash », de l'éditeur « vi » et du programme de pagination « less » dans cette catégorie de logiciels. Comme ce sont les logiciels les plus couramment utilisés, il est indispensable d'indiquer comment les configurer pour une utilisation correcte.

6.10.8.1. Configuration du clavier pour la librairie readline

Un certain nombre d'applications, dont le shell « bash », utilisent la libraire GNU « readline » pour obtenir les lignes de commandes saisies par l'utilisateur. Tous ces programmes peuvent donc être configurés de la même manière, grâce au fichier de configuration de la librairie readline.

Cette librairie recherche son fichier de configuration en premier à l'emplacement indiqué par la variable d'environnement INPUTRC. Si cette variable d'environnement n'est pas définie, le fichier de configuration `~/.inputrc` est lu pour déterminer les séquences d'échappement à utiliser. Il est donc recommandé de créer un fichier de configuration général `/etc/inputrc` et de définir la variable d'environnement INPUTRC afin de définir des paramètres communs à tous les utilisateurs. Vous pouvez également recopier ce fichier dans les répertoires personnels de tous les utilisateurs sous le nom `.inputrc`.

Ainsi, si vous voulez gérer correctement le clavier français sous le shell bash, vous devrez vous assurer que les lignes suivantes sont placées dans votre fichier `inputrc` :

```
# Active la gestion du huitième bit des caractères
```

```
# (par exemple pour les caractères accentués) :
set meta-flag on
set input-meta on
set output-meta on
set convert-meta off

# Définit les codes d'échappement associés aux touches du curseur :
"\e[1~": beginning-of-line
"\e[3~": delete-char
"\e[4~": end-of-line
"\e[5~": history-search-backward
"\e[6~": history-search-forward
"\e[C": forward-char
"\e[D": backward-char
"\e[A": previous-history
"\e[B": next-history

# Redéfinit les codes d'échappement pour l'émulateur de terminal xterm :
$if term=xterm
"\e[1~": history-search-backward
"\e[4~": set-mark
"\e[H": beginning-of-line
"\e[F": end-of-line
"\eOH": beginning-of-line
"\eOF": end-of-line
$endif
```

Ce fichier commence par autoriser le traitement des caractères dont le bit « meta », c'est à dire le huitième bit, est positionné. C'est en particulier le cas de tous les caractères accentués dans les principales pages de codes ; ces options sont donc nécessaires pour pouvoir utiliser ces caractères dans le shell. La suite du fichier définit les actions associées à chaque code d'échappement du terminal. Le caractère d'échappement est représenté ici par la chaîne de caractères « `\e` ». Comme les codes d'échappement sont différents pour la console et pour les émulateurs de terminaux de XWindow, il est nécessaire de les redéfinir en fonction de la nature du terminal.

6.10.8.2. Configuration du clavier pour vi

L'éditeur en ligne de commande vi n'est pas réputé pour être agréable à utiliser. Ceci provient essentiellement de sa distinction entre les mode de commande, d'édition et de visualisation. Mais ceci n'est pas suffisant pour le rendre haïssable : la torture psychologique imposée par la mauvaise gestion du curseur vient souvent à bout des plus résistants. Heureusement, Linux fournit un clone nettement plus puissant et qui permet de résoudre ces problèmes : vim.

L'éditeur vim peut bien entendu être parfaitement compatible avec vi afin ne pas dérouter les habitués de vi. Mais il dispose en plus de fonctionnalités avancées qui en font un outil extrêmement configurable, et il est possible de le rendre nettement plus ergonomique que son ancêtre. Malheureusement, il faut reconnaître que la plupart des distributions fournissent un vim « brut de fonderie », ce qui fait

que seuls ceux qui se lancent dans la lecture de son aide peuvent parvenir à l'utiliser correctement. Les options qui sont proposées ici sont donc données simplement à titre indicatif, mais permettront peut-être de rendre vim un peu plus ergonomique.

Les options de configuration de vim sont stockées dans deux fichiers. Le premier fichier est le fichier de configuration commun à tous les utilisateurs, `vimrc`. Ce fichier peut être placé soit dans le répertoire `/etc/`, soit dans le répertoire `/usr/share/vim/`, selon votre distribution. Le deuxième fichier est le fichier de préférences personnelles de chaque utilisateur, et se nomme `.vimrc`. Il est normalement placé dans le répertoire personnel de l'utilisateur.

Plusieurs types d'options peuvent être indiquées dans ces fichiers de configuration. Les premières associent les actions à effectuer aux codes d'échappement générés par les touches du curseur. Les autres spécifient simplement le comportement de l'éditeur et le paramétrage de ses principales fonctionnalités. Vous trouverez ci-dessous les principales options que vous pourrez ajouter à votre fichier de configuration `vimrc`. Ces options rendront sans doute l'utilisation de vim beaucoup plus agréable.

```
" Exemple d'options de configuration pour vim.
" Notez que les commentaires sont introduits ici
" par des guillemets anglais (") et non
" par des dièses ('#'), contrairement à la plupart des fichiers
" de configuration.

" Éviter à tout pris la compatibilité avec vi, qui est insupportable :
set nocompatible


" Définir les touches du clavier :

" Gestion du curseur en mode de visualisation :
map ^[OA k
map ^[[A k
map ^[OB j
map ^[[B j
map ^[OD h
map ^[[D h
map ^? h
map ^H h
map ^[OC l
map ^[[C l
map ^[[2~ i
map ^[[3~ x
map ^[[1~ 0
map ^[OH 0
map ^[[H 0
map ^[[4~ $
map ^[OF $
map ^[[F $
```

```
map ^[[5~ ^B
map ^[[6~ ^F
map ^[[E ""
map ^[[G ""
map ^[OE ""

" Gestion du pavé numérique en mode de visualisation :
map ^[Oo :
map ^[Oj *
map ^[Om -
map ^[Ok +
map ^[Ol +
map ^[OM ^M
map ^[Ow 7
map ^[Ox 8
map ^[Oy 9
map ^[Ot 4
map ^[Ou 5
map ^[Ov 6
map ^[Oq 1
map ^[Or 2
map ^[Os 3
map ^[Op 0
map ^[On .

" Gestion du pavé numérique en mode insertion :
map! ^[Oo  :
map! ^[Oj  *
map! ^[Om  -
map! ^[Ok  +
map! ^[Ol  +
map! ^[OM  ^M
map! ^[Ow  7
map! ^[Ox  8
map! ^[Oy  9
map! ^[Ot  4
map! ^[Ou  5
map! ^[Ov  6
map! ^[Oq  1
map! ^[Or  2
map! ^[Os  3
map! ^[Op  0
map! ^[On  .

" Gestion du curseur dans les modes d’insertion et de commande :
map! ^[[H <Home>
map! ^[OH <Home>
map! ^[[F <End>
map! ^[OF <End>
```

```
map! ^[OA <Up>
map! ^[OB <Down>
map! ^[OC <Right>
map! ^[OD <Left>
map! ^[[3~ <Delete>
map! ^[OE  <Space>


" Définir les autres options globales :

" Paramétrage des touches Backspace et Delete :
set t_kb=^?
set t_kD=ESC[3~

" Faire en sorte que le "backspace" efface même les sauts de lignes :
set bs=2

" Utiliser l'indentation automatique dans les fichiers C et C++
" (pour les programmeurs) :
set cindent

" Utiliser la coloration syntaxique pour les principaux langages
" de programmation :
set background=dark
if &t_Co > 1
syntax on
endif

" Signaler les correspondances de parenthèses, accolades et crochets :
set showmatch

" Rappeler le mode de fonctionnement courant :
set showmode
```

Vous constaterez que certains caractères de contrôle sont utilisés dans ce fichier de configuration, dont le caractère de contrôle « `^[` », qui représente le caractère d'échappement. Ces caractères sont représentés avec la notation classique « `^C` », où « `C` » est la lettre à utiliser avec la touche `CTRL` pour obtenir ce caractère. Ces notations ne font que représenter les caractères de contrôle, et doivent être remplacées par les caractères qu'elles représentent dans le fichier de configuration. Pour saisir ces caractères spéciaux, vous devrez passer en mode insertion dans vi, puis utiliser la combinaison de touches `CTRL+V`. Ce raccourci permet d'indiquer à vi qu'il doit insérer les codes d'échappement directement issus du clavier, sans les interpréter. Vous pourrez alors taper la séquence de touches générant le caractère de contrôle ou la séquence d'échappement désirée. Par exemple, vous pourrez obtenir le caractère « `^H` » en tapant la combinaison de touches `CTRL+H`, et le caractère « `^?` » en appuyant sur la touche `Backspace` (retour arrière). Les caractères d'échappement peuvent être générés par la touche `Echap` ou directement par les touches du curseur.

> **Note:** En fait, vous pouvez également utiliser les chaînes de caractères « `\e` » et « `<Esc>` » pour représenter le caractère d'échappement. Mais certaines options ne fonctionnent pas avec ces notations, et je vous les déconseille.

Vous pouvez bien entendu ajouter d'autres options dans ce fichier de configuration. En pratique, toutes les options utilisables dans le mode de commande de vim peuvent être fixées définitivement dans ce fichier. Vous obtiendrez de l'aide sur ces options grâce à la commande « `:help` » de vim.

6.10.8.3. Configuration du clavier pour less

Le programme de pagination less est naturellement bien plus agréable à utiliser que son ancêtre more, puisqu'il permet de revenir sur les pages déjà consultées. Cependant, il peut le devenir encore plus si l'on s'arrange pour qu'il reconnaisse les touches du curseur.

Le programme less lit ses informations de configuration dans un fichier binaire dont l'emplacement est spécifié par la variable d'environnement LESSKEY. Si cette variable d'environnement n'est pas définie, less utilise le fichier de configuration `.less` du répertoire personnel de l'utilisateur. Ce fichier binaire contient les associations entre les séquences d'échappement du terminal et les actions effectuées par less. Il est généré par le compilateur **lesskey**, à partir d'un fichier de configuration textuel classique.

Ce fichier de configuration comprend plusieurs sections, qui permettent de définir les touches utilisées en mode de commande, les touches utilisées en mode d'édition de lignes (par exemple dans une commande de recherche), et enfin les variables d'environnements utilisées par less. Vous trouverez ci-dessous un exemple de fichier de configuration pour less. Vous trouverez de plus amples renseignements sur les actions qui peuvent être associées aux séquences d'échappement et aux caractères de contrôle dans la page de manuel `lesskey`.

```
# Exemple de fichier de configuration pour less

# Première section :

#command
\e[B    forw-line
\e[A    back-line
\e[6~   forw-scroll
\e[5~   back-scroll
\177    back-screen
^H      back-screen
\e[3~   back-screen
\e[2~   visual
\e[1~   goto-line
\eOH    goto-line
\e[4~   goto-end
\eOF    goto-end
\eOM    forw-line
```

```
# Deuxième section :

#line-edit
\177    backspace
^H      backspace
\e[3~   delete
\e[1~   home
\eOH    home
\e[4~   end
\eOF    end
\e[5~   up
\e[6~   down
\e[2~   insert
\e[E    insert
\e[G    insert
\eOE    insert
\eOo    insert :
\eOj    insert *
\eOm    insert -
\eOk    insert +
\eOl    insert +
\eOM    insert
\eOw    insert 7
\eOx    insert 8
\eOy    insert 9
\eOt    insert 4
\eOu    insert 5
\eOv    insert 6
\eOq    insert 1
\eOr    insert 2
\eOs    insert 3
\eOp    insert 0
\eOn    insert .


# Troisième section :

#env
LESSCHARSET=latin1
```

Conformément à un usage courant, les commentaires sont introduits par le caractère dièse (’#’) dans ce fichier de configuration. Cependant, certains commentaires sont utilisés pour identifier le début des trois sections du fichier. Il s’agit des commentaires « `#command` », « `#line-edit` » et « `#env` ». Il ne faut donc surtout pas supprimer ces commentaires dans votre fichier de configuration.

Encore une fois, le caractère d’échappement est symbolisé par la chaîne de caractère « `\e` ». De même, les caractères de contrôle sont représentés par la notation classique « `^C` », où `C` est la touche utilisée en combinaison avec `CTRL` pour générer ce caractère de contrôle. Notez que contrairement au

fichier de configuration `/etc/vimrc`, ces notations peuvent être utilisées directement dans le fichier de configuration de less.

Ce fichier de configuration pourra être compilé avec le programme **lesskey** afin de générer le fichier binaire utilisé par less. Pour cela, il faudra simplement utiliser la syntaxe suivante :

```
lesskey fichier
```

où `fichier` est le nom du fichier de configuration à compiler. Le fichier binaire généré est par défaut celui référencé par la variable d'environnement LESSKEY. Si cette variable n'est pas définie, un fichier `.less` sera créé dans le répertoire personnel de l'utilisateur.

6.10.9. Configuration de la souris

L'installation de la souris est une opération très simple à réaliser. La seule chose importante est de bien connaître les différents types de souris et de ne pas les confondre. Autrefois, la plupart des souris étaient des souris connectées sur le port série (souris sérielles). Aujourd'hui, ces souris se font de plus en plus rares, et le port de prédilection est le port PS/2. Ce port a été introduit par IBM dans ses ordinateurs PS/2 et est quelque peu plus pratique que le port série, car il définit une interface standard pour toutes les souris. De plus, il permet de dégager un des ports séries, ce qui simplifie la configuration des modems. Le port PS/2 ressemble au port clavier du même type, et en fait on peut se tromper et brancher la souris à la place du clavier et inversement. Il ne faut surtout pas confondre les souris PS/2 avec les souris Bus, qui ont été vendues autrefois et que l'on ne trouve quasiment plus à présent. Ces souris pouvaient se connecter sur des cartes spéciales, voire, parfois, sur la carte graphique.

Pour que l'installation de la souris se fasse correctement, il faut s'assurer que les options concernant la souris ait bien été définies dans la configuration du noyau de Linux. Ceci n'est pas nécessaire pour les souris série. Vous pouvez consulter le Chapitre 9 pour plus de détails sur la configuration du noyau. Lorsque cette étape est faite, il ne reste plus qu'à indiquer au programme de gestion de la souris à quel type de souris il a affaire. Normalement ce programme est gpm, il permet d'utiliser la souris en mode texte. La configuration de la souris pour XWindow sera vue dans le Chapitre 10.

La configuration de gpm se fait normalement par l'intermédiaire du programme de configuration de votre distribution. Lorsqu'on n'utilise pas XWindow, gpm est lancé automatiquement au démarrage. Il se peut que votre programme de configuration vous demande le type de souris à utiliser. Dans ce cas, il faut choisir le bon type, faute de quoi gpm ne fonctionnera pas correctement. Attention, si vous désirez utiliser une souris à molette (souris disposant d'une petite roue entre les deux boutons, et permettant de faire défiler le contenu des fenêtres), le type de souris à utiliser est « `imps2` » et non simplement « `ps2` ». Pour avoir la liste des types de souris gérés par **gpm**, il suffit de le lui demander avec la ligne de commande suivante :

```
gpm -t help
```

Normalement, vous aurez à utiliser gpm avec la ligne de commande suivante :

```
gpm -t type -m /dev/mouse
```

où `type` est le type de la souris que gpm doit utiliser et `/dev/mouse` est un lien vers le fichier spécial de périphérique gérant votre souris.

6.11. Modules du noyau

Les modules du noyau sont des librairies que l'on peut charger dans le noyau lorsque celui-ci a besoin d'une certaine fonctionnalité. Une fois chargés, les modules font partie intégrante du noyau et ajoutent leurs fonctions à celles existantes. Ces librairies sont normalement stockées dans le répertoire `/lib/modules/version/`, où `version` est le numéro de version du noyau pour lequel ces modules ont été créés. Beaucoup de fonctionnalités du noyau peuvent être configurées pour être utilisées sous la forme de modules. Certains drivers ne fonctionnent que sous la forme de modules, aussi faut-il savoir les manipuler.

Les modules peuvent être chargés manuellement à l'aide des commandes **insmod** et **modprobe**. **modprobe** est un peu plus évoluée, car elle gère les dépendances entre les modules et est capable de charger les modules utilisés par le module demandé. Leur utilisation est des plus simple :

```
insmod module
```

ou :

```
modprobe module
```

où `module` est le nom du module à charger.

En réalité, la commande **modprobe** appelle la commande **insmod** pour chaque module qui doit être chargé, dans l'ordre des dépendances des modules. De cette manière, chaque module peut être chargé sans problème, car toutes les fonctionnalités qu'il utilise sont déjà présentes dans le noyau lors de son chargement. La commande **modprobe** va chercher ces informations de dépendance dans le fichier `modules.dep`, situé dans le répertoire `/lib/module/version/`. Ce fichier utilise une syntaxe très simple, et spécifie pour chaque module les modules dont il dépend. Bien qu'il puisse parfaitement être écrit à la main, ceci nécessiterait d'avoir une connaissance approfondie des modules du noyau. C'est donc pour cela que l'outil **depmod** a été écrit. Cet outil permet de générer le fichier `modules.def` automatiquement, pourvu qu'on l'appelle avec l'option `-a` en ligne de commande :

```
depmod -a
```

Il faudra donc appeler cette commande après chaque installation ou suppression de module dans le système.

La liste de modules qui ont été chargés peut être obtenue aisément avec la commande **lsmod** :

```
lsmod
```

Enfin, la commande **rmmod** permet de décharger un module, avec la ligne de commande suivante :

```
rmmod module
```

La plupart des modules penvent prendre un certain nombre d'options lors de leur chargement. Ces options sont en quelque sorte l'équivalent des options que l'on communique au noyau lors du boot, pour préciser par exemple la configuration matérielle utilisée. Toutes ces options sont définies dans le fichier de configuration`/etc/modules.conf`. Lors du chargement d'un module, **modprobe** consulte ce fichier et passe les paramètres indiqués au module à charger.

Les options des modules sont spécifiées dans le fichier `module.conf` à l'aide du mot-clé `option`, suivi du nom du module pour lequel ces options sont définies, lui-même suivi des paramètres de chargement du module. La syntaxe est donc la suivante :

```
option module paramètres
```

Par exemple, il est probable que le driver du port parallèle soit fourni sous la forme de module avec votre distribution. Ceci permet de le charger à la demande, et de le configurer pour les différentes utilisations possibles d'un port parallèle : communication avec une imprimante parallèle, ou avec un lecteur ZIP, ou encore avec un lecteur de CD IDE, etc… Sur les ordinateurs de type PC, le nom du module gérant le port parallèle est « `parport_pc` ». Ce module peut prendre des paramètres de chargement permettant d'indiquer le port d'entrée/sortie à utiliser et la ligne d'interruption à utiliser. En général, le port utilisé est le port numéro 378h, et la ligne d'interruption est souvent la 7. Les options suivantes doivent donc être définies dans le ficher `module.conf` :

```
option parport_pc io=0x378 irq=7
```

Le fichier `module.conf` permet également de définir des opérations complémentaires qui doivent être effectuées lors du chargement et du déchargement d'un module. Ces opérations sont introduites par les mots-clés `pre-install`, `post-install`, `pre-remove` ou `post-remove`. Selon le mot-clé utilisé, elles sont exécutées respectivement avant et après le chargement du module, et avant et après son déchargement. Ces quatre mots-clés utilisent tous la même syntaxe :

```
xxx-install module opération
```

où `xxx-install` est le mot-clé indiquant les conditions d'application de l'opération, `module` est le nom du module pour lequel une opération complémentaire doit être effectuée, et `opération` est la ligne de commande de l'opération en question.

Ainsi, si le port parallèle est utilisé pour accéder à un périphérique nécessitant une opération d'initialisation quelconque, celle-ci peut être exécutée après le chargement du module à l'aide du mot-clé `post-install` :

```
post-install parport_pc initperiph
```

(en supposant que la commande `initperiph` permet d'initialiser le périphérique en question). Évidemment, ce type d'opération dépend du matériel connecté sur le port parallèle, mais le principe est là.

Dans la plupart des cas, ces options sont facultatives, car **modprobe** utilise un jeu de valeurs par défaut qui permettent d'utiliser les modules même si le fichier de configuration `modules.conf` est absent. Vous pouvez visualiser ces options en renommant le fichier de configuration `modules.conf` et en tapant la commande suivante :

```
modprobe -c
```

Cette commande vous permettra également de recréer un nouveau fichier de configuration si d'aventure vous perdiez celui fourni avec votre distribution. Notez cependant qu'il est recommandé d'adapter le fichier de configuration `modules.conf` afin d'obtenir des performances optimales. Les modifications que nous avons données en exemple ci-dessus pour le port parallèle permettent par exemple d'éviter que le noyau consulte en permanence le port pour savoir s'il est prêt à recevoir ou à fournir de nouvelles données pendant les impressions. Le gain de performance peut ainsi atteindre 10% quelle que soit la vitesse du processeur, ce qui est très loin d'être négligeable.

Il n'est normalement pas nécessaire d'utiliser ces commandes pour charger les modules. En effet, il est possible de faire en sorte que les modules soient chargés à la demande, lorsqu'une de leurs fonctionnalités est demandée au niveau du noyau. Pour cela, le noyau utilise lui-même la commande **modprobe** pour charger les modules, ce qui assure la gestion correcte des dépendances entre les modules. À chaque fois que le noyau a besoin d'une fonctionnalité qui n'est pas encore chargée, il effectue une requête à **modprobe** en lui demandant de charger le module manquant.

Le nom du module dont le noyau demande le chargement à **modprobe** dépend évidemment de la fonctionnalité demandée et ne correspond pas forcément à un nom de module existant. **modprobe** doit donc faire la correspondance entre le nom de module demandé par le noyau et le nom d'un module réel. C'est encore une fois dans le fichier de configuration `/etc/modules.conf` que cette correspondance est stockée. Par exemple, lorsque le noyau désire accéder au port parallèle, il utilise la commande suivante :

```
modprobe -k parport_lowlevel
```

où `parport_lowlevel` est le nom que le noyau utilise pour référencer le port parallèle. Ce nom est identique quelle que soit la plate-forme sur laquelle le noyau fonctionne, qu'il s'agisse d'un PC, d'un Mac ou d'une station de travail Sun. Sur les ordinateurs dont l'architecture est de type PC, le véritable module à utiliser est, comme on l'a vu, `parport_pc`. Il faut donc donner indiquer l'association entre les noms `parport_lowlevel` et `parport_pc` dans le fichier `modules.conf`.

Cette association est réalisée par la définition d'un certain nombre d'alias de modules réels. Chaque alias est introduit par le mot-clé `alias`, dont la syntaxe est donnée ci-dessous :

```
alias nom module
```

où `nom` est le nom de l'alias, et `module` est le nom du module réel. Lorsque la commande **modprobe** est appelée avec le nom d'un alias en paramètre, elle commence par rechercher le nom du module réel dans le fichier `modules.conf`, puis elle le charge en mémoire. Pour le port parallèle, on aura donc la définition suivante dans le fichier `modules.conf` :

```
alias parport_lowlevel parport_pc
```

L'option `-k` passée en paramètre à **modprobe** par le noyau n'est quant à elle utilisée que par le noyau. Elle permet de marquer le module comme étant automatiquement déchargeable, lorsque plus personne n'utilisera ses fonctionnalités. Cette technique permet de minimiser la consommation mémoire du noyau, sur les machines dont les ressources sont très limitées. Les modules qui sont chargés automatiquement par le noyau sont en effet marqués comme étant automatiquement déchargeables, et on peut effectivement les supprimer à l'aide de la commande suivante :

```
modprobe -r
```

Cette commande décharge les modules inutilisés automatiquement, dans l'ordre inverse de leur chargement. Cette commande respecte donc les dépendances entre modules, et seuls les modules inutilisés sont réellement déchargés. Il est recommandé de placer cette commande dans le fichier `/etc/crontab` de votre système, afin que les modules inutilisés soient déchargés à intervalle de temps régulier. Vous aurez donc sans doute à placer une ligne telle que celle-ci :

```
*/2 * * * * root modprobe -r
```

dans votre fichier `/etc/crontab`. Le déchargement des modules inutilisés aura lieu toutes les deux minutes.

Vous pouvez constater que le mécanisme d'alias permet de rendre le noyau indépendant des modules utilisés, puisque l'association entre le nom du module utilisé par le noyau et le module réel et ses paramètres est maintenu dans le fichier de configuration `modules.conf`. L'inconvénient de cette méthode est en revanche qu'il faut connaître les noms de modules utilisés par le noyau. Ces noms sont assez variables et dépendent de la fonctionnalité demandée. Toutefois, les noms utilisés par le noyau peuvent facilement être déterminés pour les modules gérant les fichiers spéciaux de périphériques. En effet, il est possible de construire un nom unique pour chaque fichier spécial de périphérique, qui pourra être utilisé par le noyau pour spécifier le module capable de gérer les fonctionnalités de ce fichier. La manière de construire ces noms varie selon que le fichier spécial de périphérique accédé, et selon qu'il s'agit d'un fichier réel ou d'un fichier du système de fichiers virtuel `/dev/` du noyau.

S'il s'agit d'un vrai fichier spécial de périphérique, il est parfaitement identifié par sa nature (disque, périphérique de type caractère, interface réseau, etc…) et ses deux codes majeur et mineur. Le noyau peut donc construire un nom à partir de ces informations. Ainsi, pour un fichier spécial de périphérique de type caractère, le nom généré par le noyau est en général de la forme « `char-major-xxxx` ». Pour les périphériques de type bloc, il est de la forme « `block-major-xxxx` ». Les caractères `xxxx` identifient le code majeur de ce périphérique, plus rarement son code mineur. Par exemple, pour les

cartes son, le nom de module `char-major-14` sera utilisé, parce que les cartes sons utilisent toutes le code majeur 14. Le code mineur n'est actuellement pas utilisé pour les cartes son.

Même si les noms de modules sont toujours construit plus ou moins selon ce schéma pour les fichiers spéciaux de périphériques réels, il reste difficile de déterminer le nom exact du module associé à un fichier spécial de périphérique. Par exemple, nous avons vu que le nom de module utilisé pour le port parallèle (accessible via le fichier spécial de périphérique `/dev/lp0`) est `parport_lowlevel`. De même, le nom utilisé pour les interfaces réseau de type Ethernet est `eth0`, `eth1`, etc... Il n'est donc pas toujours évident de savoir quel est le nom que le noyau utilisera. Le plus simple dans ce cas est encore de regarder dans la configuration utilisée par **modprobe** à l'aide de son option `-c`.

En revanche, si le fichier spécial de périphérique est un fichier virtuel, le mécanisme de nommage est complètement différent. En effet, le système de fichiers virtuel `/dev/` crée les fichiers spéciaux de périphériques à la demande, lorsque les gestionnaires de périphériques s'enregistrent au niveau du noyau. Il n'est donc pas possible de déterminer les codes majeurs et mineurs des fichiers spéciaux de périphériques si les modules qui les gèrent ne sont pas chargés, parce que ces fichiers n'existent pas encore d'une part, et parce que ces codes sont déterminés dynamiquement et ne sont donc jamais fixes d'autre part.

Le chargement des modules à la demande est en fait réalisé par le démon devfsd, que nous avons déjà décrit dans la Section 6.3. Outre la création des liens symboliques permettant d'accéder aux fichiers spéciaux de périphériques sous leurs noms classiques, ce démon est capable d'effectuer des actions diverses lorsqu'un événement se produit dans le système de fichiers virtuel `/dev/`. Toutes ces actions sont décrites dans le fichier de configuration `/etc/devfsd.conf`. En particulier, il est possible d'effectuer une action lorsqu'un programme recherche un fichier spécial de périphérique dans le système de fichiers, chose que tous les programmes qui utilisent les fichiers spéciaux de périphériques font, ne serait-ce que pour pouvoir les ouvrir. C'est lors de cette recherche que le démon devfsd va demander le chargement du module correspondant au fichier spécial de périphérique demandé.

Pour que ce mécanisme fonctionne, il faut avant tout l'activer. Ceci se fait simplement en ajoutant la ligne suivante dans le fichier `devfsd.conf` :

```
LOOKUP    .*    MODLOAD
```

qui signifie que le chargement dynamique de module (action « `MODLOAD` ») doit être effectuée lorsqu'une recherche (événement « LOOKUP ») a lieu dans le système de fichiers virtuel `/dev/`, et ce pour n'importe quel nom de fichiers spécial de périphérique (expression réguière « `.*` », qui représente n'importe quelle séquence de caractères).

Bien entendu, les noms de modules utilisés par le démon devfsd ne peuvent pas se baser sur les codes majeurs et mineurs du fichier spécial de périphérique pour les raisons que l'on a indiqué ci-dessus. Mais pourquoi faire compliqué quand on peut faire simple ? Le démon devfsd ne s'embarrasse pas de détails, et demande tout simplement le chargement du module dont le nom est exactement le chemin du fichier spécial de périphérique recherché ! Il suffit donc tout simplement de définir les alias correspondant dans le fichier `modules.conf` pour éventuellement trouver le bon nom de module à charger.

En fait, le démon devfsd utilise un autre fichier de configuration, le fichier `/etc/modules.devfs`, afin de simplifier l'écriture du fichier `modules.conf`. Ce fichier lui permet d'associer un ensemble de noms de fichiers spéciaux de périphériques à un seul module. Par exemple, tous les ports série d'une machine utilisent, en général, le même gestionnaire, bien qu'ils soient accessibles par l'intermédiaire des fichiers spéciaux de périphériques `/dev/ttySn`, où `n` est le numéro du port série (0 pour le port COM1, 1 pour le port COM2, etc...). Le fichier `modules.devfs` regroupe donc tous les noms de modules `/dev/ttyS*` en un seul nom de module, à savoir `serial`. C'est donc ce nom là qui sera communiqué à modprobe pour le chargement du module de gestion des ports série.

De manière générale, vous n'aurez pas à modifier les fichiers `devfsd.conf` et `modules.devfs`, car ceux généralement fournis avec les distributions conviennent parfaitement. En revanche, vous modifierez certainement le fichier `modules.conf`. Il s'agit là d'un opération délicate, car elle nécessite de bien connaître les mécanismes de nommage utilisés par le noyau, les noms des modules réels et leurs paramètres de configuration. En général, vous trouverez les informations sur les entrées à ajouter ou à modifier dans le fichier d'aide du module correspondant, que vous pourrez trouver dans les sources du noyau (dans le répertoire `/usr/src/linux/Documentation/`).

Quoi qu'il en soit, la modification de `modules.conf` peut générer de nouvelles dépendances entre les modules. Il est donc nécessaire d'exécuter la commande **depmod -a** après chaque modification de ce fichier.

6.12. Configuration des périphériques additionnels

L'architecture initiale des PC a très mal vieilli et souffre actuellement de défauts majeurs qui nuisent à leur développement. Sans parler des goulots d'étranglement sur les différents bus systèmes qui ne suivent plus les vitesses des processeurs actuels, la plupart des périphériques sont obligés de se restreindre à des protocoles de communication obsolètes pour conserver une compatibilité ascendante. L'une des limitations majeurs de cette architecture est le nombre incroyablement restreint de ligne d'interruption permettant à un périphérique de signaler au processeur qu'il a besoin de l'intervention de son gestionnaire pour poursuivre son travail, et la quasi absence des possibilités d'accès directs à la mémoire, qui impose l'utilisation du processeur pour réaliser de simples transferts de données entre les périphériques et la mémoire.

Ces ressources limitées, en plus de pénaliser les performances, posent des problèmes évidents d'extensibilité. En effet, qui dit ressource limité dit qu'il est impossible de connecter plus de périphériques « intelligents » que ce que l'architecture matérielle autorise. L'installation de plusieurs cartes ISA gérant chacune une interruption ou un canal DMA était réellement un casse têtes il y a encore peu de temps. C'est pour résoudre ces problèmes de configuration du matériel que le standard Plug and Play a été développé. Ce standard établit un protocole de communication matériel permettant au BIOS de configurer automatiquement les périphériques ISA en évitant les conflits sur les ressources partagées (le plus souvent, les lignes d'interruption).

Ce protocole a soulagé bien des gens, mais suppose un support spécifique de la part du système d'exploitation, puisque les paramètres matériels de chaque carte Plug and Play ne sont pas fixé d'un démarrage à l'autre de la machine. Il faut donc que le système soit ou capable de récupérer ces paramètres, ou de refaire la configuration du matériel. Cette deuxième solution est souvent imposée

par le fait que bon nombre de BIOS sont bogués et effectuent une mauvaise configuration des carte Plug and Play.

Le Plug and Play a été une amélioration sensible, mais n'a pas résolu les problèmes concernant les ressources limitées des PC. Heureusement, depuis l'avènement du bus PCI, ces limitations se sont estompées. En effet, le bus PCI permet non seulement aux périphériques de s'autoconfigurer et de se déclarer dans le système PCI, mais également de partager plusieurs lignes d'interruptions et de prendre le contrôle du bus mémoire. Ainsi, il n'y a virtuellement plus aucune difficulté pour installer une carte PCI dans un ordinateur (si vous avez le choix, prenez systématiquement des périphériques PCI). Ne croyez pas pour autant que le bus PCI soit idéal, c'est loin d'être le cas. En effet, il ne permet pas l'ajout et la suppression de cartes à chaud (c'est à dire lorsque le système est allumé), et fonctionne désormais à une vitesse bien trop faible compte tenu de la vitesse des processeurs, cartes graphiques et circuits mémoire. Il est donc appelé à disparaître sous peu lui-aussi.

Les périphériques PCI sont les plus courants actuellement. Le sous-système PCI de Linux permet de les utiliser directement, sans configuration particulière. Les périphériques ISA en revanche peuvent être plus difficile à configurer. S'ils ne sont pas Plug and Play, il n'y a pas d'autre choix que de spécifier leurs paramètres matériels directement, soit lors de la compilation du noyau, soit à l'aide d'options de boot lorsque le système démarre, soit à l'aide d'options dans le fichier `/etc/modules.conf` si leur gestionnaire est compilé sous forme de module. La première solution est la plus technique, puisqu'elle nécessite de reconfigurer et de recompiler le noyau, les deux autres relèvent de la configuration simple. La compilation du noyau sera décrite en détail dans le Chapitre 9.

Les cartes ISA Plug and Play constituent un cas à part, car, comme il l'a été dit ci-dessus, la plupart des BIOS Plug and Play sont bogués et les initialisent mal. Il est donc nécessaire que le système les initialise lui-même, et détermine par la même occasion leurs configuration matérielle. En pratique, il existe deux possibilités pour effectuer ces deux opérations. La première possibilité est de laisser le noyau faire tout le travail de configuration des cartes et d'allocation des ressources. La deuxième possibilité est d'effectuer la configuration des cartes ISA Plug and Play au niveau applicatif dans les scripts de démarrage du système. Cette solution est relativement technique, et ne doit être utilisée que lorsque l'on désire contrôler finement les ressources allouées par les périphériques dans le système.

Si l'on désire utiliser les fonctionnalités Plug and Play du noyau, il n'y a strictement rien à faire. Linux se charge simplement d'initialiser automatiquement les cartes ISA Plug and Play lors du démarrage du système, leur attribuera les paramètres matériels adéquats, et communiquera ces paramètres aux gestionnaires de périphériques. Il prendra également la gestion des conflits avec les autres périphériques, qu'ils soient ISA non Plug and Play ou PCI. Par conséquent, vous n'aurez aucune configuration spécifique à effectuer.

Mais Linux fournit également la possibilité de contrôler soi-même les ressources allouées à chaque périphérique, pour ceux qui veulent avoir la maîtrise totale de leur matériel ou ceux qui ont une configuration tellement spécifique qu'elle nécessite un paramétrage manuel. Dans ce cas, l'initialisation des cartes ISA Plug and Play ne se fera pas lors du démarrage du noyau, mais plutôt ultérieurement. En général, on effectue cette tâche dans les scripts d'initialisation du système, mais ce n'est pas une obligation. Quoi qu'il en soit, comme l'attribution des ressources aux cartes est différée, les gestionnaires de périphériques ne peuvent pas être inclus dans le noyau. Il est donc nécessaire d'utiliser les modules du noyau pour ces gestionnaires. Les paramètres matériels pourront alors être communiqués

simplement à ces gestionnaires lors du chargement des modules, à l'aide d'options dans le fichier de configuration `/etc/modules.conf`.

La configuration manuelle des cartes ISA Plug and Play se fait classiquement à l'aide des outils « isapnp ». Parmi ces outils se trouve le programme **isapnp**, que l'on utilise pour initialiser les cartes ISA Plug and Play. Cet utilitaire utilise les informations qui se trouvent écrite dans le fichier de configuration `/etc/isapnp.conf` pour déterminer les plages d'entrée/sortie, les canaux DMA et les lignes d'interruption à utiliser pour chaque carte.

La rédaction manuelle du fichier `isapnp.conf` n'est pas une tâche aisée. Heureusement, elle peut être réalisée automatiquement, à l'aide d'un autre utilitaire nommé **pnpdump**. Celui-ci affiche la liste des différentes possibilités de configuration pour chaque périphérique ISA Plug and Play. Cette liste est affichée exactement sous une forme exploitable par isapnp, ce qui fait qu'il est très simple de créer un fichier de configuration correct en faisant une redirection de sa sortie standard dans un fichier :

```
pnpdump > /etc/isapnp.conf
```

Le fichier de configuration ainsi créé contient les différentes configurations possibles. Cependant, elles sont toutes commentées, et aucun périphérique ISA ne sera configuré sans intervention supplémentaire. Il va donc falloir éditer ce fichier, et retirer les commentaires devant les options de configuration qui vous intéressent. Les lignes de commentaires commencent toutes par un caractère dièse ('#'). Il suffit donc d'effacer ce caractère pour les décommenter. Vous devez choisir les options à décommenter de telle manière qu'aucun conflit d'adresse, d'interruption ou de canaux DMA n'existent dans votre système. Pour chaque périphérique, plusieurs possibilités sont offertes, mais vous ne devez retirer les commentaires que devant les lignes d'une seule de ces possibilités. Les zones à décommenter sont clairement identifiées dans le fichier isapnp.conf généré par **pnpdump**, il vous suffit donc d'effectuer le choix de la configuration à utiliser. Enfin, il ne faut pas oublier de retirer le commentaire devant la ligne suivante :

```
(ACT Y)
```

à la fin des différentes options de configuration pour chaque carte correctement configurée, faute de quoi celle-ci ne sera pas activée lors de l'appel à **isapnp**.

En général, il est souhaitable d'appeler la commande **isapnp** à chaque démarrage du système, dans un des scripts de démarrage. Vous devrez donc inclure une ligne telle que celle-ci :

```
/sbin/isapnp /etc/isapnp.conf
```

dans le fichier de démarrage principal de votre système. Ce fichier peut se trouver dans le répertoire `/etc/rc.d/` (ou dans `/sbin/init.d/` selon la distribution que vous utilisez). La plupart des distributions permettent de paramétrer l'appel à cette commande à l'aide d'une option de configuration modifiable à l'aide de leur programme de configuration. Consultez la documentation de votre distribution pour plus de détails à ce sujet.

Une fois que vous aurez déterminé la configuration correcte pour vos périphériques dans le fichier `/etc/isapnp.conf`, vous pourrez charger les modules du noyau gérant ces périphériques. Ceci

peut nécessiter la modification du fichier `/etc/modules.conf`. Afin de limiter les risques d'erreur, vous devriez procéder comme suit :

- déterminer le fichier spécial de périphérique utilisé par les applications pour accéder à votre périphérique ;
- déterminer le nom du module que le noyau tentera de charger lorsqu'un programme tentera d'utiliser ce fichier spécial de périphérique ;
- rechercher la ligne définissant l'alias pour ce nom de module, afin de savoir quel est le module qui sera effectivement chargé par **modprobe**. Vous pouvez également trouver ce nom dans la documentation de la configuration du noyau pour le gestionnaire du périphérique que vous configurez, ou en regardant directement dans le répertoire d'installation des modules `/lib/modules/` ;
- ajouter éventuellement la ligne « `options module paramètres` » permettant de spécifier les paramètres de chargement `paramètres` pour le module `module` ;
- ajouter éventuellement les lignes « `pre-install` », « `post-install` », « `pre-remove` » et « `post-remove` » permettant d'effectuer des actions complémentaires avant et après le chargement du module, ainsi qu'avant et après son déchargement.

Comme il l'a été indiqué dans la Section 6.11, la détermination des noms de modules utilisés par le noyau pour les requêtes de chargement automatique n'est pas facile. Si vous utilisez le système de fichiers virtuel `/dev/`, ce nom est exactement le chemin complet du fichier spécial de périphérique. En revanche, si vous utilisez les fichiers spéciaux de périphériques classiques, vous aurez sans doute à vous servir du type, du code majeur et du code mineur de ce fichier spécial de périphérique. Vous pourrez obtenir ces informations à l'aide de la commande **ls -l fichier**, où `fichier` est le nom du fichier spécial de périphérique. Le type de ce fichier est indiqué dans le premier caractère des droits du fichiers. Le caractère '`c`' indique que le périphérique est un périphérique de type caractère, et le caractère '`b`' indique qu'il s'agit d'un périphérique de type bloc. Les numéros de codes majeurs et mineurs quant à eux sont indiqués juste après les informations concernant le propriétaire et le groupe du fichier, généralement égaux à « `root` » ;

Dans tous les cas, il est fortement recommandé de lire la documentation du module gérant votre périphérique Plug and Play. Cette documentation est en général située dans le répertoire `/usr/src/linux/Documentatio`

Pour tester si les modifications que vous avez effectuées sont correctes, vous pouvez essayer de charger le module avec la commande suivante :

```
modprobe module
```

où `module` est le nom du module à charger. Vous pouvez également vérifier que ce module se charge bien lorsqu'une requête sur le fichier spécial de périphérique correspondant est effectuée, avec par exemple la commande suivante :

```
echo Coucou > /dev/périphérique
```

où `périphérique` est le fichier spécial de périphérique à tester. Vous pouvez voir si le module s'est correctement chargé en demandant la liste des modules chargés à l'aide de la commande lsmod. Cette commande vous indique également l'état de chaque module, ainsi que le nombre de fois qu'il est utilisé et par qui. Si un module est marqué « `uninitialized` », vous avez de fortes chances de devoir redémarrer l'ordinateur et de revoir votre configuration, car un module qui se trouve dans cet état est inutilisable.

6.13. Configuration des cartes son

La plupart des cartes son vendues actuellement sont des cartes PCI, qui se configurent relativement aisément. Cependant, il existe encore un bon nombre de cartes son ISA, dont la configuration peut être plus technique, surtout si elles ne sont pas Plug and Play. De plus, les fonctionnalités fournies par les gestionnaires de son de Linux varient dans de larges proportions, selon le matériel utilisé. En particulier, toutes les cartes son ne sont pas capables de gérer la restitution des fichiers MIDI, et celles qui le gèrent nécessitent une configuration particulière. Cependant, il est possible d'utiliser un convertisseur logiciel de fichiers MIDI en fichiers son classique afin de jouer ce type de fichiers même avec une carte son classique.

La première étape lors de la configuration de votre carte son sera la sélection du driver à utiliser au niveau du noyau. D'une manière générale, la configuration du noyau consiste simplement à répondre à des questions concernant le noyau dans un programme de configuration. C'est une opération relativement simple, mais il faut éviter de se tromper lors de cette étape cruciale, faute de quoi le noyau ne serait pas fonctionnel ou ne se comporterait pas comme prévu. La configuration du noyau est décrite en détail dans le Chapitre 9.

En ce qui concerne les cartes son, les options de configuration se trouvent dans le menu « Sound ». Ce menu permet de choisir les drivers à utiliser en fonction du matériel installé. Ils sont classés en deux catégories : les drivers classiques (première partie des options du menu) et les drivers OSS (options du sous-menu « OSS sound modules »). OSS (« Open Sound System ») est une spécification d'interface de programmation permettant l'accès aux cartes son. Cette interface est en passe de devenir le standard dans le monde Unix. C'est un produit commercial, mais l'interface en soi est libre de droits et une implémentation libre en est fournie dans le noyau de Linux. Vous devez ici choisir le driver qui convient le mieux à votre matériel. Si vous avez le choix entre les drivers classiques et les drivers OSS, choisissez les drivers OSS car ils sont plus standard et vous n'aurez aucun problème à utiliser votre carte son avec la plupart des logiciels.

L'erreur la plus classique que l'on peut faire ici est de supposer que l'on possède une carte compatible Sound Blaster alors que ce n'en est pas une. Je tiens à préciser que quasiment aucune carte dite « compatible » sous Windows ne l'est sous Linux. La compatibilité sous Linux, c'est le fait d'avoir quasiment la même électronique ou du moins les mêmes interfaces au niveau matériel. Sous Windows, la compatibilité n'existe qu'au niveau des interfaces fournies par le driver de la carte son. Par conséquent, il vous faut savoir exactement de quel nature est votre carte son, et non ce que vous avez retenu des arguments commerciaux du fabricant. Notez en particulier que certaines cartes Sound Blaster ne sont pas compatibles Sound Blaster (Creative Labs est renommé en ce qui concerne les incompatibilités entre les différents modèles de cartes son). C'est notamment le cas pour les cartes

sons SB64 PCI et SB128 PCI, qui sont en réalité des cartes son ESS1370 ou ESS1371, et dont l'électronique est fabriquée par la société Ensoniq (cette société a été rachetée par Creative Labs, qui vend ces cartes en s'appuyant sur son image de marque et qui sème ainsi la confusion sur le marché). En conclusion, si vous ne voulez pas essayer plusieurs drivers et recompiler le noyau jusqu'à ce que vous trouviez le bon, renseignez-vous bien sur la nature de votre carte son (si possible avant de l'acheter, au moins afin de savoir exactement ce que vous aurez). Vous pouvez également taper la commande **lspci** afin de voir les périphériques réellement présents sur le bus PCI. La commande système **dmesg** peut également vous être utile. Elle permet de réafficher la liste des messages générés par le noyau lors de son amorçage.

Lorsque vous aurez choisi le driver adéquat, vous aurez le choix entre le compiler à l'intérieur du noyau (en choisissant l'option '`Y`' pour ce driver) ou le compiler sous forme de modules (en choisissant l'option '`M`'). En pratique, il est plus simple de compiler les drivers à l'intérieur du noyau pour les cartes PCI et les cartes ISA Plug and Play. En effet, Linux attribuera automatiquement les lignes d'interruptions, les portes d'entrée/sortie et les canaux d'accès direct à la mémoire pour ce type de matériel. Cette configuration n'est donc pas à faire, et votre carte son fonctionnera immédiatement sans problèmes.

C'est généralement également le bon choix pour les vielles cartes sons ISA qui ne sont pas Plug and Play. En effet, pour ces cartes sons, la configuration logicielle est très simple, puisqu'on ne peut pas les configurer du tout. Bien entendu, il faut que vous ayez résolu manuellement les conflits possibles de matériel de votre ordinateur, mais ceci ne concerne pas Linux. Pour ces cartes, il faut bien entendu indiquer au noyau les paramètres matériels (IRQ, DMA et ports d'entrée/sortie) lors de la configuration.

Il se peut toutefois que vous ne puissiez pas spécifier ces paramètres matériels dans les menus de configuration de Linux. Bien qu'en théorie il soit possible de modifier les fichiers sources de Linux directement pour indiquer ces paramètres, ce n'est pas à la portée du commun des mortels. Par conséquent, on utilisera les modules du noyau, et les options matérielles seront indiquées dans le fichier de configuration `/etc/modules.conf`. Notez que c'est également de cette manière que vous devrez procéder si vous désirez configurer vous-même l'allocation des ressources pour les cartes son ISA Plug and Play, ou, autrement dit, si vous préférez utiliser l'outil isapnp au lieu des fonctionnalités Plug and Play du noyau.

Ces opérations vous permettront de faire fonctionner votre carte son pour toutes les opérations de base, à savoir la lecture et l'enregistrement de fichiers son, le réglage du volume sonore et éventuellement la lecture des CD audio par l'intermédiaire de votre carte son (si, bien entendu, vous avez relié la sortie audio de votre lecteur de CD à l'entrée auxiliaire de votre carte son). Vous ne pourrez cependant pas lire les fichiers MIDI sans réaliser les opérations décrites ci-dessous.

Les fichiers MIDI sont des fichiers ne contenant que les « partitions » des morceaux enregistrés. Ils ne contiennent pas les données audio numériques comme c'est le cas des fichiers son classiques ou des pistes de CD audio. Ceci explique leur taille réduite, mais pose un problème pour leur restitution. En effet, la lecture d'un fichier MIDI suppose que votre carte son dispose d'un jeu de sons numérisés dans sa mémoire afin de pouvoir reconstituer le signal sonore à partir des informations du fichier MIDI. Ces sons sont couramment appelés des « patches », et ils peuvent être enregistrés soit en dur dans la mémoire de la carte son, soit chargés lors de son initialisation. Il va de soi que certaines cartes

son ne disposent pas des fonctionnalités nécessaires à la lecture des fichiers MIDI. Dans ce cas, il faut utiliser un programme externe capable de synthétiser le son à partir de patches enregistrés sur le disque dur et d'envoyer les données audio ainsi créées à la carte son pour la lecture.

Pour les cartes son qui ne disposent pas des patches en mémoire morte, il est nécessaire d'utiliser un programme afin de les charger dans la mémoire de la carte. Il va de soi que ce programme est spécifique à la carte son et doit être installé. Par exemple, pour les cartes Sound Blaster AWE, ce programme se nomme **sfxload**. Son utilisation requiert un fichier de définition des patches, fichier que vous trouverez normalement soit sur une installation de Windows avec une carte son AWE, soit sur vos CD d'installation, soit sur Internet. Le fichier fourni par Creative se nomme `synthgm.sbk`. Pour charger les patches dans la mémoire de la carte, il suffit d'utiliser la commande suivante :

```
sfxload /usr/lib/synthgm.sbk
```

(en supposant que le fichier de patches soit placé dans le répertoire `/usr/lib/`). Vous pourrez placer cette commande dans les scripts de démarrage de votre système ou dans une option de chargement du module `awe_wave` qui gère la lecture des fichiers midi. Vous trouverez ci-dessous un exemple d'options du fichier `/etc/conf.modules` permettant le chargement correct des modules d'une carte son Sound Blaster AWE :

```
# Chargement des modules de son :
alias char-major-14 sb
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
options adlib_card io=0x388

# Chargement automatique du module de lecture des fichiers MIDI :
post-install sb /sbin/modprobe -k midi
alias midi awe_wave

# Chargement des patches en mémoire :
post-install awe_wave /usr/local/bin/sfxload /usr/lib/synthgm.sbk
```

Pour les cartes son ne disposant pas de la possibilité de lire les fichiers MIDI, il ne reste que la solution de l'émulation logicielle. Vous pourrez dans ce cas recourir à l'émulateur « SoftOSS » fourni avec les drivers OSS ou à un autre programme très utilisé nommé « Timidity ». « SoftOSS » a l'avantage d'être intégré au drivers OSS, mais souffre de deux inconvénients. Premièrement, il n'est disponible qu'avec ces drivers, et deuxièmement, il consomme beaucoup de mémoire du noyau pour charger les patches. Timidity en revanche n'est pas intégré au noyau, et n'utilise par conséquent pas la mémoire système. De plus, la mémoire consommée par les patches est déchargée dès que Timidity se termine. Enfin, le moteur de Timidity est utilisé par d'autres programmes externes, et en particulier par le lecteur KMidi de KDE. En raison de son caractère plus général, seule l'installation de Timidity et la configuration de KMidi seront présentées ci-dessous.

L'auteur originel de Timidity ne le maintient plus. Cependant, d'autres programmeurs ont pris le relais et l'ont amélioré pour en faire la version Timidity++. Cette dernière version peut être trouvée sur Internet sur le site de Timidity (http ://www.goice.co.jp/member/mo/timidity/). L'archive que vous

récupérerez ne contient pas forcément le programme : il peut ne s'agir que des fichiers sources permettant de le générer. Les notions de fichiers sources et de compilation de programmes exécutables seront expliquées en détail dans le chapitre suivant. En attendant, si vous récupérez les sources, vous pourrez suivre les indications données ci-dessous pour compiler et installer Timidity.

Timidity peut utiliser différentes interfaces utilisateur, accessibles via différentes librairies. Il est recommandé d'utiliser l'interface Tcl/Tk, car cette interface fonctionne aussi bien sous XWindow qu'en mode texte. Pour cela, il suffit de configurer les sources avec la ligne de commande suivante :

```
./configure --enable-dynamic --enable-tcltk=dynamic
```

Une fois ceci fait, vous pourrez générer l'exécutable avec la commande suivante :

```
make
```

et l'installer avec la commande :

```
make install
```

Ceci installera Timidity dans le répertoire `/usr/local/bin/`.

La deuxième étape est bien entendu d'installer les fichiers de patches. Timidity utilise des patches pour les cartes son Gravis Ultrasound. Vous pourrez trouver des archives contenant de tels patches sur Internet très facilement. En particulier, la collection de patches EAWPATS réalisée par Eric A. Welsh est très complète, quoique un peu grosse pour un téléchargement. Cette collection a de plus l'avantage d'être préconfigurée pour l'emploi avec Timidity.

Lorsque vous aurez installé les fichiers de patches, vous devrez indiquer à Timidity leur emplacement dans le système de fichiers. Cette information doit être stockée dans le fichier de configuration `timidity.cfg` du répertoire `/usr/local/share/timidity/`. Vous pourrez vous inspirer du fichier fourni avec la collection de patches EAWPATS. Cependant, vous aurez à modifier les chemins indiqués dans ce fichier, car il a été écrit pour la version Windows de Timidity. Le premier répertoire à indiquer est celui des librairies. Il faut ici simplement remplacer la ligne « `dir c :\timidity` » par la ligne « `dir /usr/local/lib/timidity` ». Notez que cette ligne est facultative et peut être commentée si vous n'avez pas déplacé ce répertoire. Le deuxième répertoire est plus important, puisque c'est le répertoire d'installation des patches. Il s'agit cette fois de remplacer la ligne « `dir c :\eawpats` » par la ligne « `dir répertoire` », où `répertoire` est le répertoire où vous avez extrait les fichiers patches.

Une fois que vous aurez terminé ces modifications, vous pourrez lire un fichier MIDI simplement en utilisant la ligne de commande suivante :

```
timidity fichier
```

où `fichier` est le nom du fichier MIDI à lire. Si vous désirez utiliser l'interface graphique de Timidity, vous n'avez qu'à ajouter l'option `-ik` à la ligne de commande précédente.

L'installation de KMidi ne pose pas de problème particulier, puisqu'il est fourni avec KDE et que KDE est normalement inclus dans toute bonne distribution. La seule opération que vous ayez à faire est simplement de modifier le fichier de configuration de KMidi pour lui indiquer l'emplacement des fichiers de patches. Or comme KMidi est basé sur les sources de Timidity, il utilise le même fichier de configuration `timidity.cfg` que Timidity. Ce fichier est normalement situé dans le répertoire `/base/kde/share/apps/kmidi/config/`, où `base` représente le répertoire de base d'installation de KDE. Vous n'aurez donc qu'à répéter les opérations faites sur le fichier de configuration de Timidity avec le fichier de configuration de KMidi.

6.14. Installation d'une carte d'acquisition vidéo

Linux fournit toutes les fonctionnalités nécessaires à la manipulation des flux de données vidéo par l'intermédiaire d'une interface de programmation nommée *video4linux* (ce qui se lit « Video for Linux »). Linux est capable de gérer la plupart des cartes d'acquisition TV du marché et quelques-unes des cartes d'acquisition vidéo. Comme d'habitude, seuls les constructeurs de matériel qui ont accepté de jouer le jeu de fournir les informations nécessaires à la programmation de gestionnaires de périphériques libres voient leur matériels supportés sous Linux. Par conséquent, il est encore une fois nécessaire de bien se renseigner sur la nature du produit et la politique du fabricant lorsque vous achèterez une carte d'acquisition TV.

En pratique, quasiment toutes les cartes d'acquisition TV utilisent le circuit intégré Bt848 ou un de ses dérivés, et Linux sait les gérer sans problème. Les cartes d'acquisition et de montage vidéo utilisent d'autres circuits plus puissants, dont les spécifications sont généralement non disponibles. Seule la configuration des cartes TV sera donc décrite ci-dessous.

Les applications, pour accéder aux périphériques vidéo, utilisent l'un des fichiers spéciaux de périphérique `/dev/video*`. Ces fichiers sont des fichiers de type caractère, dont le code majeur vaut 81. Le code de mineur est utilisé pour différencier les différents périphériques installés sur votre système. En général, il existe un lien symbolique `/dev/video` sur l'un de ces fichiers spéciaux, qui sera utilisé pour accéder au périphérique vidéo par défaut. Normalement, tous ces fichiers sont installés d'office dans le répertoire `/dev/` par les distributions, et vous n'aurez pas à les créer vous-même.

Le support de la vidéo sous Linux passe bien entendu par la configuration du noyau. Cette fois, il n'est pas certain du tout que le noyau fourni avec votre distribution supporte les cartes d'acquisition TV, aussi vous aurez peut-être à recompiler vous-même votre noyau. La manière de procéder est décrite en détail dans le Chapitre 9.

Sachez toutefois que les options à valider dans la configuration du noyau se trouvent dans le menu « Multimedia devices ». Vous devrez activer l'option « Video For Linux » sous la forme de module, car vous aurez sans doute à communiquer des paramètres aux modules vidéo afin de leur spécifier le type de carte utilisé et le type de tuner. Le module qui sera créé portera le nom `videodev`, c'est celui qui se chargera de répondre aux requêtes du client sur les fichiers spéciaux de périphérique `/dev/video*`. Lorsque vous aurez activé la fonctionnalité de vidéo pour Linux, vous aurez le choix des drivers de cartes vidéo dans le sous-menu « Video adapters ».

> **Note:** La configuration des cartes basées sur la puce Bt848 (option de menu « BT848 Video For Linux ») n'est accessible que si vous avez également activé l'option « I2C bit-banging interfaces » du menu « I2C support ». De même, le module prenant en charge les puces du type Bt848 ne prend pas en charge la gestion du son. En revanche, vous pourrez trouver le gestionnaire du son de ces cartes avec les autres gestionnaires de cartes son, dans le menu « Sound ». L'option que vous devez activer est l'option « TV card (bt848) mixer support ».

Une fois le noyau recompilé votre noyau et les nouveaux modules installés, il faut modifier le fichier de configuration `/etc/modules.conf`. Bien que Linux soit parfaitement capable de déterminer les ressources requises par les cartes vidéo, il est rare que le matériel soit correctement identifié par les gestionnaires de périphériques vidéo. En effet, ces gestionnaires se basent plus sur les composants courants permettant de faire l'acquisition vidéo que sur les modèles de cartes de chaque fabricant. La palme revient sans doute au gestionnaire pour les cartes basées sur les puces Bt848 et ses dérivées, puisqu'il est capable de faire fonctionner toutes les cartes vidéo de tous les fabricants qui utilisent cette puce. Par conséquent, il faut indiquer le modèle de la carte au gestionnaire, à l'aide des options de modules dans le fichier `modules.conf`.

En fait, le gestionnaire pour les cartes basées sur la puce Bt848 est constitué de deux modules. Le premier, nommé `bttv`, gère la puce Bt848 elle-même et prend en charge les flux vidéo et l'incrustation vidéo dans la mémoire de la carte graphique. Le deuxième module quant à lui s'occupe de la gestion de la puce du tuner, qui effectue la recherche des chaînes et le décodage des signaux vidéo. Pour ces deux modules, il est indispensable de préciser quelle type de carte TV est installé et quel tuner cette carte utilise. Il existe de nombreuses variantes, selon les modèles et les différents fabricants présents sur le marché.

Le type de carte peut être communiqué au module bttv à l'aide de l'option `card`. Cette option peut prendre comme valeur un numéro identifiant le modèle de la carte. Les valeurs actuellement supportées sont indiquées dans le fichier `CARDLIST` du répertoire `/usr/src/linux/Documentation/video4linux/bttv/`

Le module de gestion du tuner utilise l'option `type` pour déterminer le type de tuner utilisé. Cette option prend, elle aussi, une valeur numérique indiquant la nature du tuner utilisé. Les valeurs supportées sont également données dans le fichier `CARDLIST`. En pratique, il est fort probable que vous utilisiez le type 3, qui correspond au tuner Philips SECAM, car la France utilise le standard SECAM pour les émissions TV.

Ainsi, si vous disposez d'une carte MIRO PCTV (carte de type 1) basée sur ce tuner Philips SECAM, vous devriez avoir les lignes suivantes dans votre fichier `modules.conf` :

```
# Lien entre les fichiers spéciaux de périphériques et les modules du noyau :
alias char-major-81 videodev
alias char-major-81-0 bttv

# Options du module de gestion du Bt848 :
options bttv card=1

# Option du module de gestion du tuner :
options tuner type=3
```

Vous devrez bien entendu adapter ces lignes selon votre propre configuration.

> **Note:** Si vous avez compilé les fonctionnalités de l'interface I2C sous forme de module (option de menu « I2C bit-banging interfaces »), vous aurez également à ajouter ces lignes dans votre fichier de configuration `modules.conf` :
>
> ```
> # Lien pour les fonctionnalités I2C :
> alias char-major-89 i2c-dev
>
> # Activation de l'agorithme bit-banging :
> options i2c-algo-bit bit_test=1
> ```
>
> Pour information, I2C est un protocole de communication entre micro-contrôleurs, que la plupart des cartes mères sont capables de gérer. Cette fonctionnalité n'est nécessaire que pour les cartes basées sur la puce Bt848.

Lorsque vous aurez fini vos modifications dans le fichier de configuration `modules.conf`, n'oubliez pas d'appeler la commande **depmod -a** pour remettre à jour les dépendances entre les modules du noyau. Si tout se passe bien, vous pourrez utiliser les programmes de lecture TV, comme KWinTV par exemple si vous utilisez l'environnement de bureau KDE.

> **Note:** Vous pouvez rencontrer quelques problèmes lors de la configuration de votre carte TV. Généralement, si vous n'obtenez aucune image, c'est que vous vous êtes trompés de tuner. Revoyez dans ce cas l'option `type` du module tuner. Si vous obtenez bien une image, mais pas de son, c'est sans doute que vous vous êtes trompés dans le type de carte, ou que vous avez oublié d'inclure le support du son pour les cartes à base de Bt848. On notera que, comme pour les cartes son, seule la compatibilité matérielle importe. Par exemple, les cartes Studio PCTV de Pinacle vendues en France sont en réalité des cartes Miro PCTV et ne sont pas reconnues comme des cartes Studio PCTV par Linux... Si vous avez des problèmes de son, vous devrez donc revoir la configuration du noyau ou modifier la valeur passée à l'option `card` du module bttv. Dans tous les cas, n'hésitez pas à utiliser la commande **lspci**, qui permet de visualiser les informations du bus PCI, et la commande **dmesg**, qui permet de voir la liste des messages du noyau.
>
> Vous pouvez également avoir quelques problèmes de droits sur les fichiers spéciaux de périphériques `/dev/videoX`. Le symptôme classique est dans ce cas que tout fonctionne parfaitement sous le compte root, mais pas sous les comptes utilisateurs. Dans ce cas, on pourra résoudre ces problèmes en attribuant un groupe d'utilisateurs `video`, auxquel appartiendront tous les utilisateurs ayant le droit d'utiliser la carte d'acquisition TV, et de changer le groupe des fichiers spéciaux de périphériques `/dev/videoX`.
>
> Enfin, l'utilisation des cartes d'acquisition TV nécessite d'activer les fonctionnalités DGA du serveur X. Ces fonctionnalités permettent aux programmes d'accéder directement à la mémoire vidéo, et donc de faire l'incrustation de l'image décodée par la carte TV dans la surface d'affichage d'un écran. La manière d'activer les fonctionnalités DGA sera précisée dans le chapitre traitant de la configuration du serveur X.

6.15. Installation d'un graveur de CD-ROM

Linux dispose de tous les logiciels permettant de graver des CD-ROM, et ce avec la plupart des graveurs disponibles sur le marché. Il permet de copier des CD et de créer des images disques. Cependant, il n'est pas encore possible d'utiliser le gravage par paquet, qui permet d'utiliser les graveurs de CD comme des périphériques amovibles. Cette fonctionnalité est en effet encore à l'étude, elle sera sans doute disponible sous peu.

6.15.1. Notions de base sur le gravage sous Linux

Originellement, tous les graveurs de CD utilisaient l'interface SCSI. Ce n'est que plus tard que les graveurs sur port parallèle et les graveurs IDE ATAPI sont apparus. Actuellement, les graveurs externes connectés sur port parallèle sont de moins en moins vendus. Les graveurs IDE font une belle percée, mais les graveurs SCSI restent incontournables pour les vielles machines dont on n'est pas sûr des performances au niveau des disques. Les logiciels de gravage doivent donc être capable d'utiliser n'importe quel type de graveur, ce qui n'est pas simple. C'est pour cette raison qu'ils utilisent tous l'interface SCSI, aussi bien pour des raisons historiques que pour des raisons techniques. Ceci ne signifie pas qu'il est impossible d'utiliser les graveurs ATAPI, car le protocole de communication ATAPI n'est rien d'autre qu'une traduction des commandes IDE en commandes SCSI au niveau matériel.

L'utilisation d'un graveur de CD SCSI ne posera donc pas de problèmes et se fera de manière directe. En revanche, les graveurs de CD IDE ATAPI devront être accédés par l'intermédiaire d'un émulateur SCSI, qui traduira les commandes SCSI en commandes ATAPI correspondantes. Cet émulateur est géré au niveau du noyau de Linux. Le problème est qu'il est impossible d'activer à la fois le support des CD-ROM IDE ATAPI et l'émulateur SCSI pour les périphériques IDE dans un même noyau. On devra donc faire le choix entre les deux solutions suivantes :

- soit on n'utilise que l'interface SCSI pour accéder aux lecteurs de CD, et on ne risque aucun conflit ;
- soit on active les deux fonctionnalités sous la forme de modules, et on n'utilise qu'une seule des interfaces à chaque instant.

La première solution est évidemment la plus simple (et surtout la plus sûre). La deuxième solution permet d'utiliser ses lecteurs de CD-ROM IDE ATAPI normalement avec leur interface native, sauf lorsque l'on désire graver un CD. Comme il n'y a pas de cas où l'interface SCSI est inférieure en terme de performances et de fonctionnalités à l'interface IDE, cette méthode est déconseillée.

6.15.2. Configuration du noyau

Vous allez encore une fois avoir à effectuer des choix de configuration dans le noyau pour activer la gestion des périphériques SCSI capables de piloter votre graveur. Les questions concernant les graveurs de CD-ROM sont récapitulées ci-dessous :

« `Include IDE/ATA-2 CDROM support` »

Cette option permet d'activer la gestion des lecteurs de CD IDE ATAPI. Vous ne devez pas répondre à cette question par 'Y', faute de quoi vous ne pourrez pas utiliser votre graveur de CD. Il vous est en revanche possible de répondre par 'M', pour n'activer la fonctionnalité qu'à la demande, sous la forme de module. La réponse recommandé est 'N', à moins que vous ne sachiez ce que vous faites. En effet, si vous activez cette fonctionnalité, vous risquez d'avoir des conflits avec l'émulation SCSI. Ce problème sera détaillé plus loin.

« `SCSI emulation support` »

Cette option permet d'activer la gestion des périphériques IDE ATAPI par l'intermédiaire de commandes SCSI. C'est cette option qui va vous permettre d'utiliser votre graveur de CD IDE avec les programmes qui ne peuvent gérer que des graveurs SCSI. Vous pouvez répondre 'N' à cette question si vous n'utilisez ni lecteur de CDROM SCSI ni graveur SCSI. Dans tous les autres cas, il est recommandé de répondre 'Y' à cette question afin d'accéder à votre matériel via l'interface SCSI. Vous pouvez activer cette fonctionnalité sous la forme de module en répondant par 'M' à cette question, cette solution est recommandée si vous avez activé la gestion des CD-ROM IDE ATAPI dans la question précédente.

« `Loopback device support` »

Cette option permet d'activer une fonctionnalité permettant d'utiliser un fichier comme un périphérique normal. Elle est très utile pour tester les images de CD, car il suffit simplement de monter le fichier image comme un périphérique « loopback ». Il est donc recommandé d'activer cette fonctionnalité, que l'on dispose d'un graveur SCSI ou IDE. Cette fonctionnalité peut être activée sous la forme de module ou non, la réponse recommandée est 'Y'.

« `SCSI support` »

Cette option permet d'activer la gestion des périphériques SCSI dans votre noyau. Il va de soi qu'il faut l'activer, que l'on dispose d'un graveur SCSI ou IDE. Cette fonctionnalité peut être activée sous forme de module ou non. La réponse recommandée est 'Y'.

« `SCSI CD-ROM support` »

Cette option permet d'activer la gestion des lecteurs de CD SCSI. Il faut l'activer, que l'on dispose d'un graveur SCSI ou IDE. Il est impératif d'utiliser les modules si vous avez activé la gestion des lecteurs IDE. Vous pouvez intégrer cette fonctionnalité au noyau sans risque sinon. Cette dernière solution est la méthode recommandée, aussi est-il conseillé de répondre 'Y' à cette question.

« `Enable vendor-specific extensions (for SCSI CDROM)` »

Cette option permet d'autoriser l'emploi d'extensions au protocole SCSI définies par les fabricants de matériels. Certains graveurs utilisent de telles commandes, c'est en particulier le cas des graveurs de CD HP. Toutefois, si vous ne disposez pas d'un tel graveur, il est peu probable que vous ayez à activer cette fonctionnalité. La réponse recommandée est donc 'N'.

« `SCSI generic support` »

Cette option permet d'utiliser le périphérique SCSI avec des commandes non standard, ce qui requiert l'emploi de programmes capables de communiquer directement avec les périphériques SCSI. C'est le cas pour les graveurs, qui seront pilotés directement par les logiciels de gravage. Il faut donc activer cette fonctionnalité, que l'on dispose d'un graveur SCSI ou IDE. La réponse recommandée est '`Y`'.

« `Probe all LUNs on each SCSI device` »

Cette option permet, lorsqu'on dispose de périphériques SCSI capables de gérer plusieurs numéros d'unité logiques, de leur demander tous ces numéros. Ce type de périphériques est assez rare, et en général chaque périphérique n'utilise qu'un et un seul numéro d'unité logique. Il ne faut donc pas, en général, activer cette fonctionnalité. La réponse recommandée pour cette question est donc '`N`'.

Enfin, il faut choisir le driver bas niveau permettant de gérer son graveur de CD. Pour les graveurs IDE ATAPI, la couche d'émulation suffira et il est inutile de sélectionner un autre driver. En revanche, pour les graveurs SCSI, il faut choisir le driver approprié.

Une fois cette configuration effectuée, il ne reste plus qu'à compiler le noyau et les modules et à les installer. Vous trouverez plus de détails à ce sujet dans le chapitre traitant de la compilation du noyau.

6.15.3. Configuration des modules du noyau

Vous aurez sans doute à modifier le fichier de configuration `/etc/modules.conf` si vous avez activé certaines fonctionnalités SCSI sous la forme de modules, afin de charger les modules nécessaires lorsqu'une commande sera effectuée sur un des fichiers spéciaux de périphériques SCSI. Si toutes les fonctionnalités SCSI sont intégrées au noyau, cette étape est bien entendue facultative. Les fichiers spéciaux utiles pour les périphériques blocs SCSI sont les fichiers `scdx` et `sgx`, où x est le numéro du périphérique. Le premier groupe de fichiers spéciaux permettent d'accéder aux lecteurs de CD SCSI, et le deuxième groupe d'y accéder en tant que périphériques SCSI génériques. Les codes majeurs pour ces deux groupes de fichiers spéciaux sont respectivement 11 et 21, il faut donc ajouter les entrées du fichier `modules.conf` pour les modules nommés `block-major-11` et `block-major-21`. Normalement, ces entrées sont déjà présentes dans le fichier fourni par votre distribution. Cependant, vous aurez à ajouter les lignes suivantes si vous utilisez un graveur de CD IDE ATAPI :

```
post-install sr_mod modprobe -k ide-scsi
```

après la ligne indiquant le module à charger pour les périphériques blocks de code majeur 11, et :

```
post-install sg modprobe -k ide-scsi
```

pour les périphériques SCSI génériques, de code majeur 21.

Ces deux lignes permettent de demander au noyau de charger également le module `ide-scsi` lorsque l'un des modules SCSI est chargé. Le module `ide-scsi` est le module qui se charge d'effectuer l'émulation SCSI pour les périphériques IDE ATAPI.

Il va de soi que lorsque vous aurez utilisé un des fichiers périphériques SCSI, le module `ide-scsi` sera chargé. Vous ne pourrez donc plus utiliser le module `ide-cd`, qui permet d'accéder aux lecteurs IDE par leur interface ATAPI. De même, si vous montez votre lecteur de CD par l'intermédiaire du fichier spécial de périphérique IDE correspondant, le module `ide-cd` sera chargé. Vous ne pourrez donc plus utiliser le module d'émulation SCSI. La solution est donc, dans ce cas, de décharger les modules inutilisés à l'aide de la commande **rmmod**. Il est évident que comme vous n'avez pas besoin d'utiliser l'interface IDE de vos lecteurs de CD dans la plupart des cas, le plus simple est simplement de toujours utiliser l'émulation SCSI et ainsi d'éviter tout conflit. Dans ce cas, vous pourrez répondre par 'N' à la question « Include IDE/ATA-2 CDROM support » de la configuration du noyau, surtout si vous ne maîtrisez pas Linux ou si n'avez pas bien assimilé le fonctionnement des modules.

6.15.4. Installation des logiciels de gravage

Lorsque ceci aura été réalisé, il ne vous reste plus qu'à installer les logiciels de gravage. Les logiciels indispensables sont les suivants :

- **cdrecord**, qui permet de piloter le graveur de CD et d'effectuer les tâches nécessaires pour le gravage ;
- **mkisofs**, qui permet de créer des images disques ISO 9660 ;
- **mkhybrid**, qui permet de créer de images disques ISO 9660 avec les extensions Joliet (CD Windows 95), Rock Ridge (CD Unix) ou HFS (CD Macintosh). Ce programme est très pratique, car les CD ainsi créés pourront être lus correctement par plusieurs systèmes ;
- **cdda2wav**, qui permet d'extraire les données numériques des CD audio.

Vous pourrez vérifier que votre configuration fonctionne correctement avec la commande suivante :

```
cdrecord -scanbus
```

Cette commande recherche tous les périphériques SCSI du système. Vous devrez normalement y trouver, en plus de vos autres périphériques SCSI, votre graveur de CD.

6.15.5. Utilisation des logiciels de gravage

Nous allons à présent voir les commandes permettant de faire les principales opérations ayant trait au gravage des CD. Bien que ce document soit consacré à l'installation de Linux et non à son utilisation, elles vous permettront de tester si tout fonctionne correctement.

La copie directe d'un CD peut se faire avec la commande suivante :

```
cdrecord -dummy -v dev=bus,ID,0 speed=n -isosize /dev/scdx
```

où `bus` représente le bus SCSI sur lequel le graveur est branché, `ID` le numéro de périphérique du graveur sur ce bus SCSI et `nul` son numéro d'unité logique. Le numéro du bus SCSI et le numéro de périphérique peuvent être obtenus avec la commande **cdrecord -scanbus** présentée ci-dessus. Le numéro d'unité logique est, dans la majorité des cas, 0. Le nombre `n` doit ici valoir le facteur multiplicateur de la vitesse de gravage. Vous pouvez essayer avec des valeurs faibles pour commencer, afin d'être sûr que tout se passe bien. Le fichier de périphérique `/dev/scdx` indiqué dans la ligne de commande de cdrecord doit bien entendu être le fichier de périphérique SCSI du lecteur de CD utilisé pour lire le CD-ROM. L'option `-isosize` permet de demander à cdrecord de lire la taille de la piste à graver dans le système de fichiers ISO9660 du CD source. Enfin, l'option `-dummy` permet de faire toutes les opérations en conservant le laser du graveur éteint, ce qui revient à faire un test. Si vous voulez réellement graver votre CD, il suffit de supprimer cette option.

Il va de soi que si vous ne disposez pas d'un lecteur de CD en plus de votre graveur, vous ne pouvez pas utiliser la commande précédente. Dans ce cas, vous devrez faire une image disque en extrayant toutes les données du CD source. Ceci peut être réalisé avec la commande suivante :

```
dd if=/dev/scdx of=image
```

où `/dev/scdx` est le fichier spécial de périphérique SCSI du lecteur utilisé pour faire l'image (donc, si vous ne disposez que d'un graveur, c'est le fichier spécial de périphérique SCSI de votre graveur), et `image` est le fichier image qui doit être créé.

Le gravage d'une image disque peut être réalisé avec la commande suivante :

```
cdrecord -dummy -v dev=bus,ID,0 speed=n -data image
```

où les options sont les mêmes que dans les commandes précédentes.

Si vous désirez créer une image disque à partir des fichiers de votre disque dur, vous pouvez utiliser **mkisofs** ou **mkhybrid**. Le premier programme permet de créer une image disque ISO9660, avec éventuellement les extensions Rock Ridge activant les fonctionnalités des systèmes de fichiers Unix (noms de fichiers plus longs, liens symboliques, droits, etc...). Le deuxième programme permet de créer des images qui seront lisibles sur plusieurs systèmes d'exploitation, en fournissant les extensions Joliet sous Windows 95 et NT 4 (noms de fichiers longs), Rock Ridge sous Unix et HFS sous Macintosh. Le prix à payer pour cette amélioration est la perte de quelques centaines de kilo octets, ce qui est dérisoire étant donné le gain en portabilité.

Les options des deux utilitaires **mkisofs** et **mkhybrid** sont similaires. La ligne de commande à utiliser pour créer une image disque est la suivante :

```
mkisofs  [-r | -R]  -V "nom" -o image répertoires
```

ou

```
mkhybrid [-r | -R] [-J] [-hfs] -V "nom" -o image répertoires
```

selon le programme utilisé.

Les options `-r` ou `-R`, `-J` et `-hfs` permettent respectivement d'utiliser les extensions Rock Ridge, Juliette ou HFS. **mkhybrid** est capable de prendre plusieurs de ces options simultanément, et de créer des CD hybrides qui fonctionneront sur plusieurs systèmes. La distinction entre l'option `-r` et l'option `-R` est que `-r` modifie les attributs des fichiers Unix afin que le CD puisse être utilisé sur un autre ordinateur que celui sur lequel le CD a été créé. En particulier, le propriétaire et le groupe des fichiers voient leur valeurs fixées à 0. L'option `-V` permet de fixer le nom du volume dans l'image. Il est possible de placer ce nom entre guillemets. Cette fonctionnalité est intéressante si ce nom comprend des espaces. L'option `-o` permet de spécifier le nom du fichier image qui doit être créé.

Les paramètres suivants constituent la liste des répertoires et des fichiers qui doivent être insérés dans le CD-ROM. Par défaut, chaque répertoire ou fichier indiqué en paramètre est placé dans le répertoire racine du CD-ROM, et l'arborescence des sous-répertoires est respectée. Cependant, il est possible de placer un des répertoires ou fichiers indiqué en paramètre dans un autre répertoire du CD-ROM, à l'aide de la syntaxe suivante :

```
destination=source
```

où `destination` est le répertoire destination du CD-ROM, et `source` le répertoire ou le fichier source à ajouter à l'image disque. De même, il est possible d'exclure certains sous-répertoires de l'image disque à l'aide de l'option `-x` :

```
-x répertoire
```

Les sous-répertoires du répertoire indiqué seront également supprimés de l'image disque.

Vous pouvez tester votre image disque en la montant par l'intermédiaire du périphérique loopback. Il faut pour cela que vous ayez activé la gestion de ce périphérique dans la configuration du noyau. La commande à utilise pour monter ce type de périphérique est légèrement plus compliquée que pour les systèmes de fichiers classiques :

```
mount -t iso9660 -o ro,loop=/dev/loop0 image /cdrom
```

où `image` est le nom de votre fichier image à tester. Cette commande monte le système de fichiers de cette image dans le répertoire `/cdrom`. La commande **umount** peut être utilisée pour démonter ce système de fichiers, et elle s'utilise alors exactement comme pour les autres systèmes de fichiers.

Les commandes présentées ci-dessus ne permettent pas de travailler avec des CD audio. Pour extraire les données audio d'un CD, vous devrez utiliser le programme **cdda2wav**» :

```
cdda2wav [-H] -B nom [-tdébut[+fin]] -O wav -D bus,ID,nul
```

L'option `-H` permet d'éviter la création des fichiers d'information `.inf` sur les pistes extraites par **cdda2wav**. L'option `-B` permet d'utiliser le nom `nom` complété d'un numéro pour les fichiers de données créés. Les pistes seront donc stockées dans des fichiers portant les noms `nom_01`, `nom_02`, etc... L'option `-t` permet d'indiquer la piste `début` et la piste `fin` afin de sélectionner les pistes dont les données audio doivent être extraites. Si la piste de fin n'est pas précisée, toutes les pistes depuis la piste de début jusqu'à la dernière piste du CD seront extraites. De même, si la piste de début n'est pas précisée, l'extraction commencera à partir de la première piste. L'option `-O` permet d'indiquer le

type de fichier de sortie, `wav` indique ici que les fichiers créés seront au format WAV des fichiers sons de Windows. Il est également possible d'utiliser le type de fichier `raw`, qui sont prêts à être gravés tels quels. Enfin, l'option `-D` permet de spécifier le périphérique SCSI à utiliser pour lire les données audio. Ce périphérique est sélectionné par le numéro du bus SCSI sur lequel il se trouve, son numéro dans ce bus et son numéro d'unité logique.

Le gravage des pistes audio peut être réalisé avec la commande suivante :

```
cdrecord -dummy -v dev=bus,ID,nul speed=n -nofix -audio  piste1.wav \
    piste2.wav ...
```

L'option `-nofix` permet ici d'éviter de fermer la session. Elle doit être utilisée si l'on désire rajouter des pistes audio sur le CD-ROM ultérieurement. Si, en revanche, elle est omise, cdrecord fermera automatiquement la session après l'écriture de la dernière piste, et plus aucune donnée ne pourra être ajoutée. La commande suivante vous permettra de fixer le disque a posteriori :

```
cdrecord -dummy -v dev=bus,ID,nul speed=n -fix
```

Enfin, si vous disposez d'un graveur de CD réinscriptible, vous pourrez utiliser la commande suivante pour effacer un CDRW :

```
cdrecord -dummy -v dev=bus,ID,nul -blank=fast
```

où `bus,ID,nul` représente toujours le périphérique SCSI à utiliser en tant que graveur.

Il est peu probable que vous ayez à utiliser ces commandes directement, car il existe un programme graphique capable de les piloter de manière conviviale. Ce programme se nomme **xcdroast**, et peut être récupéré sur Internet si votre distribution ne le comprend pas. Il est vivement recommandé de l'installer. Son utilisation est assez élémentaire et ne devrait pas poser de problème particulier.

6.16. Installation des périphériques USB

Bien qu'ayant eu quelques difficulté à ses débuts, le port USB est désormais en pleine évolution. Il s'agit d'une extension des ports série classiques, qui offre les avantages suivants :

- possibilité de connecter jusqu'à 127 périphériques sur un même port, selon une structure arborescente ;
- bande passante accrue jusqu'à 12 Mbits/s théoriques ;
- capacité de connexion des périphériques « à chaud » et détection automatique par le système d'exploitation ;
- possibilité d'alimentation des périphériques par le bus lui-même, évitant ainsi d'avoir des câbles supplémentaires pour les périphériques consommant peu d'éléctricité.

Tous ces avantages font que le bus USB est appelé à remplacer les ports série que l'on connaît, et que l'on n'utilise plus désormais que pour les modems, les vieilles souris série et quelques appareils extérieurs. Notez que la simplicité du port série fera qu'il restera encore présents sur bon nombre d'appareils pour plusieurs années encore, mais les périphériques informatiques risquent de s'en détourner de plus en plus.

La gestion de l'USB sous Linux se fait au niveau du noyau en ce qui concerne les drivers, et au niveau applicatif pour la détection et la configuration dynamique des périphériques branchés à chaud. Linux est capable d'utiliser la plupart des périphériques USB existant actuellement sur le marché. La configuration à chaud des périphériques en revanche n'est pas encore tout à fait finalisée, bien que les mécanismes soient tous en place. La configuration et la compilation du noyau seront décrites en détail dans le Chapitre 9.

6.16.1. Configuration du noyau

La configuration des périphériques USB se fait, au niveau du noyau, dans le menu « `USB support` ». Il faut simplement activer l'option « `Support for USB` », sélectionner un gestionnaire pour le port USB et choisir les drivers des différents types de périphériques que l'on voudra connecter sur la machine.

Il peut être utile d'activer l'option « `Preliminary USB device filesystem` » afin de pouvoir vérifier, dans le répertoire `/proc/bus/usb/` du système de fichiers virtuel `/proc/`, que les périphériques USB connectés apparaissent bien. Ce système de fichiers doit être monté à l'aide de la commande suivante :

```
mount -t usbdevfs none /proc/bus/usb
```

pour pouvoir être utilisé. Il est également possible de le monter automatiquement dans le fichier `/dev/fstab`.

En fait, il existe deux types d'interfaces USB sur le marché : l'interface UHCI (abréviation de l'anglais « Universal Host Controller Interface »), spécifiée par Intel, et que les contrôleurs de la plupart des cartes mères utilisent (chipsets Intel et VIA), et l'interface OHCI (« Open Host Controller Interface »), spécifiée par Compaq, et qui est utilisée par les chipsets Compaq et ALi principalement. Ces deux interfaces sont incompatibles, vous devez donc sélectionner l'un des deux drivers UHCI disponibles (options « `UHCI (Intel PIIX4, VIA, ...) support` » et « `UHCI Alternate Driver (JE) support (NEW)` »)) ou le driver OHCI (option « `OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support` ».

Comme vous pouvez le constater dans le menu de configuration du noyau, un grand nombre de périphériques USB sont gérés par Linux. Vous devez bien entendu activer les options permettant de prendre en charge votre matériel. Il est conseillé d'inclure ces fonctionnalités sous forme de modules, afin de permettre le chargement dynamique des gestionnaires de périphériques. Ceci est nécessaire pour la configuration des périphériques connectés à chaud dans le système.

Il existe cependant une exception : les périphériques d'entrée tels que le clavier et la souris devront, si possible, être inclus directement au sein du noyau, afin d'éviter de se retrouver sans clavier et sans

souris en cas de problème dans la configuration du système. Notez qu'il existe deux types de drivers pour les périphériques d'entrée : un driver général (option « `USB Human Interface Device (full HID) support`), qui fonctionne avec tous les périphériques d'entrée du menu « `Input core support` », et des drivers simplifiés (options « `USB HIDBP Keyboard (basic) support` » et « `USB HIDBP Mouse (basic) support` »), que l'on utilisera pour réaliser des systèmes embarqués ou des noyaux légers. Il est évidemment recommandé d'utiliser le premier driver, si réellement le support du clavier USB est nécessaire (normalement, les claviers USB sont pris en charge par le BIOS de l'ordinateur et apparaissent exactement comme des claviers classiques pour le système d'exploitation).

6.16.2. Détection automatique des périphériques USB

Nous avons signalé ci-dessus que le bus USB permettait l'ajout et la suppression des périphériques USB à chaud, c'est à dire ordinateur allumé. C'est une nette progression par rapport aux anciens bus, qui n'ont jamais réellement accepté ce genre d'opération, tant du point de vue électrique (ils peuvent être détériorés) que du point de vue logiciel (le nouveau périphérique n'est pas détecté ou ne peut pas être configuré).

Les drivers USB de Linux sont capables de détecter l'apparition d'un nouveau périphérique sur le bus USB et de réserver les ressources systèmes nécessaires à son utilisation, mais ils ne peuvent pas faire grand chose de plus. En effet, la configuration d'un nouveau matériel requiert des opérations dépendantes de celui-ci, telles que :

- déterminer le driver à charger pour gérer ce nouveau périphérique ;
- charger et configurer ce driver ;
- signaler aux applications clientes l'apparition du périphérique.

Bien entendu, les opérations inverses doivent également être effectuées lorsqu'un périphérique est débranché.

Sous Linux, toutes ces opérations sont donc réalisées par un programme externe au noyau. La gestion des périphériques USB se fait donc comme suit :

- un mécanisme de surveillance du bus USB détecte les apparitions et disparitions des périphériques. Ce mécanisme est souvent implémenté sous la forme d'un *démon USB* ;
- ce démon appelle des programmes clients capables de réaliser la configuration des nouveaux périphériques. Ces programmes clients sont couramment appelés des « *agents USB* ».

Il existe deux solutions pour implémenter le démon USB. La première solution consiste à surveiller le système de fichiers virtuel `/proc/bus/usb/` pour détecter les apparitions et disparitions des périphériques USB. La deuxième solution est d'implémenter le démon USB directement au sein du noyau. Cette solution est plus efficace, parce que le démon fait dans ce cas partie intégrante du noyau et n'a donc pas à surveiller le système de fichiers virtuel `/proc/bus/usb/`. La suite de ce document

décrit cette méthode, parce qu'elle est la plus aboutie et plus générale (elle fonctionne également avec d'autres périphériques que les périphériques USB).

Pour activer le démon USB du noyau (nommé `khubd`), il faut simplement valider l'option « `Support for hot-pluggable devices` » du menu « `General setup` » dans la configuration du noyau. Cette option ajoute également une entrée `hotplug` dans le répertoire `/proc/sys/kernel/` du système de fichiers virtuels `/proc/` du noyau. Ce fichier contient le chemin sur le programme utilisateur qui sera appelé par le démon USB à chaque fois qu'un périphérique sera ajouté ou supprimé pendant le fonctionnement du système. Par défaut, ce chemin référence l'utilitaire `/sbin/hotplug`, mais il est possible d'en indiquer un autre en écrivant une nouvelle valeur dans ce fichier.

Le programme hotplug est généralement un script, capable de charger les modules pour la gestion de la plupart des périphériques USB. Il joue donc le rôle d'un agent universel. Cependant, il peut aussi exécuter des agents utilisateurs, s'ils existent, qui sont situés dans le répertoire `/etc/hotplug/`. Pour les périphériques USB, le script appelé se nomme `usb.agent`.

En fait, le noyau appelle le script hotplug dès qu'un périphériqe est ajouté ou supprimé, que ce soit un périphérique USB ou non. Ce mécanisme est donc absolument générique, et fonctionne également pour les cartes PCMCIA, les périphériques PCI connectables à chaud, et les interfaces réseau. Les scripts appelés par hotplug se nomment alors respectivement `usb.agent`, `pci.agent` et `net.agent`. Le script hotplug fait la distinction entre ces trois classes de périphériques grâce au premier paramètre que le noyau lui passe sur sa ligne de commande. Les valeurs de ce paramètres peuvent valoir respectivement `usb`, `pci` et `net`.

Les agents ont besoin d'informations complémentaires pour déterminer quel est le périphérique qui vient d'apparaître ou de disparaître. Ces informations sont passées à hotplug sous la forme de variables d'environnement, spécifiques à chaque classe de périphériques. Pour les périphériques USB, les variables d'environnement définies par le noyau sont les suivantes :

- ACTION, qui contient la description de l'événement qui s'est produit pour ce périphérique ;
- PRODUCT, qui contient les identificateurs du vendeur, du produit et du périphérique pour le périphérique considéré ;
- TYPE, qui contient la classe et la sous-classe du périphérique, ainsi que son protocole de communication ;
- INTERFACE, qui contient les paramètres de classe, sous classe et de protocole pour les interfaces, si le périphérique USB est de classe 0.

Si vous avez activé le support du système de fichiers virtuel USB, les variables suivantes seront également définies :

- DEVFS, qui contient le chemin sur la liste des gestionnaires de périphériques USB. Il s'agit normalement du fichier `/proc/bus/usb/drivers` ;
- DEVICE, qui contient le chemin sur le fichier spécial de périphérique pour le périphérique USB ;

Enfin, la variable d'environnement PATH est initialisée pour référencer les répertoires `/sbin/`, `/bin/`, `/usr/sbin/` et `/usr/bin/`, et la variable d'environnement HOME contient le chemin du répertoire racine.

Normalement, le script hotplug et les agents utilisateurs doivent être fournis avec votre distribution. Cependant, toutes les distributions ne sont pas encore prêtes pour le support de l'USB, et vous devrez installer ces scripts vous-même. Leur rédaction peut être extrêmement technique, je vous recommande donc d'utiliser les scripts disponibles sur le site de l'USB pour Linux (http ://www.linux-usb.org/policy.html).

6.17. Configuration de l'imprimante

Le sous-système d'impression de Linux est emprunté à un autre système Unix célèbre, BSD. Les commandes utilisateurs sont donc les mêmes, et son fonctionnement repose sur les mêmes principes. En général, les imprimantes sont connectées au port parallèle de l'ordinateur (sauf les imprimantes professionnelles, qui disposent d'une interface réseau). Il est donc nécessaire que la configuration du port parallèle soit correcte. Dans la suite, nous supposerons que l'imprimante est connectée sur le port `/dev/lp0` (premier port parallèle, soit l'équivalent de LPT1 sous DOS).

6.17.1. Filtres d'impression

Il est possible d'envoyer directement un fichier à imprimer à l'imprimante, en le recopiant directement sur le fichier spécial de périphérique de l'imprimante. Cette technique peut parfaitement convenir pour un fichier texte simple, mais elle pose quelques problèmes. Premièrement, elle ne fonctionne qu'avec des fichiers que l'imprimante peut comprendre, donc des fichiers contenant des instructions qui lui sont spécifiques. Mais le problème le plus important est sans doute le fait que cette technique n'est pas utilisable lorsque plusieurs programmes désirent envoyer un fichier en même temps à l'imprimante.

Le sous-système d'impression utilise donc un mécanisme de files d'attente pour les impressions (« spool » en anglais). Tout les travaux d'impression soumis sont placés dans une file d'attente, et attendent leur tour pour être envoyés à l'imprimante associée à cette file. Une fois que l'impression est terminée, les travaux sont supprimés de la file. Ainsi, un seul programme accède à l'imprimante : le sous système d'impression. Notez qu'il est possible de définir plusieurs files d'attentes sur une même imprimante, selon la nature du travail à effectuer. Par exemple, une file peut se charger des documents textes, et une autre des documents graphiques. Bien entendu, le sous-système d'impression contrôle tous les accès à l'imprimante et assure qu'un seul document est en cours d'impression à chaque instant.

D'autre part, le sous-système d'impression donne la possibilité de spécifier la nature des fichiers à imprimer, afin de les traduire dans un langage compréhensible par l'imprimante. Ce travail de conversion est réalisé par un ensemble de programmes que l'on appelle les *filtres*. Un filtre n'est en fait rien d'autre qu'un programme qui reçoit un fichier en entrée et qui fournit la traduction de ce fichier dans un autre format en sortie. Linux est fourni avec un certain nombre de filtres, qui permettent de gérer la plupart des formats de fichiers utilisés.

Parmi ces formats de fichiers, il en est un qui est plus important que les autres : PostScript. PostScript est un langage de description de pages inventé par Adobe, et que nombre d'imprimantes laser comprennent. En fait, il s'agit d'un véritable langage de programmation, qui permet de programmer les

périphériques dont la vocation est d'afficher ou d'imprimer des documents. Techniquement parlant, PostScript permet d'obtenir une qualité d'impression irréprochable, car c'est l'imprimante elle-même qui « dessine » la page à imprimer. Le problème est que toutes les imprimantes ne comprennent pas forcément le langage PostScript, notamment les imprimantes à jet d'encre. Pour ces imprimantes, il faut donc un filtre particulier, capable de convertir le PostScript dans le langage graphique de l'imprimante. Linux utilise pour cela un « interpréteur PostScript ». Un interpréteur PostSCript est un programme capable de comprendre les fichiers PostScript et de les convertir dans le format compris par l'imprimante. L'interpréteur couramment utilisé sous Linux est GhostScript, parce que c'est un logiciel libre (cependant, la version courante est toujours commerciale). Il est également capable d'afficher les fichiers PostScript sous XWindow, et de gérer la plupart des imprimantes du marché.

La plupart des autres filtres ont donc pour vocation de convertir un fichier en PostScript. L'avantage de cette technique est que toutes les imprimantes apparaissent comme étant des imprimantes PostScript pour les programmes désirant imprimer. Ainsi, leur programmation est beaucoup plus simple, puisqu'ils n'ont qu'un seul format à gérer. Bon nombre d'applications génèrent directement des fichiers PostScript, qui sont donc envoyés directement à GhostScript pour l'impression définitive.

Comme on le voit, pour Linux, le langage d'impression universel est le langage PostScript. Bien entendu, ceci est idéal si l'on dispose effectivement d'une imprimante PostScript, mais même dans le cas contraire, les impressions se font parfaitement grâce à PostScript.

6.17.2. Commandes d'impression

La commande d'impression sous Linux est la commande **lpr** (abréviation de l'anglais « Laser PRinter »). Cette commande est très simple à utiliser, comme le montre la syntaxe suivante :

```
lpr fichier
```

où `fichier` est le nom du fichier à imprimer. Cette commande se contente de placer le fichier à imprimer dans un répertoire affecté à la file d'attente des travaux d'impression. Le travail d'impression est ensuite effectué par le démon **lpd**, qui fait passer chaque fichier à imprimer à travers la série de filtre pour le convertir dans le langage de l'imprimante, puis qui alimente l'imprimante.

La liste des travaux d'impression en attente peut être consultée avec la commande **lpq**. Chaque travail en attente porte un numéro, grâce auquel on peut le manipuler. Entre autres opérations, il est possible de l'abandonner à l'aide de la commande **lprm**.

Enfin, pour consulter et contrôler l'état des files d'impression, on peut utiliser la commande **lpc**. Cette commande peut prendre des options en ligne de commande, afin de préciser l'opération à effectuer. Par exemple, l'option `status` permet d'obtenir l'état de chacune des files d'impression. Les autres options permettent d'arrêter le travail en cours, de le suspendre, de désactiver l'imprimante pour les travaux suivants, et inversement de relancer les travaux d'impression sur cette file.

6.17.3. Configuration des files d'impression

Les distributions modernes fournissent toutes un outil permettant d'effectuer la configuration du sous-système d'impression. Il est évidemment recommandé d'utiliser ces outils, car le résultat est assuré et la vie en est d'autant plus facile. De plus, les distributions peuvent fournir des filtres complémentaires, que seuls ces outils connaissent et sont capables d'installer. Toutefois, il est bon de savoir quels sont les mécanismes mis en ½uvre lors de l'impression d'un document. Nous allons donc nous y intéresser brièvement dans ce paragraphe.

Le démon lpd utilise le fichier de configuration `/etc/printcap` pour déterminer l'ensemble des files d'impression existantes et quel filtres doivent être utilisés. Chaque file est décrite par une ligne et une seule de ce fichier. Ces lignes sont constituées de divers champs, séparés par des deux points (':'). Comme ces lignes peuvent être relativement longues, elles peuvent être réparties sur plusieurs lignes physiques en plaçant le caractère d'échappement '\' à la fin de chaque ligne, sauf la dernière.

Le premier champ de la description d'une file d'attente est une liste des noms sous lesquels cette file sera connue. Les différents noms sont écrits les uns à la suite des autres, séparés par une barre verticale (caractère '|').

Les champs suivants décrivent l'imprimante à utiliser, ainsi que les options générales de la file d'attente. Ces champs utilisent tous la même syntaxe :

```
option = valeur
```

Il existe un grand nombre d'options, nombre d'entre elles sont facultatives. Cependant, il est impératif que le démon **lpd** puisse trouver l'imprimante à utiliser. Par conséquent, il faut lui fournir au moins l'une des deux série d'options suivantes :

- l'option `lp` permet de spécifier le fichier spécial de périphérique auquel l'imprimante est connectée ;
- les options `rm` et `rp` permettent de spécifier respectivement le nom d'un serveur d'impression distant (« remote » en anglais) et de l'imprimante à utiliser sur ce serveur (« remote printer »).

Le démon **lpd** doit également connaître le répertoire dans lequel les travaux en attente seront stocké (répertoire dit de « spool »). Ce répertoire peut être défini avec l'option `sd`.

D'autres options peuvent être utiles, comme `sh` (cette option ne prend pas de valeur), qui permet de supprimer la page de garde au début de chaque impression, et `mx`, qui permet de spécifier la taille maximale des travaux d'impression soumis. Cette dernière option permet de fixer des quotas d'impression selon la taille des documents, afin de donner la possibilité aux autres documents d'être imprimés. Cette option utilise une syntaxe particulière :

```
mx#taille
```

où `taille` est la taille maximale autorisée, exprimée en kilo-octets. Le fait de spécifier une taille nulle permet de supprimer ce contrôle.

L'exemple ci-dessous correspond à la définition d'une file d'attente locale élémentaire :

```
ascii|lp :lp=/dev/lp :sd=/var/spool/lpd/ascii :mx#0 :sh
```

Comme vous pouvez le constater, il n'y a aucune spécification des filtres d'impression à utiliser dans cet exemple. Les travaux sont donc directement envoyés à l'impression, sans traduction préalable. Il est donc nécessaire qu'ils soient déjà au format de l'imprimante. Si l'on veut utiliser des filtres d'impression, il faut utiliser l'une des options `if`, `cf`, `df`, `gf`, `nf`, `rf`, `tf` ou `vf`. Chacune de ces options permet de spécifier la ligne de commande d'un filtre d'impression spécifique. Le choix du filtre utilisé pour un travail d'impression est effectué lors de l'appel à la commande **lpr**, à l'aide d'une option en ligne de commande. Le filtre `if` est le filtre par défaut, il n'y a donc besoin d'aucune option pour l'utiliser. Les autres filtres peuvent être sélectionnés respectivement avec les options `-c`, `-d`, `-g`, `-n`, `-f`, `-t` et `-v`.

Comme on le voit, le sous-système d'impression ne reconnaît pas automatiquement le format de fichier utilisé. D'autre part, le nombre de filtres utilisables est limité à 8, ce qui peut ne pas suffire étant donné la prolifération des formats de fichiers. Pour résoudre ce problème, les distributions utilisent souvent un filtre générique (utilisé en tant que filtre par défaut), qui est capable de reconnaître le format du fichier à imprimer et de le diriger vers un autre filtre ou une série de filtres. Comme on l'a vu ci-dessus, l'ultime filtre utilisé est en général l'interpréteur GhostScript. Ainsi, il n'y a plus de limite sur le nombre de filtres utilisables, et les filtres sont sélectionnés automatiquement en fonction de la nature du document à imprimer. Malheureusement, ces filtres évolués ne sont pas standard, et chaque distribution est susceptible de fournir sa propre version. C'est pour cette raison qu'il est recommandé d'utiliser les programmes de configuration fournis par ces distributions pour installer l'imprimante.

Chapitre 7. Configuration du réseau

Linux est un système d'exploitation fabriqué par l'Internet et pour l'Internet. Inutile de préciser que c'est l'un des meilleurs systèmes pour gérer et exploiter un réseau. Certains ne l'utilisent d'ailleurs que pour cela, et profitent de ses excellentes performances sur les petites machines afin de récupérer du matériel autrement voué à la casse. En fait, les fonctionnalités réseau de Linux sont si nombreuses que j'ai été obligé d'y consacrer un chapitre à part entière.

La configuration d'un réseau est une opération qui nécessite quelques connaissances théoriques sur le fonctionnement des réseaux TCP/IP. Ces informations sont assez techniques, mais indispensables pour bien configurer les services réseau de toute machine connectée, et pas seulement les machines fonctionnant sous Linux. Il n'est en effet pas rare de trouver des réseaux de machines fonctionnant sur des systèmes dont la configuration est supposée être plus simple, mais dont l'organisation est une hérésie absolue et qui risque de nécessiter une remise à plat complète à chaque interconnexion.

Cette section va donc donner quelques explications sur les notions fondamentales des réseaux informatiques. Il traitera ensuite de la configuration des réseaux locaux, puis celle des connexions temporaires à Internet. Les services réseau évolués tels que le partage de connexion à Internet et la création d'un serveur de fichiers seront finalement traités en fin de chapitre.

7.1. Notions de réseau TCP/IP

7.1.1. Généralités sur les réseaux

Un réseau n'est en général rien d'autre qu'une interconnexion entre plusieurs machines et qui leur permet d'échanger des informations. Il existe de nombreux moyens de réaliser cette interconnexion, qui utilisent parfois des supports physiques variés. Les techniques les plus utilisées sont la liaison radio et la liaison par câble. Cette dernière technique comprend diverses variantes, dont les réseaux Ethernet, TokenRing et simplement la liaison téléphonique.

Il est évident que la manière d'envoyer et de recevoir des informations est différente pour ces différents supports physiques, parce qu'elle dépend tout simplement des possibilités techniques offertes par la technologie sous-jacente utilisée. Cependant, il est très courant de découper les informations à échanger en *paquets*, qui sont ensuite transmis sur le réseau. Ces paquets peuvent être de tailles variées, et contenir des informations utiles à la gestion du réseau. L'information la plus importante est sans doute celle permettant de connaître la machine destinataire du paquet. On l'appelle l'*adresse* de la machine cible. En général, les paquets contiennent également l'adresse de la machine source, afin qu'une réponse puisse lui-être envoyée.

Du fait de la diversité des supports physiques de réseau, il n'est pas simple d'écrire une application réseau qui puisse travailler dans des environnements réseau hétérogènes. Ceci supposerait de connaître les protocoles de communication pour chaque type de réseau, ce qui compliquerait à l'infini le moindre programme et le rendrait inutilisable avec les nouveaux réseaux. Par conséquent, cette tâche ingrate a été reléguée au plus profond des drivers spécifiques au support physique. Les applications quant à elles utilisent un protocole de communication plus évolué, dont le but est d'assurer

l'interopérabilité des différents supports physiques. Ce protocole utilise toujours des paquets et une notion d'adresse, mais cette fois ces informations sont standardisées et utilisables par toutes les applications. Les paquets de ce protocole sont stockés dans les paquets des réseaux physiques et transmis tels quels. Ils peuvent éventuellement être découpés en sous-paquets dans le cas où la taille des paquets du réseau serait trop petite pour les contenir. Cette technique s'appelle l'encapsulation d'un protocole dans un autre protocole.

7.1.2. Le protocole IP

Les machines Unix utilisent toutes le protocole de communication de bas niveau IP (« Internet Protocol »). Ce protocole a été inventé pour permettre l'interconnexion d'un grand nombre de réseaux physiques différents (le nom d'Internet provient d'ailleurs de cette caractéristique : « INTERconected NETworks »). Il permet de transmettre des informations de manière uniforme sur tous ces réseaux physiques. Ainsi, les programmes qui utilisent IP ne voient pas les spécificités des différents réseaux physiques. Pour eux, il ne semble y avoir qu'un seul réseau physique, dont le protocole de communication de base est IP. Autrement dit, les applications qui utilisent le réseau se contentent d'utiliser le protocole IP, et n'ont plus à se soucier de la manière dont il faut formater et transmettre les informations sur chaque support physique du réseau. Ce genre de détails est laissé aux drivers réseaux de chaque machine et aux passerelles reliant les divers réseaux physiques.

Comme il l'a déjà été dit ci-dessus, le protocole IP utilise des adresses pour identifier les machines sur les réseaux. Les adresses IP sont codées sur quatre octets (nombre binaire à huit chiffres, permettant de représenter des valeurs allant de 0 à 255), chacun définissant une partie du réseau. Ces adresses sont donc utilisées un peu comme les numéros de téléphone : le premier octet défini le numéro d'un « super réseau » dans lequel le correspondant se trouve (ces « super réseaux » sont appelés les réseaux de *classe A*), le deuxième octet défini le numéro du sous-réseau dans le super réseau (ces sous réseaux sont appelés réseaux de *classe B*), le troisième octet défini encore un sous sous-réseau (réseaux dits de *classe C*) et le quatrième octet donne le numéro de la machine dans ce sous sous-réseau. Cette numérotation permet d'affecter des adresses similaires pour les différentes machines d'un réseau, et de simplifier ainsi la gestion de ce dernier. Elle dispose en revanche d'un inconvénient majeur : beaucoup d'adresses sont gaspillées, car il n'y a pas suffisamment de réseaux de classe A d'une part, et qu'on ne peut pas mélanger les machines de deux sous-réseaux dans un même réseau de classe A d'autre part. Si l'on reprend la comparaison avec les numéros de téléphone, il y a énormément d'abonnés dont le numéro commence par 01, mais beaucoup moins dont le numéro commence par 02, et quasiment aucun dont le numéro commence par 08. Si on venait à manquer de place dans la liste des numéros commençant par 01, on ne pourrait pas pour autant utiliser les numéros commençant par 02 pour des raisons de zones géographiques. C'est la même chose pour les adresses IP, sauf que les les zones géographiques sont remplacées par des sous-réseaux. Le problème est que, malheureusement, on commence à manquer d'adresses disponibles (alors qu'il y en a plein de libres mais inutilisables parce qu'elles se trouvent dans d'autres sous-réseaux !). Il va donc falloir effectuer une renumérotation d'ici peu, exactement comme il y en a déjà eu dans le monde de la téléphonie…

> **Note:** Le protocole IPv6, qui remplacera le protocole IP classique (encore appelé IPv4), a pour but de résoudre les limitations du protocole IP utilisé actuellement. Les adresses du protocole IPv6

> sont codées sur 16 octets, ce qui résoudra définitivement le problème du manque d'adresses. De plus, les services modernes que sont l'authentification de l'émetteur, ainsi que la qualité de service (c'est à dire la garantie du délai de transmission des données, garantie nécessaire pour transmettre de façon correcte les flux multimédia tels que le son et la vidéo en temps réel) sont fournis par IPv6. Bien entendu, Linux est déjà capable d'utiliser IPv6 (combien de systèmes peuvent aujourd'hui l'affirmer ?) ! Notez toutefois que pour cela, il faut recompiler le noyau et toutes les applications réseau du système, ce qui est tout de même très lourd. Par conséquent, il vaut mieux se contenter du protocole IP actuel. Malgré ses limitations, ce protocole reste sans doute le meilleur protocole réseau du monde, car il allie souplesse et fonctionnalité. Il est difficilement concevable de créer un réseau aussi grand qu'Internet avec les autres protocoles existant sur le marché…

Les adresses IP sont donc parfaitement définies à l'aide de leurs quatre nombres, que l'on note les uns à la suite des autres et en les séparant d'un point. Comme on l'a vu, les adresses IP sont classées en sous-réseaux, de classe A, B et C. Les adresses des réseaux de classe C ont leur trois premiers nombres fixés, et seul le quatrième nombre change pour chaque machine du réseau. De la même manière, les réseaux de classe B ont leurs deux premiers nombres fixés, et seuls les deux derniers nombres permettent de distinguer les différentes machines du réseau. Enfin, les réseaux de classe A n'ont de fixé que leur première composante, les autres sont libres. Il est donc clair qu'il existe peu de réseaux de classe A, mais que ce sont de très gros réseaux (ils peuvent contenir jusqu'à 16 millions de machines !). En revanche, il existe beaucoup plus de réseaux de classe C, dont la taille est plus modeste (seulement 256 machines).

Pour un réseau donné, les adresses ont donc toutes la même forme. Les premiers octets des adresses du réseau sont toujours les mêmes (ce peut être le premier octet pour les réseaux de classe A, les deux premiers pour les réseaux de classe B ou les trois premiers pour les réseaux de classe C). On peut donc définir la notion d'adresse de réseau, qui est l'adresse IP d'une machine du réseau dont les parties variables ont pour valeur 0. Par exemple, si une machine d'un réseau de classe C a pour adresse 192.168.1.15, alors l'adresse de son sous-réseau est 192.168.1.0. Ceci signifie que toutes les machines de ce réseau auront une adresse de la forme « 192.168.1.xxx ».

Un réseau n'appartient qu'à une et une seule classe. Les adresses IP sont réparties sur les différentes classes de réseau, selon la valeur des bits de poids fort de leur premier octet. Par exemple, les réseaux de classe A sont identifiables au fait que le bit de poids fort de leur adresse est nul. Les adresses de réseau valides pour les réseaux de ce type sont donc les adresses comprises entre 0.0.0.0 et 127.0.0.0. Il n'existe donc que 128 réseaux de classes A en tout et pour tout. Les autres réseaux ont donc le bit de poids fort de leur adresse fixé à 1, et c'est le deuxième bit de poids fort qui est utilisé pour distinguer les réseaux de classe B des autres. Les réseaux de classe B utilisent toujours la valeur 0 pour ce bit, leurs adresses varient donc entre 128.0.0.0 et 191.255.0.0. De même, les réseaux de classe C utilisent la valeur 1 pour le deuxième bit de leur adresse, et ont nécessairement un troisième bit nul. Leurs adresses vont donc de 192.0.0.0 à 223.255.255.0. Les adresses pour lesquelles le troisième bit (en plus des deux premiers) est à 1 sont réservées (soit pour une utilisation ultérieure, soit pour s'adresser à des groupes d'ordinateurs en multicast) et ne doivent pas être utilisées. Il s'agit des adresses 224.0.0.0 à 255.255.255.255. Cette dernière adresse a une signification spéciale et permet de s'adresser à tous les ordinateurs d'un réseau.

Il est possible de déterminer l'adresse du réseau auquel une machine appartient en utilisant ce qu'on

appelle le *masque de sous-réseau*. Le masque de sous-réseau est une série de quatre nombres ayant le même format que les autres adresses IP, mais dont les composantes ne peuvent prendre que la valeur 0 ou la valeur 255, les 255 devant nécessairement apparaître en premier. Les composantes des adresses IP qui correspondent à la valeur 255 dans le masque de sous-réseau font partie de l'adresse dudit sous-réseau. Les composantes qui correspondent à la valeur 0 dans le masque de sous-réseau n'en font pas partie, et varient pour chaque machine du réseau. Pour reprendre l'exemple précédent, si une machine a pour adresse IP 192.168.1.15 et que son masque de sous-réseau est 255.255.255.0, alors l'adresse de son réseau est 192.168.1.0. Si le masque de sous-réseau avait été 255.255.0.0 (typiquement le masque d'un réseau de classe B), l'adresse du réseau aurait été 192.168.0.0. Comme on le voit, le masque de sous-réseau est utilisé par le système pour déterminer rapidement l'adresse de sous-réseau d'une machine à partir de son adresse IP.

Les adresses IP ne sont pas attribuées aux machines au hasard. Il est évident que chaque machine doit avoir une adresse unique, et que son adresse doit appartenir à la plage d'adresses utilisée pour le sous-réseau dont elle fait partie. Pour cela, la classe des réseaux, ainsi que les adresses qu'ils utilisent, sont attribués par l'IANA, un organisme de gestion de l'Internet. Le rôle de l'IANA (abréviation de l'anglais « Internet Assigned Numbers Authority ») est essentiellement d'assurer l'unicité des adresses IP sur l'Internet. Cependant, certaines adresses sont librement utilisables pour les réseaux locaux qui ne sont pas connectés directement à l'Internet. Les paquets utilisant ces adresses sont assurés de ne pas être transmis sur Internet, et peuvent donc être utilisées par quiconque. Les plages d'adresse réservées sont les suivantes :

Tableau 7-1. Plages d'adresses IP réservées pour un usage personnel

Classe de réseau	Adresses de réseau réservées
A	**10.0.0.0**
B	**172.16.0.0 à 172.31.0.0**
C	**192.168.0.0 à 192.168.255.0**

Ainsi, un réseau de classe A (d'adresse 10.0.0.0), 16 réseaux de classe B (les réseaux 172.16.0.0 à 172.31.0.0) et 255 réseaux de classe C (d'adresses 192.168.0.0 à 192.168.255.0) sont disponibles. Vous pouvez donc les utiliser librement.

Il est également possible de configurer les machines pour qu'elles récupèrent leur adresses IP auprès d'un serveur à l'aide du protocole DHCP (abréviation de l'anglais « Dynamic Host Control Protocol »). Cette technique est très intéressante quand on dispose d'un grand nombre de machines qui ne sont pas toujours toutes connectées à un réseau. Il est donc possible de redistribuer les adresses IP d'un stock d'adresses en fonction des machines qui se connectent, et d'économiser ainsi les précieuses adresses. En revanche, elle n'est pas appropriée pour les serveurs qui sont couramment accédés par des postes clients, et qui doivent donc avoir une adresse IP fixe. Les fournisseurs d'accès à Internet distribuent de cette manière les adresses IP aux postes de leurs clients lorsque ceux-ci se connectent, et ils récupèrent les adresses à leur déconnexion.

Certaines adresses IP ont une signification particulière et ne peuvent pas être attribuées à une machine. Par exemple l'adresse 127.0.0.1 représente, pour une machine, elle-même. Cette adresse est souvent

utilisée pour accéder à un programme réseau sur la machine locale. Elle fait partie du sous-réseau de classe A 127.0.0.0, qui ne comprend pas d'autres adresses. De plus, les adresses dont les derniers nombres (c'est à dire les nombres qui ne font pas partie de l'adresse du réseau) se terminent par 0 ou 255 sont réservées pour l'envoi des paquets à destination de tout le monde sur le réseau (émission dite « broadcast »). Par exemple, les adresses 192.168.1.0 et 192.168.1.255 ne peuvent pas être affectées à une machine. Ce sont typiquement ces adresses qui sont utilisées par le protocole DHCP pour émettre des requêtes sur le réseau alors que la machine n'a pas encore d'adresse fixe.

Il est important de savoir que par défaut, une machine ne communiquera qu'avec les machines de son propre réseau. C'est à dire que si une machine utilise l'adresse IP 192.168.1.15 et que son masque de sous-réseau est 255.255.255.0, elle ne pourra contacter que des machines dont l'adresse est de la forme 192.168.1.xxx. Elle ne pourra donc pas voir par exemple une machine dont l'adresse IP est 192.168.0.2. Ceci ne signifie pas que l'on doive toujours utiliser le masque 0.0.0.0 pour voir toutes les machines du monde, mais plutôt que la machine 192.168.0.2 ne fait pas partie, par définition, du même réseau physique que la machine 192.168.1.15. Il est donc inutile de chercher à la contacter (et mettre le masque de sous-réseau à 0.0.0.0 ne servirait évidemment à rien).

Arrivé à ce stade des explications, je sens venir la question suivante : « ? ! Euhhh… Mais alors, comment peut-on voir les machines sur Internet ? Je n'ai pas de réseau, et quand je me connecte à Internet, je peux y accéder ! Et même si j'avais un réseau, elles ne feraient pas partie de mon réseau… ».

Explications :

- premièrement, vous avez un réseau, même si vous ne le savez pas. Toute machine capable de se connecter à l'Internet appartient au moins à son propre réseau virtuel, sur laquelle elle est la seule machine, et où elle a l'adresse 127.0.0.1 ;
- deuxièmement, effectivement, les machines qui se trouvent sur Internet n'appartiennent pas à votre réseau, que celui-ci existe effectivement ou soit virtuel ;
- troisièmement, toutes les informations que vous envoyez et recevez transitent par un seul ordinateur, celui de votre fournisseur d'accès à Internet.

C'est ici qu'intervient la notion de *passerelle* (« Gateway » en anglais).

Une passerelle est une machine qui appartient à deux réseaux physiques distincts, et qui fait le lien entre les machines de ces deux réseaux. Les ordinateurs des deux réseaux peuvent communiquer avec la passerelle de part et d'autre, puisqu'elle appartient aux deux réseaux. Les ordinateurs de chaque réseau transmettent à cette passerelle tous les paquets qui ne sont pas destinés à une machine de leur propre réseau. Celle-ci se charge simplement de transférer ces paquets aux machines de l'autre réseau.

Lorsque vous vous connectez à Internet, vous ne faites rien d'autre que de créer un réseau (dont le support physique est la ligne téléphonique), et vous utilisez l'ordinateur que vous avez appelé comme passerelle par défaut. Tous les paquets destinés à un autre réseau que le vôtre (donc, en pratique, tous les paquets si vous n'avez pas de réseau local) sont donc envoyés au fournisseur d'accès à Internet, qui se charge de les transmettre aux autres ordinateurs. Notez que la passerelle peut elle-même transmettre ces paquets à une autre passerelle, et ainsi de suite, jusqu'à ce que la destination soit atteinte.

Dans le cas d'un particulier, le choix du réseau sur lequel les paquets doivent être transmis est très facile à faire, puisqu'en général, un paquet est soit à destination de la machine locale, soit à destination d'une machine sur Internet. Pour un paquet destiné à la machine locale, le réseau virtuel local est utilisé. Tous les autres paquets sont envoyés sur la connexion Internet. Cependant, il peut arriver qu'une machine ait le choix entre plusieurs réseaux différents pour envoyer un paquet dont la destination n'est pas sur son propre réseau. Par exemple, les passerelles des fournisseurs à Internet peuvent être elles-mêmes connectées à différents autres réseaux, qui sont eux-mêmes connectés à d'autres réseaux. L'ensemble des réseaux empruntés par un paquet dans son voyage constitue ce qu'on appelle sa *route*.

Chaque passerelle contribue donc à déterminer la route des paquets en choisissant, pour chaque paquet, l'interface réseau à utiliser pour transmettre ce paquet. Ce choix est fait en suivant un certain nombre de règles basées sur l'adresse destination des paquets. Par exemple, si une passerelle reçoit un paquet dont l'adresse destination est 125.46.10.15, et qu'elle est elle-même connectée à un réseau possédant cette machine, elle transmettra bien évidemment ce paquet sur ce réseau. Si en revanche elle ne peut localiser la machine cible sur l'un de ses réseaux, elle l'enverra à une autre passerelle. Le choix du réseau utilisé est en général déterminé par des règles de la forme suivante : « Tous les paquets destinés aux sous-réseau 125.46.0.0 doivent être envoyés vers la passerelle 137.49.20.1 du réseau 137.49.20.0 ».

Ainsi, si la passerelle est connectée à trois réseaux d'adresses respectives 192.168.1.0, 137.49.20.0 et 209.70.105.10, elle transférera le paquet à destination de 125.46.10.15 à la passerelle 137.49.20.1.

L'ensemble des règles utilisées par les passerelles sont stockées dans ce qu'on appelle des *tables de routage*. Les tables de routage peuvent être configurées statiquement dans la configuration des passerelles, c'est à dire initialisées dans les fichiers de configuration et ne jamais être modifiées. Cette technique convient parfaitement pour les petits réseaux, cependant, elle se révèle insuffisante sur les passerelles d'Internet. En effet, si les réseaux empruntés par une route sont saturés, voire hors service, il peut être intéressant d'utiliser une autre route. Ces passerelles utilisent donc des tables de routage dynamiques, qui sont automatiquement mises à jour en fonction de l'état du réseau. Bien que Linux puisse parfaitement effectuer ce type de routage, c'est une configuration réservée à un usage spécifique de Linux. Ce document ne traitera donc pas du routage dynamique.

Le problème des adresses IP est qu'elles ne sont pas très parlantes pour les êtres humains : que peut donc signifier 192.147.23.2 ? Pas grand chose... C'est pour cela qu'on affecte des noms de machines plus humains, comme par exemple `krypton.andromede.galaxie`. Ces noms suivent une convention de nommage bien précise. En général, le premier nom est le nom de la machine elle-même, et la suite du nom constitue ce qu'on appelle le *domaine* dans laquelle la machine se trouve. Ainsi, dans l'exemple précédent, `krypton` est le nom d'une machine, et `andromede.galaxie` est le nom de son domaine. En général, il existe plusieurs machines dans un même domaine, on pourrait donc également avoir `altair.andromede.galaxie` par exemple (malheureusement pour mon exemple, l'étoile Altaïr ne se trouve pas dans la galaxie d'Andromède, mais bon...). Souvent, les noms de domaines sont des noms de sociétés. La dernière partie du nom de domaine permet également de déterminer la nature du domaine, ou sa localisation. Par exemple, l'extension `.com` indique clairement que le domaine est de nature commerciale (de surcroît, il s'agit sans doute d'une société américaine, l'extension `.com` étant réservée aux États-Unis). De même, l'extension `.gov` est utilisée pour les organismes gouvernementaux américains, et l'extension `.edu` pour les universités ou les écoles américaines. L'extension

`.org` est utilisée pour les organisations non commerciales. Enfin, les noms de domaines des autres pays que les États-Unis utilisent quasiment systématiquement une extension indiquant leur pays d'origine. Ainsi, `.fr` représente la France, `.uk` les Royaumes-Unis, etc... Notez que la notion de domaine est a priori distincte de la notion de réseaux.

La question technique qui se pose avec ces conventions de nommage « humaines » est de savoir comment déterminer, à partir d'un nom littéral, l'adresse IP de la machine correspondante. Cette opération n'est pas simple, et en fait elle est effectuée de deux manières :

- soit la machine locale demande à une autre machine qu'elle connaît d'effectuer la recherche de l'adresse du correspondant ;
- soit elle dispose d'une liste de noms et d'adresses lui permettant de déterminer directement les adresses de ses interlocuteurs.

La première solution utilise ce qu'on appelle un *serveur de noms* (« DNS » en anglais, abréviation de « Domain Name Service »), qui connaît toutes les machines du réseau soit directement, soit indirectement (donc via un autre DNS). L'opération qui consiste à retrouver l'adresse IP d'une machine à partir de son nom s'appelle la *résolution de noms de domaines*. Il est évident que si le serveur de nom ne peut être contacté, il sera impossible d'utiliser les noms de machines. En revanche, il sera toujours possible d'utiliser leurs adresses IP directement, si on les connaît. La deuxième solution ne souffre pas de ce défaut, cependant, elle nécessite de mettre à jour la liste des noms sur tous les postes régulièrement, ce qui est beaucoup plus complexe que la gestion centralisée d'un DNS. On ne l'utilisera donc que pour les petits réseaux.

7.1.3. Le protocole TCP

Nous avons vu que le protocole IP fournit les briques de bases de toute la communication réseau sous Unix (et Internet). Ses principales fonctionnalités sont le découpage des informations en paquets de taille suffisamment petite pour pouvoir transiter sur tous les types de réseaux physiques, la gestion des destinations des paquets à l'aide des adresses IP, et le choix de la route permettant d'acheminer les paquets à destination. En revanche, il est incapable d'assurer les services plus évolués comme la gestion de l'ordre d'arrivée des informations et la gestion de la fiabilité du transfert des informations. C'est donc à des protocoles plus évolués, eux-mêmes encapsulés dans IP, d'effectuer ces tâches. L'un des protocoles les plus importants est le protocole TCP (abréviation de l'anglais « Transfer Control Protocol »). Ce protocole se charge d'établir les connexions entre deux ordinateurs, et assure la fiabilité des informations transmises et l'arrivée des informations dans leur ordre d'envoi. Il existe d'autres protocoles, moins connus que TCP mais tout aussi importants. On retiendra en particulier les deux protocoles suivants :

- UDP (abréviation de l'anglais « User Datagram Protocol »), qui permet d'émettre des *datagrammes* sur le réseau, qui sont de simples paquets de données (c'est un protocole semblable à IP destiné aux applications) ;

- ICMP (abréviation de « Internet Control Message Protocol »), qui est utilisé essentiellement pour transmettre des messages de contrôle du fonctionnement des autres protocoles (il est donc vital).

Les services réseaux des machines sont organisés en couche logicielles, qui s'appuient chacune sur la couche inférieure. Ainsi, TCP utilise IP, qui lui-même utilise le driver qui gère l'interface réseau. Du fait que ces couches s'appuient les unes sur les autres, on appelle souvent l'ensemble de ces couches une *pile* (« stack » en anglais). Vous avez peut-être déjà entendu parler de la *pile TCP/IP*. Lorsqu'un service réseau d'une machine n'est plus accessible, il se peut que ce service réseau ait planté. Si tout un ensemble de services réseau ne fonctionne plus, c'est certainement une des couches logicielle qui est plantée. Par exemple, une machine peut répondre à la commande **ping** (classiquement utilisée pour tester les connexions réseaux) et ne plus accepter la plupart des connexions réseau. Ceci signifie simplement que la couche TCP ne fonctionne plus, et que la couche ICMP (utilisée par **ping**) est toujours valide. Évidemment, si la couche IP tombe en panne, la machine ne sera plus accessible du tout, sauf avec d'autres protocoles réseaux complètement différents (IPX, Appletalk, etc…).

Seul les mécanismes du protocole TCP seront détaillés dans la suite de ce document. Le protocole TCP est en effet utilisé par un grand nombre de services, ce qui en fait certainement le plus connu.

Le protocole TCP utilise la notion de *connexion*. Une connexion est un canal de communication établi entre deux processus par TCP. Comme les processus sont susceptibles d'utiliser plusieurs connexions simultanément, TCP fournit la possibilité de les identifier par un numéro unique sur la machine, compris entre 0 et 65535. Chaque numéro identifie ce qu'on appelle un *port* TCP. Quand un processus désire établir une connexion avec un autre, il utilise un de ses ports et essaie de se connecter sur le port du deuxième processus.

Il faut bien comprendre que les deux numéros de ports utilisés ne sont a priori pas les mêmes pour les deux processus. Évidemment, il est nécessaire que les processus clients connaissent les numéros de port des processus serveurs auxquels ils se connectent. Les numéros de ports sont donc affectés à des services bien définis, et les serveurs qui fournissent ces services doivent bien entendu utiliser ces numéros de ports. Par conséquent, il est possible de déterminer de manière unique un programme serveur sur un réseau avec l'adresse IP de la machine sur laquelle il fonctionne et le numéro de port qu'il écoute pour les connexions extérieures. Les clients qui se connectent savent donc parfaitement à quel service ils accèdent lorsqu'ils choisissent le numéro de port destination. Leur propre numéro de port est en général choisi par le système, afin d'éviter tout conflit avec un autre processus de la même machine.

Une fois établie, une connexion TCP permet d'effectuer des communications bidirectionnelles. Ceci signifie que le clients et le serveur peuvent tous deux utiliser cette connexion pour envoyer des données à l'autre. Le client envoie ses données sur le port destination du serveur, et le serveur peut renvoyer les résultats au client en utilisant le port de celui-ci. Les paquets TCP disposent donc toujours d'un port *source* (c'est à dire le port TCP de l'émetteur), et d'un port *destination* (le port du récepteur). Ainsi, le récepteur peut renvoyer sa réponse en utilisant le port source comme port de destination du paquet renvoyé, et inversement.

Comme il l'a déjà été dit, le protocole TCP s'assure que les informations transmises arrivent à bon port (c'est le cas de le dire !). Pour cela, il utilise un mécanisme d'accusés réception, qui indiquent à l'émetteur si le destinataire a bien reçu chaque paquet envoyé. Si l'accusé réception n'est pas reçu

dans un temps fixé, le paquet est ré-émis. Un paquet reçu en double à cause d'un retard dans la communication de l'accusé réception est tout simplement ignoré. La fiabilité des informations est également assurée. Cette fiabilité est réalisée par un mécanisme de somme de contrôle. Si le récepteur constate que la somme de contrôle des données reçues n'est pas celle que l'émetteur a calculé, il rejette le paquet parce qu'il sait que les informations ont été corrompues pendant la transmission. Enfin, TCP s'assure que les informations émises en plusieurs passes sont bien reçues dans leur ordre d'émission. Cette réorganisation se fait grâce à une numérotation des paquets (cette numérotation sert également à détecter les paquets reçus en double). Elle peut paraître inutile, mais la vitesse d'arrivée des paquets est hautement dépendante de la route IP qu'ils prennent pour parvenir à destination. Les paquets qui arrivent en avance sont donc mémorisés jusqu'à ce que tous les paquets qui les précèdent soient reçus.

7.1.4. Les protocoles de haut niveau

TCP fournit donc les fonctionnalités requises pour la plupart des services réseaux existant. Il est logique que ceux-ci s'appuient sur lui pour effectuer leur travail. Cependant, il ne se charge que de communiquer les informations, rien de plus. Des protocoles de plus haut niveau ont donc été créés. Leur valeur ajoutée provient souvent du formatage et de la structuration des flux de données échangés.

La plupart des services réseaux définissent donc un protocole qui leur est propre. Il est d'ailleurs assez courant de confondre le service et le protocole, tellement ils sont intrinsèquement liés. Ainsi, le service FTP utilise le protocole FTP (« File Transfer Protocol », protocole de transfert de fichiers), les serveurs Internet utilisent essentiellement le protocole HTTP (« Hyper Text Transfer Protocol », protocole de transfert d'hypertexte), le service de courrier électronique utilise les protocoles POP (« PostOffice Protocol ») pour la réception des courriers et SMTP (« Simple Mail Transfer Protocol ») pour leur envoi.

La liste de ces protocoles n'est pas exhaustive, et de nouveaux services apparaissent régulièrement. Les protocoles de haut niveau ne seront donc pas traités dans ce document.

7.2. Configuration du réseau sous Linux

La configuration du réseau nécessite donc la configuration du protocole IP et des services TCP, UDP et ICMP (entre autres). Cette opération se fait en définissant l'adresse IP le masque de sous-réseau et les routes à utiliser. Vient ensuite la configuration du nom de la machine locale, de son domaine, des noms de machines qu'elle peut résoudre elle-même et des serveurs de DNS qu'elle doit utiliser pour les autres noms.

Il est quasiment certain que votre distribution dispose d'un outil permettant d'effectuer la configuration du réseau simplement. Connaissant à présent la signification des termes utilisés dans les réseaux TCP/IP, vous devriez pouvoir parvenir à une configuration valide relativement simplement. Il est fortement recommandé de consulter la documentation de votre distribution. Les commandes de configuration du réseau sont souvent appelées dans les scripts de démarrage de la machine, ou dans les scripts de changement de niveau d'exécution. Toutefois, il peut être utile de connaître ces commandes,

ne serait-ce que pour comprendre comment votre système fonctionne. Cette section a donc pour but de vous présenter ces commandes, ainsi que les principaux fichiers de configuration du réseau utilisés sous Linux.

7.2.1. Configuration des interfaces réseau

La principale commande permettant de configurer le réseau est la commande **ifconfig**. Comme son nom l'indique (« InterFace CONFiguration »), elle permet de configurer les interfaces réseaux de la machine. Il faut savoir qu'il existe plusieurs types d'interfaces réseaux. Les plus courants sont les trois types d'interfaces suivants :

- l'interface loopback, qui représente le réseau virtuel de la machine, et qui permet aux applications réseaux d'une même machine de communiquer entre elles mêmes si l'on ne dispose pas de carte réseau ;
- les interfaces des cartes réseaux (que ce soient des cartes Ethernet, TokenRing ou autres) ;
- les interfaces ppp, plip ou slip, qui sont des interfaces permettant d'utiliser les connexions sérielles, parallèles ou téléphoniques comme des réseaux.

La configuration d'une interface comprend l'initialisation des drivers nécessaires à son fonctionnement et l'affectation d'une adresse IP à cette interface. La syntaxe générale que vous devrez utiliser est la suivante :

```
ifconfig interface adresse netmask masque up
```

où `interface` est le nom de l'interface réseau que vous voulez configurer, `adresse` est l'adresse IP que cette interface gérera, et `masque` est le masque de sous-réseau que vous utilisez. Les interfaces que vous aurez à configurer seront certainement des interfaces Ethernet, auquel cas vous devrez utiliser les noms `eth0`, `eth1`, etc... dans la commande **ifconfig**. Si vous désirez configurer l'interface loopback, vous devrez utiliser le nom d'interface `lo`.

Le paramètre `up` donnée à **ifconfig** lui indique que l'interface doit être activée. Ceci signifie que dès que la commande **ifconfig** s'achèvera, votre interface réseau sera active et fonctionnelle. On ne peut faire plus simple... Bien entendu, il existe le paramètre inverse : `down`. Ce paramètre s'utilise tout simplement dans la commande **ifconfig** avec la syntaxe suivante :

```
ifconfig interface down
```

où `interface` est toujours le nom de l'interface.

Un exemple de configuration très classique est le suivant :

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

Par défaut, le noyau utilisera le nombre 255 pour les adresses de broadcast dans les composantes de l'adresse IP qui ne fait pas partie de l'adresse de sous-réseau. Si vous désirez utiliser une autre

adresse (en général, l'alternative est de prendre l'adresse du sous-réseau), vous devrez utiliser l'option `broadcast adresse` dans la commande **ifconfig**. Cependant, le comportement par défaut convient à la plupart des réseaux. La commande de configuration donnée en exemple ci-dessus sera alors :

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.0 up
```

La commande **ifconfig** est en général appelée dans les scripts d'initialisation du système, qui ont été générés par l'utilitaire de configuration réseau de votre distribution. Si vous effectuez un **grep** sur « ifconfig » dans le répertoire `/etc/rc.d/` (ou `/sbin/init.d/`, selon votre distribution), vous trouverez la commande de démarrage du réseau. Il se peut qu'une variable d'environnement soit utilisée à la place de l'adresse IP que vous avez choisie. Quoi qu'il en soit, vous devez sans aucun doute avoir les lignes suivantes, qui permettent l'initialisation de l'interface loopback :

```
ifconfig lo 127.0.0.1 netmask 255.0.0.0 up
```

7.2.2. Définition des règles de routage

La deuxième étape dans la configuration du réseau est la définition des règles de routage. Il est possible de définir plusieurs règles de routage actives simultanément. L'ensemble de ces règles constitue ce qu'on appelle la *table de routage*. La règle utilisée est sélectionnée par le noyau en fonction de l'adresse destination du paquet à router. Chaque règle indique donc un critère de sélection sur les adresses, et l'interface vers laquelle doivent être transférés les paquets dont l'adresse destination vérifie cette règle.

La commande utilisée pour définir une route est, chose surprenante, la commande système **route**. Sa syntaxe est donnée ci-dessous :

```
route opération [-net | -host] adresse netmask masque interface
```

où `opération` est l'opération à effectuer sur la table de routage. L'opération la plus courante est simplement l'ajout d'une règle de routage, auquel cas `add` doit être utilisé. L'option suivante permet d'indiquer si le critère de sélection des paquets se fait sur l'adresse du réseau destination ou plus restrictivement sur l'adresse de la machine destination. En général, il est courant d'utiliser la sélection de toutes les adresses d'un même réseau et de les router vers une même interface. Dans tous les cas, `adresse` est l'adresse IP de la destination, que celle-ci soit un réseau ou une machine. Si la destination est un réseau, il faut indiquer le masque de sous-réseau `masque` à l'aide de l'option `netmask`. Enfin, `interface` est l'interface réseau vers laquelle doivent être envoyés les paquets qui vérifient les critères de sélection de cette règle.

Par exemple, la règle de routage à utiliser pour l'interface loopback est la suivante :

```
route add -net 127.0.0.0 netmask 255.0.0.0 lo
```

Cette règle signifie que tous les paquets dont l'adresse de destination appartient au sous-réseau de classe A 127.0.0.0 doivent être transférés vers l'interface loopback. Ceci implique en particulier que les paquets à destination de la machine d'adresse IP 127.0.0.1 (c'est à dire la machine locale) seront envoyés vers l'interface loopback (ils reviendront donc sur la machine locale).

Une autre règle de routage classique est la suivante :

```
route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

Elle permet d'envoyer tous les paquets à destination du réseau de classe C 192.168.0.0 vers la première interface Ethernet. C'est typiquement ce genre de règle qu'il faut utiliser pour faire fonctionner un réseau local.

La commande **route** permet également de définir les passerelles à utiliser pour le routage des paquets qui sont destinés à une machine que la machine locale ne peut accéder directement. Les règles de routage faisant intervenir une passerelle sont semblables aux règles précédentes, à ceci près que l'adresse IP de la passerelle à utiliser doit être fournie. Pour cela, on utilise l'option `gw` (abréviation de l'anglais « Gateway »). La syntaxe utilisée est donc la suivante :

```
route add [-net | -host] adresse netmask masque gw passerelle interface
```

où `passerelle` est l'adresse IP de la passerelle à utiliser pour router les paquets qui vérifient les critères de cette règle. Les autres paramètres sont les mêmes que pour les règles de routage classique.

Par exemple, supposons qu'une machine soit connectée à un réseau d'adresse 192.168.0.0, et que sur ce réseau se trouve une passerelle d'adresse 192.168.0.1 permettant d'atteindre un autre réseau, dont l'adresse est 192.168.1.0. Une machine du réseau 192.168.0.0 aura typiquement les règles de routage suivantes :

```
route add -net 192.168.0.0 netmask 255.255.255.0 eth0
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.1 eth0
```

La première règle permet, comme on l'a déjà vu, de communiquer avec toutes les machines du réseau local. La deuxième règle permet d'envoyer à la passerelle 192.168.0.1 tous les paquets à destination du réseau 192.168.1.0.

Inversement, si la passerelle utilise l'adresse 192.168.1.15 sur le réseau 192.168.1.0, les machines de ce réseau qui voudront accéder au réseau 192.168.0.0 devront spécifier la règle de routage suivante :

```
route add -net 192.168.0.0 netmask 255.255.255.0 gw 192.168.1.15 eth0
```

Bien entendu, il est nécessaire que toutes les machines des deux réseaux utilisent ces règles de routage pour que la communication entre les deux réseaux se fasse dans les deux sens.

Le problème de ces règles de routage est qu'elles spécifient l'adresse du réseau destination. Il est évidement hors de question d'utiliser une règle de routage différente pour toutes les adresses de réseaux possibles. Il est donc possible de définir ce qu'on appelle une passerelle par défaut, qui n'est rien

d'autre que la passerelle vers laquelle doivent être envoyés tous les paquets qui n'ont pas vérifié les critères des autres règle de routage. La syntaxe à utiliser pour définir la passerelle par défaut est plus simple, puisqu'il n'est plus nécessaire de préciser les critères de sélection :

```
route add default gw passerelle interface
```

où la signification des paramètres `passerelle` et `interface` est inchangée.

Ainsi, pour reprendre l'exemple précédent, supposons que la machine 192.168.0.47 dispose d'une connexion à Internet. Pour que toutes les machines du réseau local puisse profiter de cette connexion, il suffit de demander à ce que tous les paquets qui ne vérifient aucune autre règle de routage soient envoyés à la passerelle 192.168.0.47. Ceci se fait avec la règle de routage suivante :

```
route add default gw 192.168.0.47 eth0
```

7.2.3. Définition du nom de la machine

La configuration du nom d'un ordinateur n'est pas à proprement parler une opération indispensable, mais elle permet de le nommer de manière plus conviviale qu'en utilisant son adresse IP. La commande de base permettant de manipuler le nom de la machine locale est la commande **hostname**. Appelée telle quelle, elle renvoie le nom actuel de la machine :

```
hostname
```

Cette commande permet également de modifier ce nom, simplement avec la syntaxe suivante :

```
hostname nom
```

où `nom` est le nom à utiliser. Il est d'usage de n'utiliser que le nom de la machine, sans son domaine. Le nom de domaine sera déterminé automatiquement par le système à partir des informations issues de la configuration de la résolution des noms de domaines. Nous verrons ceci dans le paragraphe suivant.

Cette commande est donc très simple à utiliser, et elle est en général appelée dans les scripts de démarrage du système. La plupart des distributions utilisent le fichier `/etc/HOSTNAME` pour stocker le nom de la machine. Vous êtes bien entendu libre de choisir le nom que vous voulez pour votre ordinateur, mais vous devez vous assurer que ce nom est unique dans votre domaine !

7.2.4. Résolution des noms de domaines

La commande **hostname** ne permet de fixer que le nom de la machine locale. Pour les autres machines du réseau, il faut mettre en place les mécanismes de résolution de noms de domaines. Comme il l'a déjà été indiqué au début de ce chapitre, il existe deux solutions pour trouver l'adresse IP d'une machine à partir de son nom : la consultation d'une liste de noms stockée en local, soit l'interrogation d'un serveur de nom de domaines.

Le fichier `/etc/host.conf` permet de définir le comportement du système lors de la résolution d'un nom. Sa structure est très simple, puisqu'il y a une option de recherche par ligne. Dans la plupart des cas, les lignes suivantes sont suffisantes :

```
order hosts,bind
multi on
```

Elles permettent d'indiquer que la recherche des noms pour leur résolution doit se faire d'abord localement, puis par appel aux DNS si la recherche précédente a échoué. C'est en général le comportement désiré. La deuxième ligne permet de faire en sorte que toutes les adresses correspondant à une machine soient renvoyées. Si l'on avait utilisé l'option `multi off`, seule la première adresse IP trouvée aurait été renvoyée.

La liste de noms locale est stockée dans le fichier `/etc/hosts` (ceci explique le nom `hosts` utilisé pour l'option `order` dans le fichier `/etc/host.conf`). Votre ordinateur connaîtra directement l'adresse IP de toutes les machines référencées dans ce fichier. Il est bon de placer ici une entrée pour les ordinateurs les plus couramment utilisés sur le réseau. Chaque entrée commence par une adresse IP, et est suivie de la liste des noms de la machine possédant cette adresse, séparés par des espaces. Pour ceux qui ne disposent pas de réseau local, ce fichier doit être relativement simple : seule la ligne affectant l'adresse 127.0.0.1 à la machine locale (appelée « `localhost` ») doit s'y trouver.

```
127.0.0.1    localhost
```

De la même manière, le fichier `/etc/networks` contient les adresses des réseaux. Ce fichier est utilisé par la commande route pour donner un nom aux différents réseaux. Chaque entrée est constituée du nom du réseau, suivi de son adresse IP. Encore une fois, ce fichier se réduit à sa plus simple expression pour ceux qui n'ont pas de réseau local, puisqu'il ne contiendra tout au plus qu'une entrée pour le réseau « loopback », sur lequel se trouve l'adresse de retour 127.0.0.1. Cette entrée aura donc la forme suivante :

```
loopback    127.0.0.0
```

La configuration des serveurs de noms est en revanche une opération nécessaire si l'on désire accéder à des machines dont on ne connaît que le nom. Le fichier de configuration utilisé est cette fois le fichier `/etc/resolv.conf`. Sa structure est encore une fois très simple, avec une option par ligne, chaque option étant introduite par un mot-clé.

Le mot-clé `domain` permet d'indiquer le nom du domaine dont fait partie votre machine. Par exemple, si votre nom de domaine est « andromede.galaxie », vous devrez utiliser la ligne suivante :

```
domain andromede.galaxie
```

Le mot-clé `search` permet quant à lui de spécifier une liste de noms de domaines à ajouter par défaut aux noms de machines non complètement qualifiés. Les éléments de cette liste doivent être séparés par des espaces. La recherche d'une machine dont le nom ne comprend pas la partie de nom de

domaine s'effectue en ajoutant au nom de la machine les noms des domaines indiqués ici, jusqu'à ce que la résolution du nom en adresse IP soit effectuée. Cette recherche commence bien entendu par le nom de domaine local, s'il a été défini. Il est donc recommandé d'indiquer votre nom de domaine dans cette liste de noms de domaines. Par exemple, si votre machine fait partie du domaine « andromede.galaxie », vous devrez utiliser la ligne suivante :

```
search andromede.galaxie
```

Ainsi, si vous recherchez l'adresse de la machine « krypton », la requête au DNS se fera avec le nom complètement qualifié « krypton.andromede.galaxie ». Vous êtes bien entendu libre d'ajouter d'autres noms de domaines pour le cas où la résolution de nom échouerait sur ce domaine.

Enfin, l'option `nameserver` est essentielle, puisqu'elle permet de donner les adresses IP des serveurs de DNS auxquels doivent être adressées les requêtes de résolution de noms. Par exemple, si vous disposez de deux serveurs DNS, un primaire, d'adresse 192.168.0.10, et un secondaire, d'adresse 192.168.0.15, vous utiliserez la ligne suivante :

```
nameserver 192.168.0.10 192.168.0.15
```

Cette ligne est évidemment obligatoire, faute de quoi la résolution des noms de machines en adresse IP échouera pour toute machine qui ne se trouve pas dans votre fichier `/etc/hosts`.

7.2.5. Définition des protocoles de haut niveau

Comme nous l'avons vu plus haut, le protocole IP fourni les mécanismes de base pour la transmission des paquets. Plusieurs protocoles de plus haut niveau ont été définis pour fournir des services à valeur ajoutée, qui satisfont donc plus aux besoins des applications. Tous ces protocoles sont encapsulés dans le protocole IP, ce qui signifie que leurs informations sont transmises en tant que données dans les paquets IP.

En réalité, les paquets du protocole IP contiennent un champ permettant de signaler le type de protocole de haut niveau dont ils transportent les informations. À chaque protocole est donc attribué un identificateur numérique qui lui est propre, et qui permet aux services réseaux d'interpréter les données des paquets.

Tout comme les adresses IP, ces numéros identifiant les protocoles ne sont pas très intéressant pour les humains, qui leur préfère évidemment le nom du protocole. Certains programme réseaux utilisent donc ces noms pour identifier les protocoles. Pour leur permettre de faire l'association entre ces noms et les identificateurs numériques, le fichier `/etc/protocols` est utilisé.

Le format de ce fichier est très simple. Il contient une ligne pour chaque protocole, constituée du nom du protocole, de la valeur de son identificateur, et des autres noms que ce protocole peut avoir (c'est à dire ses alias). Parmi ces protocoles, les plus importants sont les suivants :

Nom	Numéro	Alias
ip	0	IP
icmp	1	ICMP
tcp	6	TCP
udp	17	UDP

Comme on le voit, le protocole IP dispose lui-même d'un identificateur de protocole (qui vaut 0 en l'occurrence). Cet identificateur est un identificateur de « pseudo protocole », parce qu'IP est en fait le protocole de base. ICMP est identifié par le numéro 1, TCP par le numéro 6 et UDP par le numéro 27. Il existe beaucoup d'autres protocoles, qui ne seront pas décrits ici. Bien entendu, le fichier `/etc/protocols` fourni avec votre distribution doit déjà contenir la définition de la plupart des protocoles.

De la même manière, la plupart des ports TCP et UDP sont affectés à des services bien définis, et certains programmes réseau peut chercher à faire la correspondance entre les noms de ces services et les numéros de ports. Cette correspondance est stockée dans le fichier de configuration `/etc/services`.

Les informations concernant les services sont données à raison d'une ligne par service. Chaque ligne suit la syntaxe suivante :

```
nom    port/protocole    alias
```

où `nom` est le nom du service décrit par cette ligne, `port` est le numéro du port utilisé par ce service, `protocole` est le nom du protocole utilisé par ce service (ce peut être « `tcp` » ou « `udp` »), et alias la liste des autres noms sous lesquels ce service est également connu.

Vous trouverez les principaux services dans l'extrait donné ci-dessous :

Service	Port/Protocole
ftp	21/tcp
telnet	23/tcp
smtp	25/tcp
pop3	109/tcp
irc	194/tcp
irc	194/udp

Comme vous pouvez le constater, ces services n'ont pas d'alias. Ces informations sont données uniquement à titre d'exemple. Il va de soi que le fichier `/etc/services` fourni avec votre distribution contient la définition d'un grand nombre de services, et vous n'aurez en général pas à y toucher.

7.2.6. Le démon inetd

La plupart des services réseaux sont gérés par des programmes qui s'exécutent en permanence et qui

attendent des connexions sur un port TCP ou UDP. Ces programmes consomment relativement peu de ressources car ils passent la plupart de leur temps à attendre ces connexions. Ils ne se réveillent que lorsqu'un client se connecte effectivement et leur envoi une requête.

Cependant, ils peuvent être relativement nombreux, et si tous les services sont lancés simultanément, ils peuvent finir par consommer une part non négligeable des ressources. C'est pour cette raison que le démon « inetd » (de l'anglais « InterNET Daemon ») a été créé. Ce démon est à l'écoute les demandes de connexion des clients pour les autres services réseaux, et ne lancent ceux-ci que lorsqu'un client se connecte sur leur port. Une fois lancés, ceux-ci reprennent la main et communiquent directement avec leurs clients. Ainsi, inetd écoute les ports pour tout le monde, et est la plupart du temps le seul à fonctionner. Les ressources système sont économisées de cette manière, et les services réseaux sont démarrés et arrêtés à la demande.

Le démon inetd utilise le fichier de configuration `/etc/inetd.conf` pour déterminer les ports sur lesquels il doit attendre des connexions de la part des clients, et pour trouver le service réseau qu'il doit lancer lorsqu'une telle connexion arrive. Encore une fois, le fichier est structuré en lignes. Chaque ligne décrit un des services que le démon inetd prend en charge. Les informations données sur ces lignes sont les suivantes :

- le nom du service (tel qu'il est défini dans la première colonne du fichier `/etc/services`) dont inetd doit surveiller les requêtes ;
- le type de canal de communication réseau utilisé, en général « `stream` » (pour les communications en mode connecté, donc en général celles qui utilisent le protocole TCP) ou « `dgram` » (pour les communications basées sur les datagrammes, donc typiquement les communications utilisant le protocole UDP) ;
- l'un des mots-clés `wait` ou `nowait`, qui permettent d'indiquer si inetd doit attendre la fin de l'exécution du démon gérant ce service ou s'il peut attendre de nouvelles requêtes de la part des clients ;
- le nom de l'utilisateur au nom duquel le démon gérant ce service doit fonctionner (en général, c'est root) ;
- le chemin sur le fichier exécutable de ce démon ;
- les éventuels paramètres en ligne de commande pour ce démon, en commençant par l'argument 0, qui doit toujours être le nom du fichier exécutable du programme lui-même.

Par exemple, la ligne suivante permet de lancer le démon telnetd sur toute requête via le protocole TCP pour le service telnet :

```
telnet  stream  tcp     nowait  root    /usr/sbin/in.telnetd  in.telnetd
```

Il est supposé ici que le démon en charge de ce service peut être lancé avec le programme `/usr/sbin/in.telnetd`.

Le démon inetd est capable de fournir lui-même un certain nombre de services de base, et il n'est pas nécessaire de fournir un démon pour ces services. Dans ce cas, il faut utiliser le mot-clé `internal` à

la place du nom du fichier exécutable du démon de ce service. Les paramètres doivent également être remplacés par le mot-clé `internal`.

En fait, il est fort peu probable que votre fichier de configuration `/etc/inetd.conf` définisse les services comme indiqué dans cette section. En effet, un programme intermédiaire en charge d'assurer des contrôles de sécurité est souvent intercalé entre inetd et les démons gérant les services. Nous verrons ce que fait exactement ce programme dans la section suivante.

7.2.7. Configuration de la sécurité

La plupart des démons ont besoin d'avoir les privilèges du compte root pour s'exécuter. Ceci peut générer des problèmes de sécurité si d'aventure une personne mal intentionnée découvrait le moyen de les détourner de leur fonction initiale et de leur faire accomplir des tâches au nom de l'administrateur. Bien entendu, les démons sont des programmes très bien écrits, et qui vérifient en général toutes les requêtes demandées par les clients avant de les satisfaire. Cependant, tout programme peut comporter des failles et laisser ainsi un accès indésiré au sein du système.

Le principe de base de la sécurité est de ne faire confiance qu'au minimum possible de gens. Ce n'est pas un comportement paranoïaque, mais simplement du bon sens : puisque l'on ne peut garantir la fiabilité d'un système à 100%, il faut en restreindre les accès aux personnes habilitées. D'une manière générale, la plupart des surprises désagréables dans le monde environnant provient du mauvais jugement des personnes. Il est est de même dans le domaine de la sécurité informatique, il ne faut donner un accès qu'aux personnes que l'on connaît réellement bien, et ne fournir des services réseaux qu'aux personnes et machines qualifiées de « sûres ». Il va de soi que lorsqu'il y a un loup dans la bergerie, tout peut aller très mal très vite.

Tout ceci pour dire que les services réseaux d'un système Linux sont potentiellement nombreux, et parfois inutiles pour l'emploi que l'on veut en faire. Par conséquent, il vaut mieux tout simplement ne pas les proposer au monde extérieur, simplement en ne lançant pas les démons correspondants ou en commentant les lignes des services inutiles dans le fichier `/etc/inetd.conf`. Mais ceci n'est pas suffisant, il faut également restreindre le nombre de personnes et des machines auxquels les services nécessaires sont proposés.

Le service le plus sensible est bien entendu le login. Il est possible de fournir le service de login aux connexions extérieures, et il est évident que ceci constitue déjà un point d'entrée pour qui peut trouver un mot de passe valide. De toute évidence, si quelqu'un trouve le mot de passe du compte root, il a tout gagné. Ceci dit, il est possible de limiter sérieusement les dégâts en interdisant à l'utilisateur root de se connecter purement et simplement. Dans la majorité des cas cependant, il est concevable que root puisse se connecter sur la console locale (c'est à dire, avec votre clavier et votre écran…). Il est donc possible de choisir les terminaux à partir desquels l'utilisateur root peut se connecter. Ces informations sont stockées dans le fichier de configuration `/etc/securetty`.

Le format de ce fichier est très simple, puisqu'il est constitué de la liste des terminaux à partir desquels l'utilisateur root peut se connecter, à raison d'un terminal par ligne. Un fichier de configuration `/etc/securetty` typique contient donc la liste des terminaux virtuels de la machine :

```
# Exemple de fichier /etc/securetty
```

```
tty1
tty2
tty3
tty4
tty5
tty6
```

Il est donc fortement déconseillé de placer dans ce fichier les autres terminaux (en particulier, les terminaux série `ttyS0` et suivants).

Un autre service sensible est le service de téléchargement de fichiers. Il est recommandé d'interdire aux utilisateurs privilégiés les transferts de fichiers par FTP. En effet, si une personne parvient à accéder au système de fichiers, il peut supprimer tous les mécanismes de sécurité. De la même manière que l'on a interdit à l'utilisateur root de se connecter sur les terminaux distants, il faut lui interdire (ainsi qu'aux autres comptes sensibles) les connexions par FTP.

Cette opération peut être réalisée en ajoutant le nom de chaque utilisateur sensible dans le fichier de configuration `/etc/ftpusers`. Ce fichier a la même structure que le fichier securetty, puisqu'il faut donner un nom d'utilisateur par ligne :

```
# Exemple de fichier /etc/ftpusers
root
uucp
mail
```

Bien entendu, la liste des utilisateurs privilégiés de votre système dépend de la distribution que vous avez installé. Le fichier `/etc/ftpusers` fourni avec cette distribution est donc, en général, approprié.

Comme on le voit, il est possible de restreindre l'accès à certains utilisateurs pour certains services. Mais il est également possible de restreindre les accès provenant de certaines machines, considérées comme peu fiables. En particulier, on peut considérer comme dangereuse tout machine étrangère au réseau local. Le programme tcpd a été inventé pour effectuer ces contrôles. Il s'intercale entre le démon inetd et les démons que celui-ci est supposé lancer lorsqu'un client le demande. Ainsi, il lui est possible d'effectuer des vérifications de sécurité de base, principalement basées sur les adresses IP des clients.

Comme vous pouvez sans doute le constater dans le fichier `/etc/inetd.conf` de votre système, tcpd est lancé par inetd pour quasiment tous les services, et reçoit en paramètres de la ligne de commande le nom du démon à lancer si la machine cliente en a l'autorisation, ainsi que les paramètres à communiquer à ce démon. L'accès à quasiment tous les services est donc contrôlé par tcpd.

tcpd utilise les deux fichiers de configuration `/etc/hosts.allow` et `/etc/hosts.deny` pour déterminer si une machine a le droit d'accéder à un service donné. Le premier de ces fichiers indique quelles sont les machines autorisées à utiliser certains services locaux. Le deuxième fichier indique au contraire les machines qui ne sont pas considérées comme sûres et qui doivent se voir refuser toute demande de connexion sur les services locaux.

Le format de ces deux fichiers est identique. Il est constitué de lignes permettant de définir les règles d'accès pour certains démons. Chaque ligne utilise la syntaxe suivante :

```
démons : machines [ : commande]
```

où `démons` est une liste de noms de démons, séparés par des virgules, et `machines` est une liste de noms de machines ou d'adresses IP, également séparés par des virgules. `commande` est un paramètre optionnel permettant de faire exécuter une action à tcpd lorsque la règle indiquée par cette ligne est prise en compte.

Une règle définie par une de ces lignes est utilisée dès qu'une des machines indiquées dans la liste des machines tente une connexion à l'un des services de la liste des services. Si la règle se trouve dans le fichier `/etc/hosts.allow`, la connexion est autorisée et le service est lancé par tcpd. Si elle se trouve dans le fichier `/etc/hosts.deny`, la connexion est systématiquement refusée. Si aucune règle n'est utilisée, la connexion est acceptée par défaut.

Les listes de machines peuvent contenir des noms de machines partiels, et utiliser des caractères génériques. Il est également possible d'interdire la connexion à toutes les machines d'un domaine en ne donnant que le nom de domaine (précédé d'un point). Enfin, des mots-clés spéciaux permettent de représenter des ensembles de machines. Par exemple, le mot-clé `ALL` représente toutes les machines et `LOCAL` représente toutes les machines du réseau local. Le mot-clé `ALL` permet également de représenter l'ensemble des démons dans la liste des démons.

Par exemple, le fichier `/etc/hosts.deny` devrait contenir la ligne suivante :

```
# Exemple de fichier /etc/hosts.deny :
ALL : ALL
```

Ceci permet de garantir que par défaut, aucune demande de connexion n'est acceptée, ce qui est un comportement sain. Les machines ayant le droit de se connecter doivent être spécifiées au cas par cas dans le fichier `/etc/hosts.allow`, comme dans l'exemple suivant :

```
# Exemple de fichier /etc/hosts.allow :
telnetd, ftpd : LOCAL
```

Ceci permet d'autoriser les connexions telnet et ftp pour toutes les machines du réseau local.

Vous trouverez de plus amples renseignements sur le fonctionnement de tcpd dans la page de manuel `hosts_access(5)`.

> **Note:** Comme vous pouvez le constater si vous comparez les lignes du fichier de configuration `/etc/inetd.conf` utilisant tcpd et celles qui ne l'utilisent pas, la liste des paramètres passés à tcpd par inetd est différente de celle utilisée pour les démons lancés directement. En effet, elle ne comprend pas le nom de tcpd lui-même, alors que pour les démons, elle contient le nom du démon en premier paramètre. Cette différence provient du fait que le premier argument passé en ligne de commande est le nom du programme lui-même, sauf pour tcpd. En effet, tcpd suppose que ce paramètre contient le nom du démon dont il contrôle l'accès, et non son propre nom.

La sécurité basée sur tcpd suppose que les adresses IP source des paquets reçus sont effectivement les adresses IP des machines qui les ont envoyées. Malheureusement, cette hypothèse ne peut pas être vérifiée, car le protocole IP n'inclut aucun mécanisme d'authentification. Par conséquent, n'importe qui peut tenter de communiquer avec votre machine au nom d'une autre machine, qu'il aura au préalable mis hors d'état de se manifester (technique nommée « IP spoofing » en anglais). tcpd tente par défaut de contrôler ce genre de pratique en vérifiant que le nom indiqué par la machine qui se connecte correspond bien à l'adresse IP qu'elle utilise. Ainsi, pour passer cette protection, il faut d'abord infecter un DNS. Cependant, ce genre d'attaque (dite d'interposition) reste courant, surtout sur les réseaux dont les DNS sont insuffisamment protégés. Par conséquent, vous ne devez pas vous reposer uniquement sur tcpd et les techniques de firewalling si vous devez créer un site parfaitement sûr. En particulier, vous devriez vous intéresser aux réseaux virtuels et à l'encryptage des données. Le protocole Ipv6 intégrera des mécanismes d'authentification et sera donc nettement plus sûr. Bien entendu, ceci dépasse le cadre de ce document.

7.3. Configuration de la connexion à Internet

Les connexions à Internet font partie des connexions réseaux temporaires permettant de relier deux machines. La machine locale se connecte en effet au serveur du fournisseur d'accès à Internet via la ligne téléphonique. Bien entendu, ce serveur est considéré comme la passerelle par défaut pour tous les paquets à destination d'Internet. Ce type de connexion a de particulier que les adresses IP des deux machines sont, en général, déterminées dynamiquement, lors de l'établissement de la connexion. La configuration réseau des connexions à Internet se fait donc légèrement différemment de la configuration d'un réseau local normal. Heureusement, tous les détails de la configuration réseau sont pris en charge automatiquement.

7.3.1. Le protocole PPP

Les connexions à Internet utilisent le protocole PPP (abréviation de l'anglais « Point to Point Protocol »), qui permet à deux ordinateurs d'échanger des paquets TCP/IP au travers d'une ligne série (donc également au travers d'une ligne téléphonique en utilisant un modem). Ce protocole est géré par le noyau d'une part, et par le démon pppd d'autre part. Il permet d'effectuer la négociation initiale entre les deux machines, ce qui comprend notamment l'identification, l'authentification et l'attribution des adresses IP.

Le démon pppd ne gère pas le modem directement, car il est supposé être utilisable sur d'autres types de lignes que les lignes téléphoniques. La gestion du modem est donc relayée à un autre programme, qui se nomme **chat** (il tient son nom du fait qu'il permet de dialoguer directement avec le modem). Le programme **chat** ne connaît pas les différents types de modems qui existent sur le marché, il se contente d'exécuter des scripts qui décrivent les informations à envoyer au modem et les réponses attendues en retour. Ceci permet de reporter la configuration spécifique aux modems dans les scripts de connexions. L'écriture de ces scripts nécessite bien entendu de connaître les commandes que votre modem est capable de comprendre. Notez également qu'il est indispensable ici que votre modem soit un vrai modem (qu'il soit interne ou externe), et non pas un modem logiciel (ces modems sont des

modèles bas de gamme, qui nécessitent un support logiciel pour interpréter les informations analogiques reçues sur la ligne téléphonique). Renseignez-vous donc bien sur la nature du modem que vous achetez, si vous voulez ne pas avoir de problèmes sous Linux…

La séquence des opérations lors d'une connexion est donc l'initialisation du modem et l'établissement de l'appel téléphonique par le programme chat, puis le démarrage du démon pppd. Celui-ci effectue la connexion sur le serveur du fournisseur d'accès, et détermine l'adresse IP, les adresses des DNS, la passerelle et la route à utiliser pour cette connexion. Une fois ceci réalisé, toutes les fonctionnalités réseau peuvent être utilisées via Internet, et votre ordinateur fait alors partie du réseau mondial.

L'identification et l'authentification peuvent se faire de différentes manières selon le fournisseur d'accès à Internet utilisé. Un certain nombre de fournisseurs exige l'authentification lors de l'appel téléphonique, avec un mécanisme de login classique. Pour ces fournisseurs, l'authentification est faite par le programme **chat**, auquel il faut communiquer le nom d'utilisateur et le mot de passe. D'autres fournisseurs utilisent un protocole d'authentification spécifique. Pour ceux-ci, l'authentification est réalisée directement par le démon pppd, une fois que la ligne a été établie. Les deux principaux protocoles d'authentification sont PAP (abréviation de l'anglais « Password Authentification Protocol ») et CHAP (abréviation de « Chalenge Handshake Authentification Protocol »). PAP réalise l'authentification comme le mécanisme de login classique : le client envoie son nom et le mot de passe correspondant en clair au serveur. CHAP utilise en revanche la notion de défi. Le serveur envoie une requête contenant son nom, et le client doit s'identifier et authentifier son identité en lui renvoyant une réponse contenant son propre nom et une valeur calculée à partir du mot de passe correspondant. Les deux méthodes seront présentées ci-dessous. Si vous ne parvenez pas à vous connecter à Internet avec la première de ces méthodes, tentez votre chance avec PAP ou CHAP. Sachez que pour utiliser ces protocoles il est nécessaire de connaître, en plus de votre login et de votre mot de passe, le nom du serveur utilisé. Cette information doit vous être communiquée par votre fournisseur d'accès à Internet. Si vous avez le choix, utilisez de préférence le protocole CHAP, car il n'envoie jamais de mot de passe en clair sur la ligne téléphonique.

La configuration de l'accès à Internet pour un fournisseur d'accès requiert donc la configuration du réseau, la configuration de ppp, la configuration de chat et l'écriture des scripts de connexions. Les étapes seront détaillées ci-dessous. Nous supposerons dans la suite que vos paramètres de connexions sont les suivants :

Paramètre	Valeur
Fournisseur d'accès Internet	www.monfai.fr
Nom de domaine	monfai.fr
Adresse du DNS	192.205.43.1
Numéro de téléphone	08 36 76 30 18
Nom de login	jdupont
Mot de passe	gh25k ;f
Nom de serveur (pour PAP et CHAP)	serveurfai

Note: Les informations données dans le tableau ci-dessus sont fictives et ne servent qu'à donner un exemple. Vous devez bien entendu les remplacer par les valeurs vous concernant en chaque endroit où elles apparaissent dans la suite de ce document. Si par hasard une de ces informations correspondait à un numéro de téléphone, un nom ou une adresse IP valide, ce serait une pure coïncidence.

Certains fournisseurs d'accès refuseront de vous donner toutes ces informations, sous prétexte qu'ils vous fournissent un kit de connexion pour Windows. Ce n'est pas trop grave, car il est possible de leur extorquer ces informations malgré eux. Les seules informations réellement indispensables sont le numéro de téléphone, le nom de login, le mot de passe et le nom de domaine. Les adresses de DNS peuvent être déterminées automatiquement dans le cadre du protocole PPP, et le nom de serveur est arbitraire et ne sert qu'à identifier la connexion.

7.3.2. Création d'une connexion à Internet

La première étape dans la création d'une connexion à Internet est avant tout l'ajout des serveurs DNS du fournisseur d'accès à votre configuration réseau. Ceci n'est pas toujours nécessaire, car il est possible de configurer le démon pppd pour qu'il demande au fournisseur d'accès les adresses IP des DNS de celui-ci lors de l'établissement de la connexion. Cependant, il est plus facile d'utiliser le mécanisme de connexion à la demande si l'on indique les adresses des DNS du fournisseur d'accès dans la configuration réseau.

Pour l'exemple de connexion utilisé ici, vous devez simplement ajouter les lignes suivantes dans le fichier `/etc/resolv.conf` :

```
search monfai.fr
nameserver 192.205.43.1
```

Si vous ne connaissez pas les adresses de DNS de votre fournisseur d'accès, vous pourrez l'obtenir en regardant dans les traces du noyau lors de l'établissement d'une connexion. Nous verrons cela en détail plus loin.

La deuxième étape est d'écrire un script de numérotation pour le programme **chat**. Ce script pourra être placé dans le répertoire `/etc/ppp/peers/`, et pourra être nommé `monfai.chat` par exemple. Son contenu sera le suivant :

```
REPORT CONNECT
ABORT ERROR
ABORT BUSY
ABORT VOICE
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
"" ATZ
OK AT&F1
OK ATM0L0DT0836763018
CONNECT ""
```

Ce script contient le programme que chat doit exécuter pour appeler le fournisseur d'accès à Internet. Il indique toutes les erreurs possibles susceptibles de faire échouer l'opération, puis il envoie les commandes d'initialisation et de numérotation au modem. Vous devrez remplacer le numéro de téléphone utilisé dans la commande `DT<numéro>` par le numéro de téléphone de votre fournisseur d'accès à Internet.

> **Note:** Dans les scripts pour **chat**, il est possible d'utiliser indifféremment l'apostrophe simple, l'apostrophe double ou aucune apostrophe si la chaîne attendue de la part du modem ou à lui envoyer ne contient aucun espace.
>
> Si d'aventure ce script ne fonctionne pas, vous serez sans doute obligé de demander au programme chat d'afficher tous les messages envoyés et reçus par le modem. Ceci se fait normalement avec les options `-v` et `-s`, et la technique à utiliser sera décrite ci-dessous. Vous verrez alors les messages en provenance du modem, ce qui vous permettra de déterminer comment modifier ce script.

Si votre fournisseur d'accès à Internet utilise un mécanisme de login classique, vous devez faire l'identification directement dans le script chat. En général, serveur du fournisseur d'accès envoie la demande de login dès que la connexion a été établie. Pour cela, il utilise une chaîne de caractères telle que « `login :` », à laquelle le script chat doit répondre par votre nom d'utilisateur. Le serveur demande alors le mot de passe avec une chaîne telle que « `password :` », demande à suite de laquelle le script chat doit envoyer votre mot de passe. Ce n'est que lorsque ces deux informations auront été fournies que la connexion sera réellement établie. Vous pouvez compléter le script chat pour répondre à ces deux questions avec les deux lignes suivantes :

```
ogin : jdupont
ssword : gh25k;f
```

Vous devrez bien entendu remplacer le nom du login et le mot de passe par vos propres données dans ce script. Notez qu'il n'est pas nécessaire (ni recommandé) de demander la réception de la chaîne complète « `login :` », car une erreur de transmission sur la ligne peut encore arriver à ce stade et provoquer la perte des premiers caractères envoyés par l'ordinateur distant. De plus, il se peut que la première lettre soit en majuscule ou en minuscule, selon le fournisseur d'accès à Internet que vous utilisez. Il en va de même pour la demande de mot de passe (« `password :` »).

Si, en revanche, votre fournisseur d'accès utilise le mécanisme d'authentification PAP ou CHAP, vous ne devez pas ajouter ces lignes, car le serveur n'enverra pas la chaîne de caractères « `login :` ». Il tentera de communiquer directement avec votre démon pppd pour réaliser l'authentification. Bien entendu, les défis lancés par le serveur sont simplement constitués de la demande du login et de la demande du mot de passe correspondant à ce login. Le démon pppd utilisera alors les « secrets » stockés dans le fichier `/etc/ppp/pap-secrets` ou le fichier `/etc/ppp/chap-secrets` pour répondre à ces défis, selon que le protocole utilisé par le serveur du fournisseur d'accès est PAP ou CHAP. C'est donc dans ces fichiers que vous devrez enregistrer votre login et votre mot de passe. Le format de ces fichiers est très simple. Les deux fichiers utilisent la même syntaxe pour les secrets. Vous devez simplement y ajouter une ligne telle que celle-ci :

```
# Secrets for authentification using PAP/CHAP
# client server secret IP addresses
jdupont serveurfai gh25k;f
```

pour que l'identification et l'authentification se fasse correctement. Comme on le voit, le nom du serveur est indiqué dans ce fichier : il permet de déterminer quel login et quel mot de passe doivent être utilisés pour les protocoles PAP et CHAP.

> **Note:** En fait, le protocole PPP ne permet pas d'identifier des utilisateurs, mais des machines. Cependant, pour votre fournisseur, votre machine est identifiée par votre login, et il faut donc indiquer ce nom comme nom de machine cliente dans les fichiers `pap-secrets` et `chap-secrets`.
>
> Le nom de serveur n'est pas utilisé pour la connexion. Il ne sert qu'à déterminer quel secret doit être utilisé pour une connexion donnée. Comme la plupart des gens n'ont qu'un seul fournisseur d'accès à Internet, ce nom est purement et simplement facultatif. Dans ce cas, on peut parfaitement remplacer le nom du serveur par une étoile. Ce caractère générique indique simplement que le même secret doit être utilisé pour toutes les connexions.
>
> Pour les connexions à Internet, il est souvent impossible de connaître a priori l'adresse IP que le serveur donnera au client. On peut donc laisser vide la colonne contenant les adresses IP utilisables par les clients.

Lorsque vous aurez écrit le script chat et éventuellement complété les fichiers de secrets de PAP ou de CHAP, vous pourrez écrire le script de connexion à Internet. Ce script est très simple :

```
#!/bin/sh
# Script de connexion à Internet

# Effectue le ménage :
rm -f /var/spool/uucp/LCK* /var/lock/LCK* /var/run/ppp*.pid

# Établit la connexion :
/usr/sbin/pppd /dev/modem 115200 connect "/usr/sbin/chat -v \
    -f /etc/ppp/peers/monfai.chat" defaultroute usepeerdns \
    ipcp-accept-remote ipcp-accept-local
```

Prenez garde à écrire la ligne exécutant pppd et la ligne des options en une seule ligne, sans retour. Notez que le caractère '\' placé en fin de ligne permet d'indiquer que la commande se poursuit sur la ligne suivante. Ce script commence par faire le ménage sur les fichiers éventuellement laissés par les anciennes connexions, et appelle pppd en lui demandant d'utiliser la connexion établie par le programme **chat** avec la vitesse maximale de connexion indiquée. Lorsque la connexion est établie, le serveur du fournisseur d'accès est enregistré comme passerelle par défaut dans la table de routage du noyau. L'envoi et la réception des paquets IP en direction de l'Internet se fera donc en passant par cette passerelle. Les adresses de DNS du fournisseur d'accès sont également récupérées et ajoutées dans le fichier `/etc/resolv.conf`. Vous pourrez les déterminer simplement en consultant les fichiers de log de votre distribution dans le répertoire `/var/log/`. Les deux dernières options permettent d'autoriser le serveur du fournisseur d'accès à fixer les adresses IP locales et distantes.

Dans ce script de connexion, le programme chat est appelé avec le script de numérotation pour **chat** que l'on a déjà écrit. Si l'on désire voire exactement ce qui se passe, on peut ajouter l'option `-s` à la commande d'exécution de **chat** :

```
/usr/sbin/chat -v -s -f /etc/ppp/peers/monfai.chat
```

dans le script de connexion vu ci-dessus. Ceci vous permettra de déboguer votre script de numérotation.

Vous remarquerez que le programme pppd utilise le fichier spécial de périphérique `/dev/modem` pour la communication. Il va de soi que si ce fichier spécial n'existe pas, vous devrez le créer. La solution la plus simple est de faire un lien symbolique vers `/dev/ttS0` ou `/dev/ttS1`, selon que votre modem est connecté sur le premier ou le deuxième port série de votre ordinateur.

La commande de lancement donnée ci-dessus suppose que le script de connexion `/etc/ppp/peers/monfai.chat` réalise l'identification et l'authentification. Si vous utilisez l'un des protocoles PAP ou CHAP, il faut demander à pppd d'effectuer ces deux opérations lui-même. Pour cela, il faut ajouter des options dans la ligne de commande utilisée pour le lancer dans le script de connexion, afin de préciser le nom du serveur et le nom d'utilisateur pour l'identification. La ligne complète à utiliser pour PAP ou CHAP doit donc être celle-ci :

```
# Établit la connexion :
/usr/sbin/pppd /dev/modem 115200 connect "/usr/sbin/chat -v \
    -f /etc/ppp/peers/monfai.chat" defaultroute usepeerdns \
    ipcp-accept-remote ipcp-accept-local noauth \
    remotename serveurfai user jdupont
```

Encore une fois, cette commande doit être écrite sur une seule ligne, ou sur plusieurs lignes séparées par le caractère '\'. L'option `noauth` signale qu'il n'est pas nécessaire que le serveur du fournisseur d'accès s'authentifie en tant que tel (ce n'est pas nécessaire, car les fournisseurs d'accès ne jouent pas vraiment aux pirates du web), et les deux options suivantes permettent respectivement d'indiquer le nom de ce serveur ainsi que le nom d'utilisateur à utiliser pour le login. Lors de l'authentification, le démon pppd lira le fichier de secret du protocole choisi par le fournisseur d'accès pour trouver le mot de passe à utiliser. Notez encore une fois qu'il est facultatif de préciser le nom du serveur du fournisseur d'accès si vous avez utilisé le caractère générique `'*'` dans les fichiers de secrets pour PAP et CHAP.

Il est possible de faire en sorte que la connexion à Internet s'établisse automatiquement dès qu'un programme cherche à utiliser une adresse devant passer par la route par défaut. Toute demande à destination d'une machine dont l'adresse est inconnue localement provoque dans ce cas l'établissement de la connexion à Internet. Pour cela, il suffit simplement :

- d'ajouter l'option `demand` à la ligne de commande de pppd dans le script de connexion ;
- de lancer ce script en arrière plan dans un des scripts de démarrage du système.

Si vous désirez utiliser cette fonctionnalité, il est recommandé d'activer également la déconnexion automatique, faute de quoi vous risqueriez de rester connecté involontairement. Ceci se fait simplement en ajoutant l'option `idle n` dans la ligne de commande de pppd, où `n` est le nombre de secondes pendant lequel le lien PPP doit rester inactif avant que la connexion ne se coupe automatiquement. Un choix judicieux est par exemple 180 (ce qui correspond à trois minutes), étant donné que les trois premières minutes sont systématiquement facturées lors d'un appel téléphonique.

> **Note:** L'interface réseau est automatiquement créée lorsqu'on lance le démon pppd avec l'option `demand`. Par conséquent, le démon pppd définira automatiquement une adresse IP locale pour cette interface et une adresse IP distante pour la passerelle par défaut. Malheureusement, la plupart des fournisseurs attribuent les adresses IP dynamiquement, et en général, la connexion ne se fait pas toujours sur le même serveur (il peut y avoir plusieurs serveurs pour un même numéro de téléphone). C'est pour cela qu'il faut ajouter les options `ipcp-accept-remote` et `ipcp-accept-local` pour indiquer à pppd de modifier ces adresses lorsqu'il établira la connexion. Le problème, c'est que les connexions TCP/IP utilisant ces adresses seront systématiquement cassées à la suite de ce changement. C'est en particulier le cas pour le programme qui a initié l'établissement de la liaison PPP. C'est pour cette raison qu'il est recommandé de demander une adresse IP fixe à son fournisseur d'accès lorsque l'on veut utiliser la connexion à la demande. Hélas, ce service peut faire l'office d'une facturation supplémentaire.
>
> Même si l'on utilise l'option `usepeerdns` dans la ligne de commande de PPP, il est recommandé de rajouter les adresses des DNS dans le fichier `/etc/resolv.conf`. En effet, en l'absence de DNS, les noms de domaines ne seront pas résolus et aucune requête vers l'extérieur ne se fera. Par conséquent, la route par défaut ne sera pas utilisée, et pppd n'établira donc pas la connexion automatiquement. Notez que si vous définissez les DNS dans le fichier `/etc/resolv.conf`, vous pouvez vous passer de l'option `usepeerdns` dans la ligne de commande de pppd. Si vous désirez procéder de cette manière, vous devrez vous assurer que l'ordre spécifié pour la résolution des noms dans le fichier `/etc/host.conf` est bien le fichier `/etc/hosts` (option `hosts`) avant le DNS (option `bind`), faute de quoi votre ordinateur se connectera systématiquement à Internet dès que vous utiliserez un nom de machine local.
>
> La plupart des options passées en ligne de commande peuvent être spécifiées dans le fichier d'options `/etc/ppp/options` du démon pppd. Ceci peut permettre de simplifier les scripts de connexion.

Si l'on n'utilise pas les mécanismes de connexion / déconnexion automatiques, on devra se déconnecter manuellement. Pour cela, rien de plus simple : il suffit de détruire le démon pppd. Ceci peut être réalisé automatiquement avec le script suivant :

```
#!/bin/sh
# Script de terminaison de la connexion PPP
#

# Détermine la connexion à fermer (fournie sur la ligne de commande,
# ppp0 par défaut) :
if [ "$1" = "" ]; then
        DEVICE=ppp0
else
```

```
        DEVICE=$1
fi

# Teste si la connexion indiquée est active :
if [ -r /var/run/$DEVICE.pid ]; then

        # Détruit le processus ppp correspondant (son PID est stocké
        # dans /var/run/) :
        kill -INT ‘cat /var/run/$DEVICE.pid‘

        # Effectue le ménage :
        if [ ! "$?" = "0" ]; then
                rm -f /var/run/$DEVICE.pid
                echo "ERREUR : Un fichier de PID résiduel \
a dû être effacé manuellement"
                exit 1
        fi
        rm -f /var/spool/uucp/LCK* /var/lock/LCK*
        echo "La connexion PPP sur $DEVICE a été fermée correctement..."
        echo

        # Termine le script :
        exit 0
fi

# La connexion indiquée n’est pas active :
echo "ERREUR : Il n’y a pas de connexion PPP sur $DEVICE"
exit 1
```

7.3.3. Utilisation du mail

À présent, vous pouvez vous connecter à Internet simplement en exécutant le script de connexion. Pour vous déconnecter, il suffit d’appeler le script de déconnexion. Dès que vous êtes connecté à Internet, vous pouvez utiliser toutes les fonctionnalités réseau de Linux, qui sont sans limites. Parmi ces fonctionnalités, on trouve l’envoi de courriers électroniques. Cette opération peut être réalisée à l’aide du programme **mail**, qui se manipule à l’aide de commandes simples. Les commandes disponibles à chaque instant sont affichées dans le bas de l’écran de mail.

Le programme **mail** n’est qu’un client, qui ne fait rien d’autre que d’envoyer et recevoir des mails au serveur de mails local. Sur la plupart des distributions, le serveur de mails utilisé se nomme « sendmail ». Si vous désirez utiliser le courrier électronique, vous devrez donc configurer sendmail. Hélas, cette opération est technique et difficile. Il est recommandé d’utiliser l’outil de configuration de votre distribution, qui fera tout le sale boulot pour vous. Toutefois, si vous désirez configurer sendmail vous-même, sachez que la plupart des options sont stockées dans le fichier `/etc/sendmail.cf`. Vous devriez vous documenter avant de vous lancer dans cette opération. Le fichier `/etc/sendmail.cf` ne sera pas décrit ici, il peut faire à lui seul l’objet de plusieurs livres…

Classiquement, lorsque vous envoyez un mail, ce mail est envoyé à sendmail. Si vous êtes connecté à Internet, celui-ci le transmet immédiatement à votre fournisseur d'accès à Internet, qui le rediffuse. En revanche, si vous n'êtes pas connecté, sendmail stockera votre message dans le répertoire `/var/spool/mqueue/`. Les courriers ne seront envoyés que lorsque vous vous connecterez à Internet. sendmail vérifie périodiquement s'il peut envoyer les mails en attente, cependant, si vous ne restez pas connecté suffisamment longtemps, il peut ne pas s'en apercevoir. Pour éviter cela, vous pouvez utiliser la commande suivante :

```
sendmail -q
```

Cette commande demande à sendmail d'envoyer les courriers en attente immédiatement.

De la même manière, le programme **mail** ne récupère pas lui-même les mails depuis le serveur de mail de votre fournisseur d'accès à Internet. En général, le programme utilisé pour récupérer les mails se nomme « fetchmail ». Si vous voulez récupérer immédiatement vos mails, vous devrez donc appeler fetchmail manuellement :

```
fetchmail
```

Bien entendu, ces opérations sont ennuyeuses. Si vous voulez être tranquille, vous pouvez placer ces deux commandes à la fin du script de connexion à Internet. De la sorte, vous recevrez et enverrez votre courrier à chaque fois que vous vous connecterez…

7.3.4. Les autres outils de connexion

Vous en savez suffisamment à présent pour vous connecter à Internet avec Linux. Les opérations qui ont été décrites ici sont légèrement compliquées, et vous les trouvez sans doute un peu lourdes. Ceci est naturel, car elles le sont effectivement. En fait, les opérations décrites ici vous montrent comment vous connecter manuellement, mais elles ne constituent pas la manière de faire la plus facile. En effet, si vous installez XWindow, vous pourrez utiliser des programmes graphiques permettant de vous connecter à Internet, d'envoyer et recevoir des courriers électroniques (« mails », de naviguer (« surfer ») et de lire les groupes de discussions (« news »). Cependant, l'utilisation de ces outils ne sera pas décrite ici, car il en existe trop pour que l'on puisse tous les présenter. Heureusement, ces outils sont prévus pour être très simple d'emploi et leur configuration ne pose réellement aucun problème.

7.4. Firewalls et partages de connexion à Internet

Supposons qu'une machine située sur un réseau local a accès à Internet. Il peut être intéressant de faire en sorte que les autres machines du réseau puissent également y accéder, en utilisant cette machine comme passerelle. Ceci est parfaitement réalisable et ne pose aucun autre problème que la définition des règles de routage si toutes les machines ont une adresse IP attribuée par l'IANA. Cependant,

ceci est rarement le cas, car les réseaux locaux utilisent normalement les adresses réservées à cet usage, qui ne sont pas routables sur Internet. Dans ce cas, il est évident que les machines du réseau local ne pourront pas envoyer de paquets sur Internet, et qu'a fortiori elles ne recevront jamais de paquets provenant d'Internet. Heureusement, il existe une technique nommée *masquerading*, basée sur un mécanisme de translations d'adresses (« NAT » en anglais, abréviation de « Network Address Translation »), qui permet de modifier les paquets émis à destination d'Internet à la volée, afin de pouvoir partager une connexion Internet même avec des ordinateurs qui utilisent des adresses locales. Comme nous allons le voir, partager une connexion à Internet avec d'autre ordinateurs d'un réseau local est un jeu d'enfant sous Linux grâce à cette technique.

Il faut toutefois bien se rendre compte que le fait que fournir un accès à Internet à un réseau local pose des problèmes de sécurité majeurs. Pour des réseaux locaux familiaux, les risques de piratages sont bien entendu mineurs, mais lorsque la connexion à Internet est permanente ou lorsque les données circulant sur le réseau local sont sensibles, il faut tenir compte des risques d'intrusion. Lorsque l'on utilise des adresses IP dynamiques, il est relativement difficile d'accéder à des machines du réseau local, sauf si la passerelle expose des services internes au reste du réseau. En revanche, si les adresses utilisées sont fixes et valides sur Internet, le risque devient très important. La configuration de la passerelle doit donc se faire avec soin dans tous les cas, et l'installation d'un Firewall est plus que recommandée (un « firewall », ou « pare-feu » en français, est un dispositif qui consiste à protéger un réseau local du « feu de l'enfer » d'Internet, sur lequel tous les crackers sont supposés grouiller).

Les paragraphes suivants exposent les mécanismes de filtrage de paquets réseau de Linux, qui sont utilisées tant pour définir les règles de protection contre les intrusions indésirables sur un réseau par l'intermédiaire d'une passerelle que pour effectuer des traitements sur les paquets. Cependant, de tous ces traitements, nous ne verrons que la translation d'adresses, car c'est sans doute celui que la plupart des gens cherchent à utiliser.

7.4.1. Mécanismes de filtrage du noyau

Linux est capable de filtrer les paquets circulant sur le réseau et d'effectuer des translations d'adresses depuis la version 2.0, cependant, les techniques utilisées ont été profondément remaniées à chaque version. À partir de la version 2.4.0, une architecture extensible a été mise en place et semble répondre à tous les besoins de manière simple : *Netfilter*.

Netfilter est simplement une série d'entrée dans les couches réseau du noyau au niveau desquels des modules peuvent s'enregistrer afin d'effectuer des contrôles ou des traitements particuliers sur les paquets. Il existe un entrée en chaque point clé du trajet suivi par les paquets dans les couches réseau du noyau, ce qui permet d'intervenir de manière souple et à n'importe quel niveau. Un certain nombre de modules permettant d'effectuer les traitements les plus courants sont fournis directement dans le noyau, mais l'architecture Netfilter est suffisamment souple pour permettre le développement et l'ajout des modules complémentaires qui pourraient être développés par la suite.

Les fonctionnalités fournies par ces modules sont regroupées par domaine fonctionnel. Ainsi, les modules permettant de réaliser des Firewall se chargent spécifiquement de donner les moyens de filtrer les paquets selon des critères bien définis, et les modules permettant d'effectuer des translations d'adresses ne prennent en charge que la modification des adresses sources et destination des paquets

à la volée. Afin de bien les identifier, ces fonctionnalités sont regroupées dans ce que l'on appelle des *tables*. Une table n'est en fait rien d'autre qu'un ensemble cohérent de règles permettant de manipuler les paquets circulant dans les couches réseau du noyau. Les tables les plus couramment utilisées sont les tables « filter » et « nat », qui permettent respectivement de réaliser des Firewall et d'effectuer des translations d'adresses.

Chaque table utilise ses propres points d'entrées de Netfilter. Pour chacun de ces points d'entrée, des listes de règles de gestion des paquets peuvent être définies. Ces listes sont communément appelées des *chaînes*. Il y a donc toujours une chaîne de règles prédéfinie pour chaque point d'entrée. L'utilisateur peut bien entendu définir de nouvelles chaînes et les utiliser dans les chaînes prédéfinies.

Les règles permettent de spécifier les paquets qui les vérifient, selon des critères précis, comme par exemple leur adresse source ou le type de protocole qu'ils transportent. Les règles indiquent également les traitements qui seront appliqués à ces paquets. Par exemple, il est possible de détruire purement et simplement tous les paquets provenant d'une machine considérée comme non sûre si l'on veut faire un Firewall. On pourra aussi envoyer ces paquets dans une autre chaîne, qui peuvent indiquer d'autres règles et d'autres traitements.

Les critères de sélection des paquets des règles de filtrage se basent sur les informations de l'entête de ces paquets. Rappelez-vous que chaque paquet de données émis sur le réseau transporte avec lui des informations nécessaires au fonctionnement dudit réseau, entre autres les adresses source et destination du paquet. Ce sont sur ces informations que la sélection des paquets est effectuée dans Netfilter. Dans le cas d'un protocole encapsulé dans un autre protocole, comme par exemple un paquet TCP dans un paquet IP, les informations du protocole encapsulé peuvent également être utilisées, si l'on a chargé des modules complémentaires dans NetFilter. Par exemple, le numéro de port TCP peut faire partie des critères de sélection d'une règle.

Les traitements que l'on peut appliquer à un paquet sont également fournis par des modules de Netfilter. Les traitements les plus simples sont bien entendus fournis en standard avec le noyau. En général, on peut chercher à réaliser les opérations suivantes sur un paquet :

- l'enregistrer, pour analyse ultérieure ;
- le rediriger vers un port local, pour traitement par un programme dédié (par exemple, par un serveur proxy) ;
- l'accepter, l'abandonner ou le rejeter directement (le rejet se distingue de l'abandon par l'émission d'une notification « hôte inaccessible » à la machine ayant émis le paquet) ;
- le modifier, et en particulier, le marquer pour le reconnaître dans un traitement ultérieur ou effectuer une translation d'adresses ;
- l'envoyer dans une autre chaîne pour effectuer des vérifications complémentaires ;
- le faire sortir immédiatement de la chaîne courante.

De plus, les statistiques tenues par le noyau pour la règle ainsi vérifiée sont mises à jour. Ces statistiques comprennent le nombre de paquets qui ont vérifié cette règle ainsi que le nombre des octets que ces paquets contenaient.

Lorsqu'un paquet arrive à la fin d'une chaîne (soit parce qu'il n'a pas été rejeté ou détruit par une rèfle de Firewalling, soit parce qu'il ne correspond aux critères d'aucune règle, ou soit parce qu'il est arrivé à ce stade après avoir subi des modifications), le noyau revient dans la chaîne appelante et poursuit le traitement du paquet. Si la chaîne au bout de laquelle le paquet arrive est une chaîne prédéfinie, il n'y a plus de chaîne appelante, et le sort du paquet est déterminé par ce qu'on appelle la *politique*. La politique (« policy » en anglais) des chaînes prédéfinies indique donc ce que l'on doit faire avec les paquets qui arrivent en fin de chaîne. En général, on utilise une politique relativement stricte lorsqu'on réalise un Firewall, qui rejette tous les paquets par défaut et qui n'ont pas été accepté explicitement par une règle de la chaîne.

> **Note:** Certains paquets peuvent être découpés en plusieurs paquets plus petits lors de la traversée d'un réseau. Par exemple, un paquet TCP un peu trop gros et initialement encapsulé dans un seul paquet IP peut se voir répartir sur plusieurs paquets IP. Ceci peut poser quelques problèmes aux règles de filtrage, puisque dans ce cas les données spécifiques à TCP (par exemple les numéros de ports) ne sont disponibles que sur le premier paquet IP reçu, pas sur les suivants. C'est pour cette raison que le noyau peut effectuer une « défragmentation » des paquets lorsque cela est nécessaire, et que les paquets transmis par la passerelle ne sont pas toujours strictement identitiques aux paquets qu'elle reçoit.

7.4.2. Translations d'adresses et masquerading

Nous avons vu que grâce aux mécanismes de filtrage du noyau, il est possible de modifier les paquets à la volée. Il peut être intéressant de modifier un paquet pour diverses raisons, les trois plus intéressantes étant sans doute de le marquer à l'aide d'un champ spécial dans son en-tête afin de le reconnaître ultérieurement, de modifier sa priorité pour permettre un traitement privilégié ou au contraire en arrière plan de ce type de paquet (libérant ainsi les ressources réseau pour d'autres connexions), et de modifier ses adresses sources et destination, pour effectuer une translation d'adresse. Nous allons nous intéresser plus particulièrement aux différentes formes de translation d'adresses dans la suite de cette section.

Une *translation d'adresse* n'est en fait rien d'autre qu'un remplacement contrôlé des adresses sources ou des adresses destination de tous les paquets d'une connexion. En général, ceci permet de détourner les connexions réseau ou de simuler une provenance unique de plusieurs connexions. Par exemple, les connexions à Internet peuvent être détournées vers un proxy fonctionnant sur la passerelle de manière transparente, en modifiant les adresses destination des paquets émis par les clients. De même, il est possible de simuler un seul serveur en remplaçant à la volée les adresses source des paquets que la passerelle envoie sur Internet.

On distingue donc deux principaux types de translation d'adresses : les translations d'adresses sources et les translation d'adresse destination. Parmi les translations d'adresses source se trouve un cas particulier particulièrement intéressant : le « masquerading ». Cette technique permet de remplacer toutes les adresses sources des paquets provenant des machines d'un réseau local par l'adresse de l'interface de la passerelle connectée à Internet, tout en conservant une trace des connexions réseau afin d'acheminer vers leur véritable destinataire les paquets de réponse. Ainsi, une seule connexion à Internet peut être utilisée par plusieurs machines distinctes, même si elles ne disposent pas d'adresses

IP fournies par l'IANA.

Lorsqu'une machine du réseau local envoie un paquet à destination d'Internet, ce paquet est donc acheminé vers la passerelle, qui est la route par défaut. Celle-ci commence par modifier leur adresse source et mettre sa propre adresse à la place, puis les transfert vers la machine du fournisseur d'accès. Tous les paquets émis par les machines du réseau local semblent donc provenir de cette passerelle, et seront acheminés normalement à destination. En fait,la complication provient des paquets réponse, puisque les machines situées sur Internet croient que la machine avec laquelle elles communiquent est la passerelle. Les paquets réponse sont donc tous envoyés à la passerelle directement, et celle-ci doit retrouver la machine du réseau local à laquelle ce paquet est destiné. Cette opération est réalisée de différentes manières selon les protocoles utilisés, et elle suppose que la passerelle conserve en permanence une trace des connexions réseaux effectuées par les machines du réseau local.

Pour TCP, ce suivi de connexion est réalisé en modifiant également le port source des paquets provenant des machines locales. La passerelle utilise un port unique pour chaque connexion, qui va lui permettre de retrouver la machine à laquelle un paquet est destiné lorsqu'elle en reçoit un provenant d'Internet. Par exemple, si une machine locale fait une connexion à Internet, la passerelle alloue un nouveau numéro de port pour cette connexion et modifie tous les paquets sortants comme ayant été émis par la passerelle elle-même, sur ce port. Lorsque l'autre machine prenant part à la connexion, située sur Internet, envoie un paquet réponse, celui-ci sera à destination de la passerelle, avec comme port destination le port de la connexion. La passerelle peut donc retrouver, à partir de ce port, l'adresse de la machine destination réelle, ainsi que le port source que cette machine utilisait initialement. La passerelle modifie donc ces deux champs du paquet, et le renvoie sur le réseau local. Finalement, la machine destination reçoit le paquet sur son port, et ce paquet semble provenir directement d'Internet, comme si la connexion avait été directe. Notez bien que la passerelle ne modifie pas les adresses sources des paquets provenant d'Internet, elle ne fait que les réacheminer vers la bonne destination.

Ainsi, le masquerading est un mécanisme complètement transparent pour les machines du réseau local. En revanche, pour les machines de l'Internet, il ne semble y avoir qu'un seul interlocuteur : la passerelle. Celle-ci utilise des numéros de ports variés, mais ceci ne les regarde pas. Les machines du réseau local sont donc complètement « masquées » par la passerelle, d'où le nom de *masquerading*.

Tout ce travail effectué par la passerelle nécessite un traitement spécifique sur chaque paquet qu'elle achemine, et consomme bien entendu des ressources système. Cependant, les débits utilisés pour les connexions à Internet, même les plus rapides, sont très loin de mettre à genoux une passerelle sous Linux, même sur les plus petites machines (386 et 486). Vous voilà rassuré, et peut-être aurez-vous trouvé d'ici peu une utilité à l'un de ces dinosaures qui traînent encore dans votre garage…

7.4.3. Trajet des paquets

Il existe cinq points d'entrée au total pour les modules de Netfilter dans le code des couches réseau IP de Linux. À ces cinq points d'entrée correspondent cinq chaînes prédéfinies, qui peuvent être accédées par l'intermédiaire des diverses tables de Netfilter. Les paragraphes suivants décrivent ces cinq chaînes, ainsi que l'ordre dans lequel les paquets les traversent.

Les paquets réseau provenant de l'extérieur sont d'abord contrôlés par le noyau afin de détecter les erreurs les plus simples. Les paquets mal formés ou corrompus sont systématiquement éliminés à ce

niveau, avant même d'atteindre le code de Netfilter. Les autres paquets commencent alors leur trajet dans les couches de plus haut niveau. C'est à ce moment là qu'ils rencontrent la chaîne `PREROUTING`, qui est la première chaîne prise en compte. Cette chaîne précède le code de routage, qui est en charge de déterminer le trajet que le paquet devra suivre par la suite à partir de l'adresse destination des paquets. C'est donc dans cette chaîne que l'on pourra modifier l'adresse destination des paquets, si l'on veut faire une translation d'adresse destination. Cette chaîne est donc utilisable dans la table `nat`.

Les paquets qui traversent la chaîne `PREROUTING` sont ensuite orientés par le code de routage de Linux. S'ils sont à destination de la machine locale, ils rencontrent une autre chaîne prédéfinie : la chaîne `INPUT`. C'est à ce niveau que l'on peut empêcher un paquet d'entrer dans le système, aussi cette chaîne est-elle naturellement disponible dans la table `filter`.

Les autres paquets doivent être transférés vers une autre machine, souvent par une autre interface réseau. Ils traversent alors la chaîne `FORWARD`. Cette chaîne permet de contrôler les paquets qui transitent au travers d'une passerelle, et fait donc partie de la table `filter`.

Les paquets qui sont acceptés par la chaîne `FORWARD` retournent ensuite dans le code de routage du noyau, afin de déterminer l'interface réseau par laquelle ils doivent être réémis. Une fois ceci réalisé, ils sont prêts à être réémis, mais ils doivent auparavant traverser la chaîne `POSTROUTING`. C'est dans cette dernière chaîne que se font les translations d'adresse source, aussi est-elle accessible dans la table `nat`.

Enfin, les paquets émis par la machine locale à destination de l'extérieur doivent impérativement traverser la chaîne `OUTPUT`, dans laquelle on pourra filtrer les informations sortantes. Cette chaîne est donc accessible dans la table `filter`. Les paquets qui se révèlent avoir le droit de sortir traversent alors de code de routing, puis, tout comme les paquets qui ont été transférés, entrent dans la chaîne `POSTROUTING`. La chaîne `OUTPUT` est également accessible au travers de la table `nat`, afin de permettre la modification des adresses destination des paquets avant leur routage.

> **Note:** Pour des raisons de clarté, les chaînes prédéfinies et les points d'entrée de Netfilter ont été volontairement confondus dans ce paragraphe. En fait, il faut bien comprendre que les chaînes appartiennent aux tables, et qu'il peut exister plusieurs chaînes du même nom dans plusieurs tables différentes. L'ajout d'une règle dans une chaîne ne se fait donc que pour une table donnée, les autres tables ne contiendront donc pas cette règle. En revanche, les paquets qui traversent ces chaînes le font bien lorsqu'ils sont au niveau du point d'entrée correspondant dans le code des couches réseau de Linux.

7.4.4. Configuration du noyau et installation des outils

Les fonctionnalités de Netfilter sont toutes fournies par le noyau lui-même, aussi leur configuration doit-elle se faire au niveau du noyau. La manière de procéder pour configurer le noyau sera indiquée ultérieurement dans le paragraphe traitant de la compilation du noyau. Sachez cependant que les options à activer pour mettre en place les fonctionnalités de filtrage des paquets et de translation d'adresses sont les suivantes :

- « `Network packet filtering (replace ipchains)` », pour activer les fonctionnalités de filtrage des paquets en général ;
- « `Network packet filtering debugging` », pour obtenir les message de débogage du noyau. Ces messages peuvent être très utiles pour mettre au point les règles de filtrage
- « `Connection tracking (required for masq/NAT)` » (menu « IP: Netfilter Configuration »), pour permettre le suivi des connexions auxquelles appartiennent les paquets IP reçus par la passerelle. Cette option est nécessaire pour réaliser un partage de connexion à Internet ;
- « `FTP protocol support` », pour permettre la gestion des connexions FTP, qui exigent un traitement particulier pour les translations d'adresses ;
- « `IP tables support (required for filtering/masq/NAT)` », pour permettre la gestion des tables de Netfiler ;
- « `Packet filtering` », pour inclure le support de la table « filter ». Cette option est nécessaire pour réaliser un Firewall ;
- « `REJECT target support` », pour permettre le rejet des paquets (par défaut, seul l'abandon des paquets est fourni dans le code de filtrage) ;
- « `Full NAT` », pour inclure la table « nat » de Netfilter. Cette option est nécessaire pour réaliser un partage de connexion à Internet ;
- « `MASQUERADE target support` », pour permettre le masquerading afin de réaliser un partage de connexion à Internet.

Vous devrez ensuite compiler le noyau et les modules, et les installer comme indiqué dans la partie traitant de la compilation du noyau.

> **Note:** Le noyau de Linux dispose d'un paramètre global permettant de contrôler le routage des paquets IP d'une interface réseau à une autre. Par défaut, ce paramètre est à 0, ce qui implique que la transmission des paquets ne sera pas autorisée. Il faut donc activer ce paramètre à chaque démarrage afin de pouvoir utiliser votre passerelle Linux. Ceci peut être réalisé en écrivant la valeur `1` dans le fichier `/proc/sys/net/ipv4/ip_forward` du système de fichiers virtuel `/proc/` :
>
> ```
> echo "1" > /proc/sys/net/ipv4/ip_forward
> ```

La manipulation des chaînes et la manipulation des règles de Netfilter se fait grâce à l'outil **iptables**. Cette commande doit normalement avoir été installée par votre distribution, toutefois si ce n'est pas le cas, vous pouvez la compiler manuellement. Pour cela, il vous faudra récupérer l'archive des sources d'iptables. Cette archive peut être trouvée sur Internet (http ://netfilter.filewatcher.org/iptables-1.1.2.tar.bz2). La version courante est la 1.1.2, aussi l'archive se nomme-t-elle `iptables-1.1.2.tar.bz2`.

L'installation d'iptables ne pose pas de problème particuliers en soi. Vous devrez d'abord décompacter l'archive avec les deux commande suivantes :

```
bunzip2 iptables-1.1.2.tar.bz2
```

```
tar xf iptables-1.1.2.tar
```

puis modifier le fichier `Makefile` pour définir les répertoires d'installation dans les variables LIBDIR, BINDIR et MANDIR. En général, les répertoires utilisés seront respectivement les répertoires `/usr/lib/`, `/usr/bin/` et `/usr/man/`. Une fois ceci fait, il ne restera plus qu'à compiler le tout avec la commande **make** et à faire l'installation avec la commande **make install**.

7.4.5. Utilisation d'iptables

Il est possible, grâce à **iptables**, d'effectuer toutes les opérations d'administration désirées sur les tables de Netfilter. Vous pourrez donc créer des nouvelles chaînes, les détruire, définir leur politique par défaut, ainsi que manipuler les règles de ces chaînes (en ajouter, en supprimer ou en remplacer une). En fait, **iptables** dispose d'un grand nombre d'options, et seules les options les plus importantes seront présentées ici. Vous pouvez lire la page de manuel `iptables` si vous désirez plus de renseignements sur la manipulation des chaînes et des règles de filtrage du noyau.

7.4.5.1. Manipulation des chaînes

Toutes les options peuvent être utilisées avec toutes les les tables gérées par le noyau. La table sur laquelle une commande s'applique peut être précisée à l'aide de l'option `-t`. Si cette option n'est pas fournie, la table utilisée par défaut sera la table `filter`, qui est celle qui est normalement utilisée pour définir des règles de filtrages des paquets dans un Firewall.

La création d'une nouvelle chaîne se fait simplement avec l'option `-N` :

```
iptables [-t table] -N chaîne
```

où `chaîne` est le nom de la chaîne à créer. Il est interdit d'utiliser un des mots-clés réservé par **iptables**. En pratique, il est recommandé d'utiliser des noms de chaînes en minuscules, car les chaînes prédéfinies sont en majuscules.

Une chaîne peut être détruite avec l'option `-X` :

```
iptables [-t table] -X chaîne
```

Une chaîne ne peut être détruite que lorsqu'elle ne contient plus de règles. De plus, il est impossible de détruire les chaînes prédéfinies d'une table.

Vous pouvez lister l'ensemble des règles d'une chaîne avec l'option `-L` :

```
iptables [-t table] -L chaîne
```

Enfin, il est possible de supprimer toutes les règles d'une chaîne avec l'option `-F` :

```
iptables [-t table] -F chaîne
```

où `chaîne` est le nom de la chaîne dont on veut lister les règles.

7.4.5.2. Manipulation des règles

La syntaxe générale pour ajouter une règle dans une chaîne est la suivante :

```
iptables [-t table] -A chaîne [-s source] [-d destination] [-p protocole]
    [-i itfsource] [-o itfdest] [-j cible]
```

où :

- `table` est la table dans laquelle se trouve la chaîne manipulée (par défaut, il s'agit de la table `filter`) ;
- `chaîne` est le nom de la chaîne à laquelle la règle doit être ajoutée ;
- `source` est l'adresse source du paquet ;
- `destination` est l'adresse destination du paquet ;
- `protocole` est le protocole du paquet (spécifié avec son numéro de port ou par le nom du protocole tel qu'il est déclaré dans le fichier `/etc/services`) ;
- `itfsource` est le nom de l'interface réseau source par laquelle le paquet doit arriver ;
- `itfdest` est le nom de l'interface destination par laquelle le paquet doit sortir ;
- et `cible` est la cible des paquets qui vérifient les critères de sélection de la règle.

Chacun des paramètres placé entre crochets dans la syntaxe est facultatif, mais il faut au moins qu'un critère de sélection soit donné (sur les adresses sources ou destination, ou sur le port ou l'interface).

Les adresses sources et destination peuvent être indiquées directement par le nom des machines (comme par exemple « www.monrezo.com ») ou par leurs adresses IP. Vous pouvez spécifier directement toutes les adresses d'un réseau avec la syntaxe « réseau/masque ». Il est également possible d'utiliser la syntaxe « réseau/n », où « n » est le nombre de bits mis à 1 dans le masque de sous réseau. Ainsi, `réseau/24` est équivalent à `réseau/255.255.255.0`, `réseau/16` à `réseau/255.255.0.0`, etc... Par extension, « réseau/0 » peut être utilisé pour spécifier toutes les adresses possibles (dans ce cas, l'adresse donnée pour le réseau n'est pas prise en compte).

La cible détermine les opérations qui seront appliquées à ces paquets. Les cibles autorisées dépendent de la chaîne dans laquelle la règle est ajoutée, ainsi que des modules de Netfilter qui ont été compilés. Les principales cibles sont les suivantes :

- `ACCEPT`, pour accepter le paquet qui vérifie le critère de sélection de la règle ;
- `DROP`, pour éliminer purement et simplement le paquet ;
- `REJECT`, pour rejeter le paquet (en signalant l'erreur à la machine émettrice). Cette cible n'est utilisable que dans les chaînes `INPUT`, `FORWARD` et `OUTPUT`, ainsi que dans les chaînes utilisateurs appelées à partir de ces chaînes ;
- `QUEUE`, pour envoyer le paquets à un programme utilisateur capable de communiquer avec NetFilter ;

- `RETURN`, pour sortir de la chaîne immédiatement, ou appliquer la règle de la politique par défaut pour les chaînes prédéfinies ;
- `REDIRECT`, pour rediriger le paquet sur une autre machine, souvent la machine locale. Cette cible n'est utilisable qu'avec la table `nat`, car il s'agit d'une translation d'adresse. On ne peut l'utiliser que dans les chaînes `PREROUTING` et `OUTPUT` et les chaînes utilisateur appelées à partir de ces chaînes ;
- `SNAT`, pour permettre la modification des adresses sources des paquets. Cette cible n'est bien entendu accessible qu'au niveau de la table `nat`. Comme la modification de l'adresses source n'a de signification que pour les paquets devant sortir de la passerelle, cette cible ne peut être utilisée que dans la chaîne `POSTROUTING` et les chaînes utilisateur appelées à partir de cette chaîne ;
- `DNAT`, pour effectuer la modification des adresses destination des paquets, afin par exemple de les détourner vers une autre machine que celle vers laquelle ils devaient aller. Cette cible n'est accessible que dans la table `nat`, et n'est utilisable que dans les chaînes `PREROUTING` et `OUTPUT` ainsi que dans les chaînes utilisateur appelées à partir de ces chaînes ;
- `MASQUERADE`, pour effectuer une translation d'adresse sur ce paquet, dans le but de réaliser un partage de connexion à Internet. Cette cible n'est accessible que dans la chaîne `POSTROUTING` de la table `nat`, ainsi que dans les chaînes utilisateur appelées à partir de cette chaîne. ;

Toute autre spécification de destination doit être le nom d'une chaîne utilisateur, avec les règles de laquelle le paquet devra être testé. Si aucune cible n'est spécifiée, aucune action n'est prise et la règle suivante est traitée. Cependant, les statistiques de la règle sont toujours mises à jour.

La suppression d'une règle d'une chaîne se fait avec la commande suivante :

```
iptables -D chaîne numéro
```

où `chaîne` est la chaîne dont on veut supprimer la règle, et `numéro` est le numéro de la règle à supprimer dans cette chaîne. Il est également possible d'utiliser l'option `-D` avec les mêmes options que celles qui ont été utilisées lors de l'ajout de la règle, si l'on trouve cela plus pratique.

Enfin, la politique d'une chaîne, c'est à dire la cible par défaut, peut être fixée avec l'option `-P` :

```
iptables -P chaîne cible
```

où `chaîne` est l'une des chaînes prédéfinie, et `cible` est l'une des cibles `ACCEPT` ou `DROP`. Remarquez que l'on ne peut pas définir de politique pour les chaînes créées par l'utilisateur, puisque les paquets reprennent les vérification de la chaîne appelante lorsqu'ils sortent de la chaîne appelée.

7.4.6. Exemple de règles

Ce paragraphe a pour but de présenter quelques unes des règles les plus classiques. Le but est ici de présenter les principales fonctionnalités d'**iptables** et non de réaliser un Firewall fiable. Reportez-vous à des documents plus spécialisés pour cela.

7.4.6.1. Exemple de règles de filtrage

Les règles de filtrage peuvent être utilisées pour mettre en place un Firewall afin de protéger votre réseau. La définition des règles de filtrage constitue bien souvent l'essentiel du problème, et nécessite bien entendu de parfaitement savoir ce que l'on veut faire. De manière générale, la règle d'or en matière de sécurité informatique est de tout interdire par défaut, puis de donner les droits au compte goutte. C'est exactement ce que permet de faire la police des chaînes. On définira donc toujours la police par défaut des chaînes pour utiliser la cible DROP. Ceci peut être réalisé simplement avec les trois commandes suivantes :

```
iptables -P INPUT -j DROP
iptables -P FORWARD -j DROP
iptables -P OUTPUT -j DROP
```

Ces règles commencent par blinder la passerelle pour ne rien laisser entrer, ne rien laisser passer et ne rien laisser sortir. Dans ce cas, on ne craint plus rien, mais on ne peut plus rien faire non plus (pas même en local). Aussi faut-il redonner les droits nécessaires pour permettre l'utilisation normale de votre système. Par exemple, on peut autoriser l'arrivée et l'émission de tous les paquets du réseau local avec les règles suivantes :

```
iptables -A INPUT -s 192.168.0.0/24 -i eth0 -j ACCEPT
iptables -A OUTPUT -s 127.0.0.0/8 -o eth0 -j ACCEPT
```

et autoriser la communication des paquets de la machine locale vers la machine locale (ces paquets sont vitaux !) :

```
iptables -A OUTPUT -s 127.0.0.0/8 -o lo -j ACCEPT
iptables -A INPUT -d 127.0.0.0/8 -i lo -j ACCEPT
```

Comme vous pouvez le constater, ces règles ne permettent l'entrée des paquets prétendant provenir du réseau local (c'est à dire les paquets ayant comme adresse source une adresse de la forme 192.168.0.0/24) que par l'interface réseau connectée au réseau local (dans notre exemple, il s'agit de l'interface réseau `eth0`). De même, seule la machine locale peut émettre des paquets sur le réseau local. Il va de soi que ces paramètres sont trop restrictifs pour réaliser une passerelle. Dans ce cas, on devra relâcher un peu plus de contraintes, et en utilisant des règles de filtrages plus fine. On devra autoriser le routage des paquets (chaîne `FORWARD`), par exemple en n'autorisant que les paquets provenant d'un ensemble de machines considérées comme fiables. Notez que ces règles sont beaucoup trop restrictives pour permettre un accès à Internet (et, a fortiori une intrusion provenant d'Internet…).

Les règles présentées ici se basent uniquement sur les adresses sources et destination des paquets, ainsi que sur les interfaces réseau par lesquelles ils entrent et ressortent. Sachez cependant que l'on peut réaliser des règles beaucoup plus fines, qui se basent également sur les protocoles réseau utilisés, et, pour chaque protocole, sur des critères spécifiques à ces protocoles. Vous pouvez consulter la page de manuel d'**iptables** pour plus de détails à ce sujet.

7.4.6.2. Exemple de partage de connexion à Internet

Le masquerading est un cas particulier de translation d'adresse source. En effet, celle-ci est couplée d'un suivi des connexions, afin d'effectuer la translation d'adresses destination des paquets réponse renvoyés par les serveurs sur Internet pour les acheminer vers la machine du réseau local qui a initié la connexion.

En pratique, seule la chaîne `POSTROUTING` sera utilisée pour le masquerading, parce que c'est à ce niveau que la translation d'adresses est effectuée. La mise en ½uvre du masquerading se fait extrêmement simplement, puisqu'il suffit de spécifier que toutes les paquets du réseau local sortants de l'interface réseau connectée sur Internet doivent subir le masquerading. En général, l'interface de la connexion à Internet est une interface PPP, aussi la commande à utiliser est-elle simplement la suivante :

```
iptables -t nat -A POSTROUTING -s réseau/24 -o ppp0 -j MASQUERADE
```

où `réseau` est l'adresse de votre réseau (par exemple, 192.168.0.0). Notez que la commande donnée ci-dessus suppose que votre réseau soit de classe C, car le masque de sous réseau utilisé est 255.255.255.0. La cible spécifiée pour toutes les paquets provenant de votre réseau local sortant par l'interface ppp0 est donc `MASQUERADE`, ce qui active la translation d'adresses. Remarques que les paquets provenant de la machine locale ne subissent pas le masquerading, car c'est complètement inutile.

> **Note:** Les chaînes et leurs règles ne sont pas enregistrées de manière permanente dans le système. Elles sont perdues à chaque redémarrage de la machine, aussi faut-il les redéfinir systématiquement. Ceci peut être réalisé dans les scripts de démarrage de votre système.
>
> N'oubliez pas que votre passerelle doit définir une route par défaut pour que tous les paquets qui ne sont pas destinés au réseau local soient envoyés par l'interface réseau connectée à Internet. Cette route par défaut est établie automatiquement lorsque vous vous connectez à Internet à l'aide de PPP. Dans les autres cas, vous aurez à la définir manuellement.
>
> Le routage des paquets est, par défaut, désactivé sous Linux. Si votre distribution ne le fait pas, vous aurez également à ajouter une ligne telle que celle-ci :
>
> ```
> # Activation du routage :
> echo "1" > /proc/sys/net/ipv4/ip_forward
> ```
>
> dans vos scripts de démarrage de votre système.
>
> Assurez-vous que les règles de filtrage que vous utilisez permettent bien aux machines du réseau local d'accéder à Internet, et réciproquement, que les machines d'Internet peuvent leur répondre. Sans cela, toute tentative de masquerading sera vouée à l'échec, avant même d'atteindre la chaîne `POSTROUTING`.

7.4.7. Configuration des clients

La configuration des autres machines du réseau est très simple. Vous devrez tout d'abord définir la machine possédant la connexion à Internet comme passerelle par défaut de tous vos postes clients. Cette opération peut se réaliser de différentes manière selon le système d'exploitation utilisé par ces machines. Sous Linux, il suffit d'utiliser la commande suivante :

```
route add default gw passerelle eth0
```

où `passerelle` est l'adresse de votre passerelle vers Internet sur le réseau local. Cette commande suppose que l'adaptateur réseau utilisé est `eth0`

La deuxième étape est ensuite de donner accès aux postes clients au DNS de votre fournisseur d'accès. Ceci permettra en effet d'utiliser les noms de machines autres que ceux de votre réseau local. Encore une fois, la manière de procéder dépend du système utilisé. Sous Linux, il suffit d'ajouter les adresses IP des serveurs de noms de domaines de votre fournisseur d'accès dans le fichier de configuration `/etc/resolv.conf`.

7.5. Configuration des fonctions serveur

En général, les gens qui ne voient Internet que du point de vue client en terme d'architecture client / serveur. Ceci signifie qu'ils se contentent de se connecter à Internet, et n'utilisent que des logiciels clients qui communiquent avec des serveurs situés sur Internet. Ce mode d'utilisation est de loin le plus courant et celui qui convient à la majorité des gens, qui n'ont en général même pas conscience de l'existence du monde des serveurs.

Cependant, il est parfois nécessaire de mettre en place des petits serveurs, sans forcément être un fournisseur d'accès à Internet ou un hébergeur de site Web. Par exemple, il peut être intéressant de fournir la possibilité de réaliser une connexion par téléphone à un ordinateur distant que l'on doit laisser allumer pour une raison ou une autre, afin de réaliser une opération de maintenance. Il est également possible de faire une liaison PPP entre deux ordinateurs par un câble série pour échanger des données, ou pour relier deux réseaux séparés physiquement par l'intermédiaire du réseau téléphonique.

Cette section présente donc les techniques de base permettant de paramétrer Linux pour accepter les communications entrantes, tant pour satisfaire aux demandes de connexions distantes que pour établir une liaison PPP.

7.5.1. Paramétrage des connexions extérieures

Nous avons vu dans la Section 6.9 que le programme en charge de demander l'identification des utilisateurs sur les terminaux de login était le programme getty. Ce programme ne permet aux utilisateurs que de s'identifier. Lorsque quelqu'un saisit son nom, getty se contente de lancer le processus login pour que celui-ci vérifie réalise l'authentification de l'utilisateur en lui demandant son mot de passe. En fait, getty peut très bien lancer un autre programme que login, si l'identification n'est pas nécessaire sur le terminal qu'il gère, mais en général, il faut identifier les utilisateurs.

Le programme getty est donc le point d'entrée par lequel il faut nécessairement passer pour utiliser le système. Dans le cas des terminaux virtuels, getty utilise une ligne de communication virtuelle, mais il peut parfaitement utiliser une ligne de communication réelle (c'est d'ailleurs ce pour quoi il a été conçu à l'origine). En particulier, il est possible de lancer un processus getty sur une ligne série, afin de permettre l'utilisation de terminaux réels connectés sur le port série correspondant.

La ligne série sur laquelle le processus getty fonctionne n'est pas forcément connectée à un terminal physique. Elle peut parfaitement être connectée à un modem. Dans ce cas, les utilisateurs devront appeler le serveur pour se connecter à l'aide d'un logiciel émulateur de terminal de terminal. Il existe plusieurs logiciels permettant d'appeler un ordinateur distant et d'émuler un terminal. On citera par exemple le logiciel HyperTerminal de Windows, et les programmes Telix et Minicom. Ce dernier est couramment fourni avec les distributions de Linux, car c'est un logiciel libre.

En fait, il existe plusieurs variantes de getty, spécialisées pour des usages spécifiques, et permettant de faciliter le paramétrage du fichier de configuration `/etc/inittab`. Les distributions de Linux sont souvent fournies avec les versions suivantes :

- `mingetty`, qui est un getty simplifié, couramment utilisé pour les terminaux virtuels ;
- `mgetty`, qui est une version spécialement conçue pour être utilisée sur les lignes séries connectées à un modem. mgetty est en particulier capable d'initialiser les modems, de décrocher automatiquement la ligne lorsqu'il y a un appel, et de gérer les principales commandes des modems compatible Hayes ;
- `agetty`, qui est une version complète de getty, utilisable aussi bien sur les lignes séries (avec ou sans modem) que pour les terminaux virtuels. On utilisera de préférence cette version si l'on désire proposer une connexion par l'intermédiaire d'un câble null-modem.

> **Note:** Les câbles `null-modem` sont des câbles série symétriques, qui permettent de relier deux ordinateurs par l'intermédiaire de leurs ports série. Ces câbles peuvent être trouvés chez la plupart des revendeurs de matériel informatique (vous pouvez également en fabriquer un vous-même si vous avez du temps à perdre).

Connaissant les diverses possibilités de ces programmes, proposer un service de connexion extérieur est relativement simple. Il suffit en effet de rajouter dans le fichier de configuration `/etc/inittab` les lignes permettant de lancer les programmes getty sur les bonnes lignes. Par exemple, pour offrir une connexion modem sur le deuxième port série (couramment appelé COM2), on ajoutera la ligne suivante :

```
S1:23:respawn:/usr/sbin/mgetty -n 3 -s 38400 ttyS1
```

Cette commande indique à init que mgetty doit être relancé à chaque fois qu'il se termine dans les niveaux d'exécution 2 et 3. Il doit prendre le contrôle de la ligne série utilisée par le fichier spécial de périphérique `/dev/ttyS1`, utiliser la vitesse de communication de 38400 bauds sur cette ligne (option `-s`) et ne décrocher qu'au bout de la troisième sonnerie (option `-n`). Vous pouvez bien entendu ajuster ces options en fonction de vos besoins.

En revanche, si l'on désire offrir une possibilité de connexion sur un port série par l'intermédiaire d'un câble null-modem, on utilisera plutôt le programme agetty. La ligne à ajouter dans le fichier `/etc/inittab` sera alors :

```
S1:123:respawn:/sbin/agetty -L 38400 ttyS1
```

Notez que les options sont quelques peu différentes. La vitesse de la ligne est donnée directement, et l'option `-L` indique que la ligne série utilisée est locale et ne gère pas le signal de porteuse CD (abréviation de l'anglais « Carrier Detect ») habituellement utilisée pour détecter les coupures téléphoniques. Encore une fois, vous pouvez adapter ces commandes selon vos besoins. N'hésitez pas à consulter les pages de manuel des programmes mgetty et agetty pour obtenir plus de renseignements.

> **Note:** Vous pouvez tester vos connexions séries à l'aide de l'émulateur de terminal minicom. S'il est installé sur votre distribution, vous pourrez le lancer simplement en tapant la commande **minicom**. Si vous l'utilisez sur une connexion directe (c'est à dire par un câble null-modem), vous devrez utiliser l'option `-o`, afin d'éviter que minicom n'envoie les codes d'initialisation du modem sur la ligne série.
>
> Il se peut que vous ne parveniez pas à vous connecter sur vos ports série. Dans ce cas, la première des choses à faire est de regarder les paramètres de la ligne série à l'aide de la commande **stty**. Vous devriez vous intéresser tout particulièrement à la vitesse utilisée, et éventuellement vous assurer que les deux ports série des deux ordinateurs utilisent la même vitesse, ou que le modem est capable de gérer la vitesse utilisée par le port série. Si besoin est, vous pouvez modifier la vitesse d'un port série à l'aide de la commande **stty**. Par exemple, pour configurer le port série COM2 à la vitesse de 115200 bauds, vous devez taper la commande suivante :
>
> ```
> stty -F /dev/ttyS1 ispeed 115200
> ```

7.5.2. Configuration des liaisons PPP

Le protocole PPP utilisé par les fournisseurs d'accès à Internet est, comme son nom l'indique (« Point to Point Protocol », ou « protocole de liaison point à point »), un protocole permettant de relier deux ordinateurs uniquement. Ce protocole est complètement symétrique, et la mise en place d'un serveur PPP n'est pas plus compliquée que celle d'un client. Les liaisons PPP sont donc couramment utilisées pour relier deux réseaux distants par l'intermédiaire des lignes téléphoniques.

Lorsque nous avons présenté la configuration des accès à Internet, nous avons utilisé des options spécifiques à ce cas de figure afin de gérer le fait que les paramètres réseau sont imposés par le fournisseur d'accès. Dans le cas d'un serveur, ces paramètres ne sont pas nécessaire, puisque nous sommes cette fois à la place du fournisseur d'accès. En revanche, certains paramètres supplémentaires devront être spécifiés, notamment pour configurer le modem afin d'accepter les connexions externes.

En général, il faut lancer une instance du démon pppd pour chaque ligne série utilisée pour les connexions entrantes, car pppd ne peut gérer qu'une seule connexion à la fois. Cette technique peut paraître lourde, mais en fait elle est relativement pratique, car elle permet de spécifier des options spécifiques à chaque ligne série. Comme d'habitude, ces options peuvent être spécifiées sur la ligne de

commande de pppd, mais elle peuvent également être enregistrées dans un fichier de configuration. Le fichier de configuration global `/etc/ppp/options` est généralement utilisé pour stocker les options communes à toutes les lignes séries. Les options spécifiques à chaque ligne sont enregistrées quant à elles dans les fichiers `options.ttySx`, où `ttySx` est le nom du fichier spécial de périphérique utilisé pour accéder à cette ligne. L'usage de ces fichiers de configuration permettent de simplifier notablement les lignes de commandes utilisées pour lancer pppd.

La ligne de commande à utiliser pour lancer pppd en mode serveur diffère peu de celle utilisée pour établir une connexion PPP en tant que client. Les deux principales différences proviennent du fait que :

- le serveur doit spécifier l'adresse IP qu'il utilisera pour l'interface PPP et l'adresse IP qu'il donnera au client lors de l'établissement de la liaison ;
- le script chat utilisé ne doit pas composer un numéro, mais initialiser le modem pour qu'il réponde aux appels entrants.

Les adresses IP sont spécifiées en ligne de commande à l'aide de la syntaxe suivante :

```
locale:distante
```

où `locale` est l'adresse IP de l'interface PPP du serveur, et `distante` est l'adresse IP que le client recevra lorsqu'il se connectera. Il est important que ces deux adresses ne soient pas sur le même réseau, car les deux machines sont effectivement séparées l'une de l'autre ! Un lien PPP ne constitue donc pas un réseau en soi, mais plutôt un pont entre deux réseaux distincts. Ainsi, l'option suivante :

```
192.168.30.1:192.168.31.1
```

indiquera à pppd que l'adresse IP de l'interface PPP locale sera `192.168.30.1`, et que les clients qui se connecteront sur la ligne correspondante devront recevoir l'adresse `192.168.31.1`.

Notez que les adresses utilisées ici font partie des adresses IP réservées pour les réseaux locaux. Ce n'est pas ce que font les fournisseurs d'accès à Internet en général, car ils disposent d'un ensemble d'adresses IP réelles redistribuables, mais les adresses réservées sont parfaitement utilisables. Le seul cas où cela pourrait être gênant serait si l'on voulait faire passer les adresses des deux passerelles de la liaison PPP sur Internet, mais les paquets utilisant ces adresses ne circulent justement que sur cette liaison (à moins que les passerelles ne se trouvent sur des réseaux locaux et utilisent des adresses de ces réseaux pour la liaison PPP, auquel cas, de toutes manières, il faudrait utiliser la technique du masquerading pour connecter ces réseaux sur Internet).

Le script d'initialisation du modem, quant à lui, peut être spécifié à l'aide de l'option `init` de pppd. Cette option fonctionne exactement comme l'option `connect` que l'on a utilisée pour établir la liaison téléphonique dans la Section 7.3.2. La différence est ici que l'on ne cherche pas à composer un numéro de téléphone, mais plutôt à initialiser le modem pour qu'il décroche automatiquement la ligne lorsqu'il reçoit un appel.

Il existe malheureusement plusieurs dialectes de langages de commandes utilisés par les modems. En général, chaque modem dispose donc de ses propres commandes, et il faut consulter sa documenta-

tion pour savoir leur syntaxe. Malgré cela, ce n'est pas trop gênant, car la plupart des modems sont compatibles avec le standard Hayes, dont les commandes sont bien connues. Pour l'initialisation d'un modem pour un serveur, les commandes d'initialisation suivantes seront très utiles :

Commande	Signification
ATS0=n	Décrochage automatique si `n vaut 1, manuel si n vaut 0.`
AT&C1	Activation du contrôle de la ligne CD (« Carrier Detect ») pour signaler les raccrochages des clients. Cette commande n'est utile que si l'option `modem est passée en paramètre à pppd sur sa ligne de commande.`
AT&K3	Activation du contrôle de flux matériel (pour éviter optimiser les échanges de données entre le modem et l'ordinateur). Cette option nécessite d'utiliser l'option `crtscts dans la ligne de commande de pppd.`

Au final, la ligne de commande à utiliser est donc la suivante :

```
pppd port vitesse detach modem crtscts init "/usr/sbin/chat -v \
        -f /etc/ppp/script" &
```

où `port` est le nom du fichier spécial de périphérique à utiliser pour cette connexion, par exemple `/dev/ttyS0`, `vitesse` est la vitesse de la ligne, par exemple `115200`, et `/etc/ppp/script` est le nom du fichier du script chat devant servir à l'initialisation du modem. L'option `detach` permet de demander à pppd de se détacher du terminal courant, afin que l'on puisse le lancer en arrière plan.

Vous trouverez ci-dessous un script chat d'exemple, permettant de spécifier les options vues ci-dessus :

```
"" ATZ
OK ATS0=1
OK AT&C1
OK AT&K3
OK ""
```

> **Note:** Vous remarquerez qu'il n'est pas nécessaire, contrairement à ce que l'on a vu lors de la configuration d'un client PPP, de spécifier le nom de l'utilisateur et le protocole d'identification et d'authentification utilisé. Ceci est normal, car le serveur n'a pas à s'identifier auprès de la machine cliente en général (il peut bien entendu le faire s'il le désire, mais dans ce cas seuls les clients capables de gérer ce type de connexion pourront se connecter).
>
> En revanche, il peut être nécessaire d'exiger que les clients s'identifient et s'authentifient à l'aide de l'un des protocoles PAP ou CHAP. Pour cela, il suffit simplement d'ajouter les noms des clients et leurs secrets respectifs dans les fichiers de secrets `/etc/ppp/pap-secrets` et `/etc/ppp/chap-secrets`, et d'ajouter l'une des options `require-pap` ou `require-chap` sur la ligne de commande du démon pppd. Remarquez que le format des fichiers `pap-secrets` et `chap-secrets` est le même aussi bien pour les connexions entrantes que pour les appels vers

l'extérieur. Ceci signifie que l'on doit donner d'abord le nom du client, ensuite le nom du serveur, et ensuite son secret et enfin la liste des adresses IP que les clients peuvent utiliser.

Remarquez que la colonne spécifiant la liste des adresses IP ne doit pas rester vide lorsque l'on désire créer un secret pour le serveur. En effet, pppd se base sur l'adresse IP de la machine distante et sur le nom du client pour déterminer le secret à utiliser lors de l'authentification. Cependant, si l'on désire utiliser les mêmes secrets pour toutes les lignes (et donc pour toutes les adresses que le client peut se voir imposer), on pourra utiliser une étoile. Ceci signifie que toutes les adresses correspondront à la ligne, et seul le nom du client sera utilisé pour déterminer le secret à utiliser. Par exemple, si le serveur se nomme `monserveur` et le poste client s'appelle `jdupont`, on utilisera la ligne suivante :

```
#Client      Serveur          Secret     Adresses
jdupont    monserveur         gh25k;f      *
```

Lorsque la liaison PPP est établie, le démon pppd configure l'interface réseau correspondante. Le protocole utilisé par défaut est bien entendu le protocole IP. En général, il est nécessaire de configurer cette interface, afin de spécifier les règles de routage permettant au client d'envoyer et de recevoir des paquets. Lors de l'établissement de la liaison, le démon pppd ajoute automatiquement une règle de routage pour indiquer que tous les paquets à destination de l'adresses du client doivent passer par l'interface ppp de la liaison courante. Cette règle convient dans la majorité des cas, et en tout cas pour les fournisseurs d'accès à Internet, car le client ne dispose que d'une seule machine. En revanche, si la liaison PPP est utilisée pour relier deux réseaux, les règles de routages par défaut ne suffiront plus, car des paquets à destination de toutes les machines du réseau distant doivent être envoyés via l'interface ppp (et inversement). Pour résoudre ces petits problèmes, on devra compléter ou modifier les scripts `/etc/ppp/ip-up` et `/etc/ppp/ip-down`. Ces deux scripts sont appelés par le démon pppd respectivement lors de l'établissement de la liaison PPP et lors de sa terminaison. On placera donc les commandes de routage complémentaires dans le script `ip-up`, et on fera le ménage correspondant dans le script `ip-down`.

Remarquez que ces deux scripts sont communs à toutes les interfaces ppp que les différents démons pppd peuvent utiliser. Ils sont également utilisés lors de l'établissement d'une connexion en tant que client. Leur modification devra donc être entourée d'un soin extrême. Afin de distinguer les différents cas d'utilisation, ces scripts sont appelés avec les paramètres suivants :

Paramètre	Signification
$1	Nom de l'interface réseau (par exemple, « ppp0 ».
$2	Nom du fichier spécial de périphérique de la ligne série utilisée (par exemple, « ttyS0 »).
$3	Vitesse de la ligne série.
$4	Adresse IP locale de l'interface réseau.
$5	Adresse IP du client.

Paramètre	Signification
$6	Paramètre complémentaire, que l'on peut passer au script à l'aide de l'option `ipparam de pppd`.

Un script classique fait un test sur le nom de l'interface réseau et ajuste les règles de routage en fonction de cette interface, en se basant sur les adresses IP locales et distantes reçues en paramètre. La plupart des distributions fournissent un script d'exemple que vous pouvez bien entendu modifier soit directement, soit à l'aide de leur outil de configuration.

Le démon pppd se termine à chaque fois que la liaison PPP se termine. Ce comportement est normal pour un client, mais ce n'est pas généralement ce que l'on cherche à faire pour un serveur. Il faudra donc relancer pppd régulièrement lorsqu'il se terminera. Ceci peut être réalisé en ajoutant sa ligne de commande dans le fichier de configuration `/etc/inittab` dans les niveaux d'exécution adéquats. Comme init impose une taille réduite sur les lignes de commandes des programmes qu'il peut lancer, il est nécessaire d'utiliser les fichiers d'options de pppd. Par exemple, si l'on veut assurer le service PPP pour une ligne entrante sur le deuxième port série dans les niveaux d'exécution 2 et 3, on utilisera la ligne suivante :

```
ppp1:23:respawn:/usr/sbin/pppd /dev/ttyS1 115200 \
192.168.30.1:192.168.31.1"
```

avec les options suivantes activées dans le fichier d'options `/etc/ppp/options.ttyS1` :

```
# Initialisation du modem :
init "/usr/sbin/chat -v -f /etc/ppp/client.ttyS1
# Protocole d'authentification :
require-pap
# Contrôle de la ligne matériel :
modem
# Contrôle de flux matériel :
crtscts
```

7.5.3. Liaison de deux ordinateurs par un câble série

La connexion entre deux ordinateurs par l'intermédiaire d'un câble série est un cas particulier des liaisons PPP. Les techniques utilisées sont les mêmes, mais plusieurs simplifications importantes peuvent être faites :

- les deux ordinateurs sont reliés directement par un câble null-modem, et aucun modem ne s'intercale pour compliquer les choses. Par conséquent, on peut supprimer les scripts chat d'initialisation et de connexion ;

- on a le contrôle physique sur les deux ordinateurs, ce qui implique que les mécanismes d'identification et d'authentification sont superflus. Il est donc inutile de préciser les secrets PAP ou CHAP sur chaque machine ;
- il n'y a que deux ordinateurs à relier, donc les règles de routages sont élémentaires ;
- enfin, les adresses choisies sont connues d'avance et peuvent être spécifiées de manière symétrique sur le client et le serveur.

Il découle de ces simplifications que l'établissement d'une liaison PPP au travers d'un câble null-modem est une opération très facile. Comme vous allez le constater, le protocole PPP est effectivement symétrique.

Sur le « serveur », il suffit de lancer par exemple la commande suivante :

```
pppd /dev/ttyS0 115200 192.168.30.1:192.168.31.1 detach crtscts &
```

Cette commande permet de lancer le démon pppd sur le port série COM1 (fichier spécial de périphérique `/dev/ttyS0`) en utilisant la vitesse de 115200 bauds (vitesse maximale généralement supportée par les ports série des ordinateurs), en utilisant l'adresse IP locale `192.168.30.1` et l'adresse IP distante `192.168.31.1`. Le démon pppd doit se détacher du terminal, car il est lancé en arrière plan. Enfin, le contrôle de flux matériel est utilisé (option `crtscts`).

La commande sur le « client » est absolument symétrique, puisque seules les adresses locales et distantes sont interverties :

```
pppd /dev/ttyS0 115200 192.168.31.1:192.168.30.1 detach crtscts &
```

Bien entendu, le port série utilisé peut être différent de celui du serveur.

Une fois ces deux commandes lancées, la connexion PPP est établie. Le démon pppd ajoute alors automatiquement une règle de routage pour acheminer les paquets d'un ordinateur à l'autre. Vous n'aurez donc pas à modifier les scripts `/etc/ppp/ip-up` et `/etc/ppp/ip-down`. Comme vous pouvez le constater, l'utilisation de PPP pour relier simplement deux ordinateurs n'est pas une opération très compliquée…

7.6. Installation d'un proxy

Lors d'une connexion à Internet, bon nombre d'informations sont échangées entre le serveur et le client. En particulier, lorsque l'on navigue, un certain nombre d'images et d'informations plus ou moins lentes à télécharger transitent systématiquement. Or la plupart des gens visitent régulièrement les mêmes sites, dont seulement quelques pages sont modifiées entre deux visites successives. Par conséquent, on peut se demander pourquoi les pages complètes devraient être rechargées à chaque visite, si seulement quelques informations ont été modifiées.

Cette question prend encore plus de sens lorsque l'on réalise un partage de connexion à Internet. Il est tout à fait concevable que plusieurs clients du même réseau demandent plusieurs fois les mêmes

informations, ce qui provoque l'engorgement du lien partagé inutilement. C'est pour résoudre ce genre de problème que les programmes que l'on nomme « proxies » ont été développé.

Un *proxy* n'est rien d'autre qu'un programme qui s'intercale entre des clients et un serveur. Il a pour principale fonction de mémoriser les réponses les plus courantes renvoyées par le serveur, afin de répondre à la fois plus rapidement au serveur et d'éviter de surcharger le serveur. Il existe différents types de proxy, mais nous allons nous intéresser uniquement aux proxies Web, qui permettent donc de stocker des pages Web en local afin de soulager une liaison trop faible.

Le principe de fonctionnement d'un proxy est le suivant. Les navigateurs utilisés par les clients se connectent, en temps normal, directement sur le port 80 des serveurs Web. Leur configuration doit être modifiée pour utiliser le proxy à la place. De leur point de vue, le proxy se comporte comme un serveur Web capable de répondre aux requêtes des clients. Lorsqu'il peut le faire, il renvoie les données qu'il a stockées dans son cache, sinon, il va les chercher directement sur Internet à la place du client. Le cache est maintenu de telle manière que les données obsolètes ou les données les moins utilisées sont supprimées, afin de laisser la place aux données qui changent qui sont le plus souvent demandées. Pour utiliser un proxy, il faut donc modifier la configuration des navigateurs des clients, afin de donner l'adresse de la machine sur laquelle le proxy fonctionne, ainsi que le port sur lequel il peut être contacté (généralement, c'est le port 8080).

Il existe plusieurs proxies sur le marché, mais le plus utilisé sous Linux est sans doute squid. C'est un logiciel performant, puissant et libre, fourni avec la plupart des distributions modernes. Sa configuration peut être relativement technique, mais nous n'allons pas entrer dans tous les détails de sa configuration. Il s'agit simplement ici de donner un aperçu des fonctionnalités disponibles.

squid utilise le fichier de configuration `/etc/squid.conf` pour lire ses options de fonctionnement. Ce fichier contient en particulier les informations concernant le réseau, une liste de règles indiquant qui a le droit de lui demander des pages Web, et l'emplacement du cache dans lequel les données partagées seront enregistrées.

L'option la plus importante dans ce fichier de configuration est sans doute `cache_dir`, qui indique l'emplacement du cache ainsi que sa taille et sa structure. La syntaxe de cette option est la suivante :

```
cache_dir répertoire taille n m
```

où `répertoire` est le répertoire dans lequel le cache sera stocké (usuellement `/var/squid/cache/`), `taille` est la taille de ce cache en méga-octets, et `n` et `m` deux nombres donnant la structure de l'arborescence du cache. La signification de ces deux nombres doit être expliquée un peu plus en détail.

Par défaut, squid essaie de répartir tous les objets qu'il place dans le cache de manière uniforme dans plusieurs répertoires, afin d'accélérer les temps de recherche sur ces objets lorsqu'un client les demande. Il utilise à cette fin une *fonction de répartition* (les programmeurs disent souvent une fonction de *hash*) qui indique dans quel répertoire se trouve un objet. La recherche dans ce répertoire se fait donc plus rapidement, car si la fonction de répartition est bien faite (et c'est le cas), les objets sont répartis de manière équilibrée dans tous les répertoires du cache, qui sont donc tous relativement peu peuplés. En fait, squid peut stocker tellement d'objets que le nombre de répertoires peut lui-même devenir très grand. Il utilise donc un découpage à deux niveaux, les répertoires du deuxième niveau étant répartis dans les répertoires du premier niveau. Le premier nombre spécifié dans l'option

`cache_dir` indique le nombre de répertoire du premier niveau, et le deuxième nombre indique le nombre de sous-répertoires dans chacun de ces répertoires. En général, on utilise respectivement les valeurs 16 et 256 :

```
cache_dir /var/squid/cache 100 16 256
```

Cette ligne permet de cacher 100 Mo, en répartissant les objets dans 4096 répertoires répartis en 16 groupes de 256 répertoires.

Les options suivantes spécifient les paramètres réseau du cache :

- L'option `http_port` indique le port TCP que les clients doivent utiliser pour accéder au proxy ;
- L'option `ftp_user` permet de donner l'adresse email à utiliser comme mot de passe dans les connexion FTP anonymes ;
- L'option `cache_peer` permet d'intégrer le proxy local dans une hiérarchie de proxies.

  Les proxies peuvent en effet être organisés dans une structure arborescente. Lorsqu'un proxy d'un niveau ne dispose pas d'une donnée demandée par un client, il peut contacter ses frères pour le cas où ceux-ci auraient cette donnée dans leur cache, ou demander à son proxy père de récupérer cette donnée pour lui. Les proxies communiquent entre eux en utilisant un protocole dédié, le protocole *ICP* (abréviation de l'anglais « Inter Cache Protocol »).

  En général, les petits réseaux ne disposent que d'un seul proxy, l'option `cache_peer` est donc souvent utilisée de la manière suivante :

  ```
  cache_peer père parent port port_icp options
  ```

  où `père` est le nom du cache père, `port` est son port, `port_icp` est le port à utiliser pour les communications ICP, et `options` sont des options complémentaires. Parmi toutes les options possibles, on utilisera le plus souvent l'option `default`, qui spécifie les valeurs par défaut des options, et `no-query`, qui permet de dire au cache de ne pas utiliser le protocole ICP (dans ce cas, le paramètre `port_icp` sera inutilisé) ;
- L'option `prefer_direct` permet d'indiquer au proxy s'il doit chercher avant tout à récupérer lui-même les données qui lui manquent (valeur `on`) ou s'il doit contacter d'abord son proxy père (valeur `off`). Si l'on a spécifié un proxy père à l'aide de l'option `cache_peer`, on aura intérêt à utiliser la valeur `off` ;
- L'option `dns_children` permet de donner le nombre de processus en charge de cacher les requêtes sur les DNS.

  Les requêtes de DNS font partie des requêtes les plus courantes lorsque l'on navigue sur Internet. En effet, la moindre adresse, le moindre lien suivi nécessite une requête DNS. On pourrait limiter ce traffic en installant un serveur DNS local, mais c'est un peu compliqué et peut poser des problèmes de sécurité. squid propose donc une autre solution, qui consiste à cacher les requêtes DNS et à fournir des processus locaux prenant en charge la résolution de noms.

Plus le réseau local est important, plus il faudra de tels processus, afin que tous les clients puissent obtenir les adresses IP des machines à partir de leur noms. La valeur par défaut est de 5 processus, ce qui devrait convenir dans la plupart des cas.

Les valeurs données par défaut à toutes ces options conviennent dans la plupart des cas. Elles sont parfaitement documentées dans le fichier de configuration `squid.conf`.

Par défaut, squid n'autorise personne à accéder au cache. Il est donc nécessaire de donner les droits nécessaires pour permettre aux machines de votre réseau d'accéder au cache. La gestion de la sécurité se fait par l'intermédiaire de ce que l'on appelle des ACL (abréviation de l'anglais « Access Control List »).

Une *ACL* est simplement un critère permettant de classifier les requêtes que le proxy reçoit. Ces critères sont définis à l'aide du mot-clé `acl`, suivi du nom de la liste, suivi lui-même d'un critère que toutes les requêtes concernées par cette ACL devront vérifier. Les critères sont eux-mêmes exprimés à l'aide d'un mot-clé qui indique sur quoi porte le critère et des options de ce critère. Les principaux mots-clés que vous pourrez utiliser sont les suivants :

- `src`, qui permet de sélectionner filtrer les requêtes par les adresses des machines qui les ont émises, en indiquant l'adresse de leur réseau donné sous la forme `adresse/masque` ;
- `dst`, qui permet de filtrer les requêtes par leurs adresses destination ;
- `port`, qui permet de sélectionner les requêtes s'adressant sur les ports fournis en paramètres ;
- `proto`, qui permet de spécifier la liste des protocoles utilisés par les requêtes ;
- `method`, qui permet de sélectionner les requêtes HTTP par la méthode utilisée dans la requête.

Ce tableau n'est pas exhaustif, mais il permet d'avoir une idée des capacités de filtrage des requêtes dont dispose squid. Vous trouverez ci-dessous quelques exemples pratiques de définitions d'ACL :

```
# ACL caractérisant toutes les machines :
acl all src 0.0.0.0/0.0.0.0

# ACL définissant la machine locale :
acl localhost src 127.0.0.1/255.255.255.255

# ACL définissant les machines d'un réseau privé local :
acl localnet src 192.168.0.0/255.255.255.0

# ACL caractérisant le proxy lui-même :
acl manager proto cache_object

# ACL spécifiant les ports acceptés par le proxy :
acl safe_ports port 80 21 443 563 70 210 1025-65535

# ACL définissant les requêtes de connexions :
acl connect method CONNECT
```

La définition des règles de sécurité utilisent les ACL de manière séquentielle. Les règles doivent être données les unes après les autres, suivant le format suivant :

```
ressource politique ACL
```

où `ressource` est un mot-clé indiquant la ressource sur laquelle la règle de sécurité porte, `politique` est l'action prise pour les requêtes vérifiant cette règle, et `ACL` une liste d'ACL permettant de spécifier les requêtes concernées par cette règle. La ressource la plus utilisée est sans doute le protocole HTTP, que l'on peut spécifier à l'aide du mot-clé `http_access`. Les actions peuvent être l'acceptation (mot-clé `allow`) ou le refus (mot-clé `deny`) de la requête. Enfin, les requêtes concernées par une règle sont celles qui appartiennent à toutes les ACL de la liste d'ACL indiquée.

Vous trouverez ci-dessous quelques exemples de règles de sécurité courantes :

```
# Autorise les accès au gestionnaire de cache local :
http_access allow manager localhost

# Interdit les accès aux gestionnaires de cache étrangers :
http_access deny manager

# Interdit les accès pour toutes les requêtes provenant de port non autori-
sés :
http_access deny !safe_ports
http_access deny connect !safe_ports

# Autorise les accès aux clients de la machine locale :
http_access allow localhost

# Autorise les accès aux machines du réseau local :
http_access allow localnet

# Politique par défaut :
http_access deny all
```

Notez que si une requête ne correspond à aucune règle, la politique par défaut utilisée par squid est de prendre l'action opposée à la dernière règle spécifiée. En pratique, il est plus sage de toujours indiquer une politique par défaut pour toutes les requêtes restantes, par exemple en utilisant l'ACL `all` définie ci-dessus.

L'exemple de jeu de règles de sécurité donné ci-dessous convient pour un réseau local d'adresses 192.168.0.0. Vous pouvez bien entendu modifier les options de sécurité comme bon vous semble. Encore une fois, rappelons que le fichier de configuration de squid est très bien documentée et que sa lecture est relativement facile. Je ne saurais que trop vous recommander de le consulter si vous désirez en savoir plus.

7.7. Systèmes de fichiers en réseau

L'une des utilités principales d'un réseau est de permettre le partage des fichiers entre plusieurs ordinateurs. Ceci nécessite de disposer d'un système de fichiers réseau sur les machines qui désirent accéder à des fichiers d'une autre machine, et d'un programme serveur sur les machines ainsi accédées par les clients.

Il existe deux grands standards de protocole pour les systèmes de fichiers en réseau. Le premier standard fonctionne quasiment exclusivement sous Unix, il s'agit du protocole « NFS » (abréviation de l'anglais « Network File System »), introduit originellement par Sun Microsystems et repris par la suite par les autres éditeurs de systèmes Unix. Le deuxième protocole est le protocole « SMB » (abréviation de l'anglais « Server Message Block »), qui a été introduit par Microsoft pour les systèmes DOS et Windows.

Les deux protocoles sont incompatibles, et si les deux solutions fonctionnent parfaitement en environnements homogènes, il est assez difficile de faire communiquer les deux types de systèmes. Les principaux problèmes proviennent de l'impossibilité d'utiliser le protocole réseau NetBIOS d'IBM sur les machines Unix, et des limitations des systèmes Microsoft en ce qui concerne la gestion des utilisateurs, les droits d'accès et les liens symboliques. De plus, la différence de gestion des fins de lignes dans les fichiers textes entre Unix et les systèmes Microsoft pose des problèmes qui peuvent difficilement être résolues de manière systématique.

Malgré ces limitations, les machines fonctionnant sous DOS ou Windows peuvent accéder aux systèmes de fichiers NFS des machines Unix grâce à des programmes spéciaux. Ces programmes sont souvent des extensions propriétaires aux systèmes Microsoft, et peuvent être relativement coûteux sans pour autant garantir une grande fiabilité et des performances honnêtes. L'utilisation de NFS sur les machines Windows ne sera donc pas traitée ici. Inversement, les machines Unix peuvent accéder aux partages Microsoft, mais là encore, il peut se présenter des difficultés. La principale difficulté est ici la nécessité d'encapsuler NetBIOS dans le protocole TCP/IP. En effet, le protocole SMB utilise les messages NetBIOS, protocole qui est n'est pas géré nativement par les machines Unix. Dans tous les cas, des logiciels complémentaires sont nécessaires.

7.7.1. Installation d'un serveur de fichiers NFS

L'installation d'un serveur NFS est la solution de prédilection pour le partage des fichiers sous Unix. Il faut savoir cependant que NFS n'offre pas des performances remarquables d'une part, et qu'il ouvre des brèches dans la sécurité du système d'autre part.

Comme il l'a été dit ci-dessus, l'installation d'un serveur NFS requiert l'utilisation de programmes spécifiques. Plus précisément, NFS utilise deux démons pour fournir les services NFS aux machines clientes. Ces deux démons se nomment respectivement « mountd » et « nfsd ». Le premier démon prend en charge les requêtes de montage des systèmes de fichiers NFS, et nfsd effectue les requêtes d'accès aux fichiers pour les clients. Par ailleurs, ces deux démons utilisent les mécanismes d'appels de procédure à distance « RPC » (abréviation de l'anglais « Remote Procedure Call » de Sun Microsystems. Ces services sont désormais disponibles sur toutes les machines Unix et constitue la norme en la matière. Il est donc également nécessaire de mettre en place les services RPC avant de lancer les démons NFS.

Le principe de fonctionnement des appels de procédures à distance est le suivant. Chaque programme

désirant fournir des services RPC écoute sur un port TCP ou UDP pour des requêtes éventuelles. Les clients qui veulent utiliser ces services doivent envoyer leurs requêtes sur ce port, en précisant toutes les informations nécessaires à l'exécution de cette requête : numéro de la requête et paramètres de la requête. Le serveur exécute cette requête et renvoie le résultat. Les librairies RPC fournissent les fonctions nécessaires pour effectuer le transfert des paramètres et les appels à distance eux-mêmes.

En pratique cependant, les clients ne savent pas sur quel port le serveur RPC attend leurs requêtes. Un mécanisme a donc été mis en place pour leur permettre de récupérer ce port et communiquer ensuite avec le serveur. Chaque serveur RPC est identifié par un numéro de programme unique, ainsi qu'un numéro de version. Lorsqu'ils démarrent, les serveurs s'enregistrent dans le système en précisant le port sur lequel ils écouteront les requêtes. Les clients peuvent alors interroger le système distant pour demander le port sur lequel ils trouveront un serveur donné, à partir du numéro de programme et du numéro de version de celui-ci.

Il existe donc un service RPC particulier, nommé « portmapper », qui fournit aux clients qui le demandent les numéros de ports des autres serveurs. Bien entendu, le portmapper doit être toujours contactable, ce qui implique qu'il utilise systématiquement le même numéro de port. Par convention, le portmapper est identifié par le numéro de programme 100000, et il écoute les requêtes des clients sur les ports 111 des protocoles TCP et UDP.

La mise en place des serveurs de fichiers passent donc par le lancement de ces trois démons : le démon RPC, le démon mountd et le démon nfsd. Les fichiers exécutables de ces démons sont respectivement `portmap`, `rpc.mountd` et `rpc.nfsd`. Ils sont normalement placés dans le répertoire `/sbin/` ou `/usr/sbin/`. Vous devrez donc lancer ces trois démons sur la machine serveur de fichiers, il est probable que votre distribution fasse le nécessaire dans ses scripts de démarrage.

> **Note:** Il est prévu d'intégrer les fonctionnalités de serveur NFS dans le noyau de Linux. Cependant, le serveur de fichiers du noyau n'est pas encore finalisé, et ne sera donc pas décrit ici.

Une fois les démons lancés, vous pourrez configurer les systèmes de fichiers exportés par votre serveur. Ces systèmes de fichiers sont en fait de simples répertoires, que vous mettez à disposition à certaines machines. La liste de ces répertoires est placée dans le fichier de configuration `/etc/exports`. Chaque ligne de ce fichier caractérise un répertoire accessible par les autres machines du réseau. Ces lignes utilisent le format suivant :

```
répertoire    machines
```

où `répertoire` est le chemin sur le répertoire à exporter, et `machines` est une liste de machines pouvant accéder à ce répertoire. Cette liste contient des noms de machines, des adresses IP ou des noms de réseaux, séparés par des espaces. Il est également possible d'utiliser les caractères génériques ' ?' et '*' afin de spécifier des groupes de machines.

Des options peuvent être utilisées pour chaque machine, à l'aide de la syntaxe suivante :

```
machine(options)
```

où `machine` est l'une des entrées de la liste de machines d'une ligne du fichier exports, et `options` est la liste des options pour cette entrée, séparées par des virgules. Les options les plus utilisées sont bien

entendu `ro` et `rw`, qui permettent de fournir à cette machine ou à ce groupe de machine respectivement un accès en lecture seule et en lecture et écriture sur le répertoire.

Le problème fondamental de NFS est la sécurité. En effet, les fichiers sont exportés par défaut avec un identificateur d'utilisateur qui est celui qui possède le fichier sur la machine serveur. Or il est tout à fait possible que cet identificateur ne correspondent pas au même utilisateur sur tous les clients. Ceci signifie que les utilisateurs des machines clientes peuvent parfaitement avoir accès à des fichiers qui ne leur appartient pas sur le serveur. Ce problème est fondamental, aussi faut-il le prendre en considération sérieusement.

NFS fourni plusieurs solutions pour assurer la sécurité sur les systèmes de fichiers exportés. La première, qui est aussi la plus restrictive, est d'attribuer les fichiers exportés à un utilisateur ne disposant de quasiment aucun droit (opération que l'on nomme souvent « squashing de l'utilisateur »). Cet utilisateur spécial est l'utilisateur « nobody », dont l'identificateur est 65534 par défaut. Ainsi, tous les clients accèdent à ces fichiers au nom de l'utilisateur nobody, et ne peuvent donc pas modifier les fichiers du serveur. Le squashing de l'utilisateur root est toujours réalisé par défaut, pour des raisons de sécurité évidentes.

La deuxième solution est nettement moins sûre. Elle nécessite de lancer le démon « ugidd » sur chaque machine client. Ce démon est appelé par le serveur NFS pour déterminer l'identificateur des utilisateurs du client à partir de leur nom. Ainsi, chaque fichier est exporté avec l'identificateur de l'utilisateur qui porte le même nom que celui qui possède le fichier accédé sur le serveur. Les problèmes de sécurité posés par cette solution sont énormes : rien ne garantit que deux utilisateurs distincts sur deux machines différentes ne puissent pas avoir le même nom d'une part, et un attaquant potentiel peut utiliser le démon ugidd pour obtenir la liste des utilisateurs de la machine cliente d'autre part (ce qui constitue déjà la moitié du travail pour s'introduire dans le système de la machine cliente). Cependant, cette solution est très pratique pour les réseaux dont on contrôle chaque machine, à condition de restreindre l'accès au démon ugidd au serveur uniquement, par exemple en ayant recours à tcpd.

La troisième solution est de définir sur le serveur l'association entre les identificateurs des utilisateurs du serveur et les identificateurs du client, ce pour chaque machine cliente. Cette technique est sûre, mais nettement plus compliquée à mettre en ½uvre.

Enfin, la dernière solution est d'utiliser les services d'information du réseau NIS (abréviation de l'anglais « Network Information Service »), qui permettent de définir un certain nombre d'information de manière globale sur un réseau. En particulier, il est possible de centraliser la définition des utilisateurs et des mots de passe. Cette solution est très lourde à mettre en ½uvre, puisqu'elle nécessite de configurer NIS au préalable. Elle n'est donc mise en place que sur les grands réseaux, et un particulier n'a pas de raisons d'y recourir (à moins de vouloir explorer ces mécanismes bien entendu). Nous n'en parlerons donc pas.

Toutes ces techniques peuvent être activées à l'aide d'options fournies dans la liste des options pour chaque machine déclarées dans le fichier de configuration `/etc/exports`. Les options utilisées sont décrites ci-dessous :

- l'option `all_squash` permet d'exporter tous les fichiers comme appartenant à l'utilisateur nobody. C'est l'option la plus sûre, mais aussi la plus restrictive ;

- l'option `map_daemon` permet d'utiliser le démon ugidd, qui doit être lancé sur les machines clientes et accessibles du serveur. L'accès aux répertoires exportés sera refusé si ce démon ne peut être contacté par le serveur NFS. C'est certainement la solution la plus pratique pour un particulier ;
- l'option `map_static=fichier` permet d'utiliser un fichier de correspondance des identificateurs des utilisateurs du serveur NFS sur les identificateurs des utilisateurs de la machine cliente.

Comme on peut le voir, cette dernière option permet de définir spécifiquement, pour chaque machine, la correspondance des identificateurs d'utilisateurs. Le fichier fourni en paramètre à l'option `map_static` contient un certain nombre de lignes, chacune définissant l'association entre les identificateurs d'utilisateurs de la machine distante et les identificateurs de ces utilisateurs sur le serveur. Ces lignes sont introduites à l'aide du mot-clé `uid`. Il est également possible de donner les correspondances sur les groupes des utilisateurs avec le mot-clé `gid` :

```
# Exemple de fichier de correspondance d'identificateurs :
#    client     serveur
uid  0-99       -
uid  500-1000   1000
gid  0-49       -
gid  50-100     1000
```

Dans l'exemple donné ci-dessus, les utilisateurs ayant un identificateur compris entre 0 et 99 inclus seront associés à l'utilisateur nobody (ils subissent le « squashing »). Il en est de même pour les groupes allant de 0 à 49. En revanche, les utilisateurs dont les identificateurs vont de 500 à 1000 sur la machine cliente se voient respectivement considérés comme les utilisateurs d'identificateur 1000 à 1500 par le serveur NFS. De même, les groupes d'identificateurs 50 à 100 sont considérés comme les groupes d'identificateurs 1000 à 1050.

L'exemple qui suit va permettre d'éclaircir un peu ces notions. Il montre comment les trois types de contrôle des identificateurs peuvent être mis en place, pour trois types de clients différents :

```
# Exemple de fichier /etc/exports :

# Répertoire /pub : accessible de tout le monde, avec les droits
# de l'utilisateur nobody :
/pub    (ro,all_squash)

# Répertoire /home : accessible de toutes les machines du réseau local,
# avec contrôle des utilisateurs effectuée par le démon rpc.ugidd :
/home   *.monrezo.org(map_daemon)

# Répertoire /usr : accessible par la machine 192.168.5.2, avec mapping
# des utilisateurs :
/usr    192.168.5.2(map_static=/etc/nfs/192.168.5.2.map)
```

Le fichier `/etc/nfs/192.168.5.2.map` utilisé dans la troisième ligne définit la correspondance entre les utilisateurs de la machine 192.168.5.2 et le serveur NFS :

```
# Fichier de correspondance pour 192.168.5.2 :
#       client  serveur
uid     0-99    -
uid     500     507
gid     0-49    -
gid     50      50
```

Notez que toute modification du fichier `/etc/exports` doit être signalée aux démons mountd et nfsd, pour qu'ils puissent le relire. Cette notification peut être effectuée en envoyant le signal `SIGHUP` à ces deux démons. Les deux commandes suivantes suffiront à effectuer cette tâche :

```
killall -HUP rpc.mountd
killall -HUP rpc.nfsd
```

7.7.2. Configuration d'un client NFS

Une fois la configuration du serveur NFS effectuée, il ne reste plus qu'à configurer les postes clients. Bien entendu, la manière de réaliser ceci dépend du système utilisé. Nous allons voir ici comment accéder à un répertoire exporté par un serveur à partir d'une machine fonctionnant sous Linux.

Du point de vue du client, les fichiers accédés par NFS sont considérés comme des fichiers normaux d'un système de fichiers classique. Ils sont donc gérés directement par le noyau, en tant que système de fichiers à part entière. L'utilisation de NFS est donc directe, pour peu que l'on ait activé les fonctionnalités NFS dans le noyau.

Encore une fois, vous devrez modifier la configuration du noyau si celui fourni avec votre distribution ne gère pas les systèmes de fichiers NFS (ce qui est fort peu probable). La seule fonctionnalité à activer, en plus de la gestion du réseau bien entendu, est l'option « `NFS filesystem support` » du menu « `Network File Systems` » (ce menu est un sous-menu du menu « `Filesystems` » si vous utilisez la configuration en mode texte). La compilation du noyau sera décrite en détail ultérieurement.

Lorsque le noyau sera compilé et installé, il sera capable d'utiliser les systèmes de fichiers NFS nativement. Vous n'aurez donc pas besoin des démons nfsd et mountd, tout sera pris en charge par le noyau. Cependant, si le serveur exporte ses répertoires avec l'option `map_daemon`, il sera nécessaire de lancer le démon ugidd sur la machine cliente (le fichier exécutable se nomme `rpc.ugidd` et se trouve dans le répertoire `/usr/sbin/`). Il est conseillé de lancer ce démon à partir d'inetd, en l'encapsulant à l'aide du démon tcpd.

Le montage d'un système de fichiers NFS se fait classiquement, avec la commande **mount**. Il suffit simplement ici de préciser le type de système de fichiers « `nfs` » avec l'option `-t`, et d'utiliser la syntaxe suivante pour le fichier spécial de périphérique à utiliser :

```
machine :répertoire
```

où `machine` est le nom du serveur NFS, et `répertoire` est le chemin absolu sur le répertoire exporté par cette machine et auquel on désire accéder. Ainsi, si la machine « `mon.serveur.nfs` » exporte le

répertoire « `/mon/répertoire/exporté` », la commande suivante permettra de monter ce système de fichiers NFS dans le répertoire `/mnt` :

```
mount -t nfs \
    mon.serveur.nfs :/mon/répertoire/exporté /mnt
```

Le système de fichiers NFS accepte des options permettant d'optimiser les transferts d'informations. Ces options peuvent être fournies en ligne de commande à mount à l'aide de l'option `-o`. Les plus utiles sont sans doute `rsize`, qui permet de fixer la taille des blocs de données transférés pour la lecture des fichiers, et `wsize`, qui permet de fixer cette taille pour l'écriture des fichiers. Il est recommandé d'utiliser des paramètres raisonnables afin d'éviter des erreurs de transfert et des pertes de données. La valeur par défaut est 1024, mais il est recommandé d'utiliser des blocs de taille 8192 pour obtenir de meilleures performances. Ainsi, la commande suivante pourra être optimisée comme suit :

```
mount -t nfs -o rsize=8192,wsize=8192 \
    mon.serveur.nfs :/mon/répertoire/exporté /mnt
```

Bien entendu, ces valeurs dépendent de la bande passante de votre réseau, et vous aurez sans doute à effectuer des tests de transferts de fichiers pour trouver les valeurs optimales.

7.7.3. Installation d'un serveur de fichiers SMB

Bien que les protocoles SMB et NFS soient profondément différents, les principes utilisés pour mettre en place un serveur SMB sont quasiment les mêmes que ceux utilisés pour un serveur NFS. L'installation d'un serveur SMB requiert en effet l'utilisation de deux démons qui prennent en charge les services fournis par les serveurs de fichiers SMB. Ces démons utilisent tous deux le fichier de configuration `smb.conf`, placé le plus souvent dans le répertoire `/etc/`. Seuls les serveurs doivent faire fonctionner ces démons, les clients Linux quant à eux se contentent d'utiliser le système de fichiers SMB du noyau.

Ces démons, ainsi que tous les outils nécessaires à l'utilisation du protocole SMB, sont fournis dans la suite logicielle « Samba », dont le nom est évidemment inspiré de « SMB ». Cette suite logicielle est distribuée sous la licence GNU et est donc complètement libre d'utilisation. Elle est normalement fournie avec toutes les distributions. La version actuelle est la 2.0.7., si ce n'est pas celle fournie avec votre distribution, il est fortement recommandé d'effectuer une mise à jour.

L'installation de Samba ne pose pas de problèmes particuliers. Nous ne la décrirons donc pas ici. Sachez cependant qu'elle est recommandée même sur les postes clients, car bien que les démons ne soient pas lancés sur ceux-ci, les outils permettant de se connecter à un serveur de fichiers sont fournis avec Samba et restent nécessaires.

Comme on l'a vu ci-dessus, le protocole SMB se base sur le protocole de bas niveau NetBIOS. NetBIOS est un protocole très primitif, qui n'utilise pas un mécanisme d'adressage semblable à celui de la plupart des protocoles modernes. Chaque machine est identifiée sur le réseau par un nom unique, qu'elle doit déclarer avant de pouvoir utiliser les ressources du réseau. Tout nouvel arrivant sur le

réseau signale donc aux autres qu'il désire utiliser son nom, afin de déterminer si ce nom est déjà pris ou non par une autre machine.

L'un des plus gros défaut de NetBIOS est tout simplement qu'il n'est pas routable tel quel, et qu'il est impossible de structurer un réseau en se basant seulement sur les noms NetBIOS. C'est pour cette raison que l'on encapsule souvent NetBIOS dans un protocole plus évolué capable de traverser les routeurs, comme TCP/IP par exemple. C'est exactement ce que fait Samba, qui apparaît donc comme un service réseau Unix classique permettant de comprendre les paquets NetBIOS encapsulés dans les paquets TCP/IP. Cette technique suppose évidemment que tous les clients du réseau, y compris les postes Windows, soient configurés pour encapsuler NetBIOS dans TCP/IP. L'utilisation de TCP/IP est de toutes manières fortement recommandée si l'on veut réaliser un réseau un tant soit peu sérieux.

Outre les noms d'ordinateurs, le protocole NetBIOS permet de définir des groupes d'ordinateurs, qui contiennent tous les ordinateurs supposés échanger des informations couramment. Un ordinateur ne peut faire partie qu'à un seul de ces groupes, il peut également ne faire partie d'aucun groupe s'il le désire. Ces groupes d'ordinateurs sont créés uniquement en fonction du travail qu'ils ont à effectuer, et ne sont a priori absolument pas lié à leur emplacement géographique ni à leur emplacement sur le réseau. Ainsi, il est possible de communiquer avec tous les ordinateurs d'un groupe, un peu comme le protocole TCP/IP permet d'utiliser des adresses de diffusion. Dans le monde Windows, ces groupes de machines sont plus couramment dénommés des « *workgroup* », et il n'y a en fait aucune différence technique. Pour le protocole SMB, chaque ordinateur du réseau doit faire partie d'un groupe bien défini, bien que ce ne soit pas une nécessité au niveau de NetBIOS.

Lorsqu'un groupe de travail possède une base de donnée centrale permettant de réaliser l'authentification des utilisateurs lors des accès aux machines par le réseau, on dit qu'il s'agit d'un *domaine*. En pratique, un domaine n'est donc rien d'autre qu'un groupe de travail dont fait partie un ou plusieurs serveurs d'authentification. Il existe plusieurs types de serveurs de domaines. Un *serveur primaire* est un serveur central, sur lequel se fait la gestion globale des utilisateurs. Outre un serveur primaire, un domaine peut également contenir un ou plusieurs *serveurs de sauvegarde* si nécessaire. Comme leur nom l'indique, ces serveurs sont capables de remplacer le serveur primaire en cas de défaillance de celui-ci. Ils maintiennent donc une copie des tables d'utilisateurs, et la synchronisent régulièrement avec celle du serveur primaire. Enfin, un domaine peut contenir des *serveurs membres*, qui sont comme des serveurs de sauvegarde, à ceci près qu'ils ne peuvent pas remplacer le serveur de domaine primaire. Bien entendu, le domaine contient également les stations de travail classiques, qui effectuent des requêtes sur le serveur primaire.

Afin de gérer la liste des noms NetBIOS du réseau, Samba fournit le démon « nmbd ». Ce démon est essentiel au fonctionnement correct de Samba et doit donc toujours être lancé. Il peut également être configuré en tant que serveur WINS (abréviation de l'anglais « Windows Internet Name Server »), afin de pouvoir fournir les noms NetBIOS sur Internet. WINS est une technique développée par Microsoft pour résoudre le problème du nommage des postes de travail sur Internet. Un serveur WINS est donc à NetBIOS un peu ce qu'un serveur DNS est à TCP/IP. Le démon nmbd est installé classiquement dans le répertoire `/usr/bin/` ou dans le répertoire `/usr/sbin/`. Vous devrez le lancer avec l'option `-D`, faute de quoi il ne démarrera pas en tant que démon :

```
/usr/sbin/nmbd -D
```

Le deuxième démon fourni par Samba est « smbd ». Celui-ci gère effectivement le protocole SMB, et est donc en charge d'effectuer la gestion des utilisateurs et des partages. Il doit être lancé également avec l'option `-D` si l'on veut qu'il démarre en tant que démon. Ce démon est normalement installé au même endroit que nmbd :

```
/usr/sbin/smbd -D
```

Vous pourrez éventuellement faire en sorte que ces démons soient lancés automatiquement en modifiant les scripts de démarrage de votre système. Une fois ces deux démons lancés, votre machine Linux se comportera exactement comme une machine Windows serveur classique. Ainsi, les ressources partagées par ce serveur seront accessibles par les clients SMB, comme les postes Windows par exemple.

Malheureusement, la plupart des clients SMB ne gèrent pas les notions avancées des systèmes de fichiers Unix. Le protocole SMB ne supporte d'ailleurs qu'une partie infime de ces fonctionnalités. Samba utilise donc une politique bien précise pour effectuer la correspondance entre les fichiers Unix et les fichiers au sens des clients SMB.

La première difficulté à résoudre concerne la gestion des droits d'accès aux fichiers des partages. Sous Unix, chaque fichier et chaque répertoire dispose de droits d'accès, et seuls les utilisateurs autorisés peuvent les manipuler. Dans le monde Windows, il n'y a quasiment pas de notion d'utilisateur, et en général pas de gestion de la sécurité en général. Seul Windows NT dispose de ces fonctionnalités, à condition qu'il utilise des systèmes de fichiers NTFS. Cependant, les informations de sécurité de Windows NT sont toutes perdues lorsque l'on accède aux fichiers d'un partage : lourd héritage que celui du DOS !

En fait, la gestion de la sécurité se fait de manière globale, pour tous les fichiers d'un partage. Il existe principalement deux modes de contrôle d'accès aux partages SMB :

- le contrôle d'accès au niveau ressource, qui se fait à partir d'un mot de passe unique pour la ressource, sans tenir compte de l'utilisateur qui cherche à y accéder ;
- le contrôle d'accès au niveau utilisateur, qui nécessite de fournir un nom d'utilisateur et un mot de passe.

Le contrôle d'accès au niveau utilisateur nécessite de se trouver dans un domaine, et non dans un simple Workgroup. Il faut donc disposer d'un serveur de domaine afin de réaliser l'authentification des utilisateurs qui cherchent à accéder à ces ressources. Pour des raisons commerciales, Microsoft s'est arrangé pour que seul Windows NT puisse servir de contrôleur de domaine, ce qui fait qu'il est impossible d'utiliser le contrôle d'accès au niveau utilisateur sur un réseau ne disposant que de machines sous Windows 95 ou Windows 98 (heureusement, Samba est capable de servir de contrôleur de domaine, cependant, nous ne verrons pas ce type de configuration).

Pour couronner le tout, les partages ne permettent que de fixer les droits de lecture et d'écriture pour celui qui accède aux fichiers du partage. Les droits d'exécution n'ont pas de signification sous Windows, puisque, contrairement aux fichiers Unix, le type des fichiers est déterminé par leur extension. Les notions de sécurité des partages sont donc profondément différentes de celles utilisées par Unix, surtout pour le contrôle d'accès au niveau ressource, puisqu'aucun nom d'utilisateur n'est requis lors

de l'authentification ! Mais ce n'est pas tout. Les fichiers Unix appartiennent tous à un utilisateur et à un groupe d'utilisateurs, chose qui n'a pas de signification pour Windows. De plus, les systèmes de fichiers Unix font la distinction entre les majuscules et les minuscules dans les noms de fichiers, ce que ne fait pas Windows. Et ils disposent de la notion de lien physiques et symboliques. Autant de problèmes qui doivent être gérés par Samba.

> **Note:** Le fait de ne pas disposer des informations concernant les droits des utilisateurs sur les fichiers accédés par SMB n'empêche pas le serveur de fichiers de contrôler les accès effectués par le client. Si par exemple, un fichier est marqué comme en lecture seule, aucun client ne peut y écrire, même par l'intermédiaire d'un partage accédé en lecture et écriture.

Pour résoudre tous ces problèmes, Samba dispose d'un grand nombre d'options, qui peuvent être utilisées dans le fichier de configuration `smb.conf`. Ces options permettent d'indiquer la manière dont l'authentification doit être faite, les droits avec lesquels les fichiers sont accédés ou créés, les utilisateurs au nom desquels les fichiers sont créés, et le comportement à suivre lorsqu'un client essaie de suivre un lien symbolique. Seules les principales options gérées par Samba seront décrites ici. Les options avancées seront laissées de côté, car Samba propose un nombre incroyable de fonctionnalités, et les traiter toutes dépasserait le cadre de ce document. Vous pourrez trouver des informations complémentaires en cas de nécessité dans la documentation de Samba, ou bien dans l'excellent livre « Using Samba » de Robert Eckstein, publié aux éditions O'Reilly. Une version en ligne de ce livre est fournie avec les sources de Samba et complète ainsi la documentation officielle de ce programme.

La plupart des distributions fournissent un fichier de configuration `/etc/smb.conf` contenant la plupart des options par défaut, plus quelques exemples de configuration. Par exemple, les répertoires des utilisateurs sont souvent exportés par défaut. Vous pourrez bien entendu modifier ce fichier de configuration. Il est donc peut-être nécessaire de donner quelques précisions.

Le fichier `smb.conf` est structuré en sections permettant de fixer les paramètres de différentes fonctionnalités du logiciel. Chaque section est introduite par son nom, donné entre crochets. Certaines sections sont spécifiques à Samba et d'autres permettent de définir les partages que vous désirez créer. Les paramètres définis dans les sections se présentent sous la forme de couple « paramètre = valeur », où `paramètre` est le nom d'un des paramètres possibles dans cette section, et `valeur` est la valeur que ce paramètre peut prendre.

Parmi les sections propres à Samba, on compte la section « [global] ». Cette section est particulièrement importante, car elle fixe les paramètres globaux de Samba, ainsi que les valeurs par défaut des paramètres des autres sections. La section `[global]` suivante peut servir d'exemple :

```
[global]
# Définit le nom NetBIOS du serveur :
    netbios name = Linux
    server string = Linux fait la Samba (version %v)

# Définit le nom du groupe de travail :
    workgroup = monrezo

# Fixe les règles de gestion de la sécurité :
```

```
    security = user
    encrypt passwords = yes
    smb passwd file = /etc/samba/passwd

# Définit la correspondance entre les noms d'utilisateurs Windows
# et les noms Unix :
    username map = /etc/samba/usermap

# Définit le compte Unix à utiliser pour invité :
    guest ok = no
    guest account = nobody

# Définit les paramètres réseau :
    socket options = TCP_NODELAY
    interfaces = eth0
    bind interfaces only = yes
```

Cette section commence par définir le nom NetBIOS du serveur et sa description. Par défaut, le nom utilisé par Samba est le nom de domaine Unix de la machine, mais vous pouvez changer ce nom comme bon vous semble. Vous avez dû remarquer que dans la description de la machine (mot-clé `server string`), la version de Samba est récupérée avec `%v`. En fait, Samba défini un certain nombre de variables qui sont remplacées dynamiquement lors de la connexion à un partage. Ainsi, `%v` représente la version de Samba utilisée, `%u` représente le nom de l'utilisateur qui se connecte, etc… Toutes ces variables sont décrites dans la page de man du fichier `smb.conf`.

Le nom du groupe de travail dont fait partie la machine est ensuite introduit avec le mot-clé `workgroup`. Dans le cas présent, il s'agit du groupe de travail « monrezo ».

Viennent ensuite les options de sécurité. Le mot-clé `security` permet de définir le mode de contrôle d'accès aux partages mis à disposition aux clients par le serveur Samba. La valeur `user` indique que ce contrôle d'accès se faut au niveau utilisateur, ce qui est le plus cohérent sous Unix. Chaque utilisateur doit donc fournir son nom et son mot de passe pour accéder à ces partages. Samba peut également gérer le contrôle d'accès au niveau ressource, si l'on utilise la valeur `share`. Dans ce cas, Samba essaiera de retrouver l'utilisateur qui cherche à se connecter à partir de son mot de passe. Il faut donc ajouter dans ce cas une ligne telle que celle-ci dans la section globale ou dans l'une des sections définissant un partage :

```
    username = utilisateurs
```

où `utilisateurs` représente la liste des utilisateurs que Samba utilisera pour tester le mot de passe fourni. Bien entendu, si un nom d'utilisateur est fourni par le client pour accéder à cette ressource, ce nom est utilisé directement. Samba mémorisera également ce nom pour les demandes d'accès ultérieures à la ressource partagée. Cet algorithme n'est pas très sûr, car un utilisateur peut se connecter par hasard au nom d'un autre. Aussi n'est-il pas recommandé d'utiliser le contrôle d'accès au niveau ressources.

Jusqu'à Windows 95 et Windows NT4 Service Pack 2 compris, les mots de passe fournis par les clients étaient transférés en clair sur le réseau. Ceci n'étant pas sûr, Microsoft a décidé d'adopter une nouvelle

technique, dans laquelle les mots de passe à transférer sont encryptés par une clef fournie par le serveur. Ainsi, une personne mal intentionnée écoutant les paquets transférés sur le réseau ne pourrait pas capter ces mots de passe. Samba est capable de gérer les deux cas de configuration, à l'aide de l'option `encrypt password`. Cependant, si vous décidez d'activer l'encryptage des mots de passe, vous devrez également créer un fichier de mots de passe pour les clients SMB. L'emplacement de ce fichier est indiqué avec l'option `smb passwd file`.

Le format du fichier de mot de passe de Samba ressemble fortement à celui du fichier de mot de passe Unix `/etc/passwd`. Vous pourrez trouver dans la documentation de Samba une description détaillée de ce fichier. Cependant, la chose la plus importante, c'est bien sûr de pouvoir ajouter des entrées pour chaque utilisateur dans ce fichier. Cette opération se fait avec l'utilitaire **smbpasswd**. Exécutée sous le compte root, la commande suivante permet d'ajouter un utilisateur et de définir son mot de passe :

```
smbpasswd -a utilisateur
```

où `utilisateur` est le nom de l'utilisateur dont on veut créer un nouveau mot de passe. L'utilisateur peut bien entendu changer son mot de passe avec **smbpasswd**.

Il est possible que les noms des utilisateurs Windows ne soient pas identiques à leur login sur la machine serveur de fichiers. En fait, il est même possible que plusieurs utilisateurs utilisent un même compte, créé uniquement pour les partages SMB. Samba fournit donc la possibilité d'utiliser un fichier de correspondance entre les noms des utilisateurs avec l'option `username map`. Le format des fichiers de correspondance est très simple, puisqu'il est constitué de lignes définissant chacune une association entre un nom de login Unix et un nom d'utilisateur Windows. Ces lignes utilisent la syntaxe suivante :

```
login = utilisateur
```

où `login` est le nom de login de l'utilisateur sur la machine Unix, et `utilisateur` est le nom qu'il utilisera pour accéder aux partages.

L'option `guest ok` permet d'autoriser les connexions sans mot de passe sous un compte spécial, que l'on nomme le compte invité. Pour cela, il suffit de lui affecter la valeur `yes`. Cette option peut être fixée à `no` dans la section de configuration globale, et redéfinie dans les sections spécifiques de certains partages. Dans tous les cas, les connexions qui s'effectuent en tant qu'invité doivent utiliser un compte utilisateur Unix classique. Ce compte peut être défini à l'aide de l'option `guest account`, en général, l'utilisateur nobody utilisé par NFS est le plus approprié.

Enfin, il est possible de définir des paramètres réseau dans la section de configuration globale. Ces paramètres réseau permettent de fixer des options de sécurité complémentaires et des options d'optimisation. L'option `interfaces` donne la liste des interfaces à partir desquelles les demandes de connexion des clients seront acceptées. Ce type de contrôle est activé lorsque le paramètre `bind interfaces only` prend la valeur `yes`. Ces options conviennent parfaitement pour un réseau local de particulier. L'option `socket options` quant à elle permet de définir les paramètres de fonctionnement des communications TCP. L'option `TCP_NODELAY` indique au système d'envoyer les paquets TCP dès que Samba le demande, sans chercher à en regrouper plusieurs pour optimiser la taille des paquets. Cette option accélère significativement le fonctionnement des programmes tels que Samba, qui utilisent beaucoup de petits paquets pour envoyer des requêtes ou des réponses. Sans cette op-

tion, le système chercherait à regrouper ces paquets, même si Samba n'en a pas d'autres à envoyer, et ralentirait ainsi inutilement les transferts de données.

Chaque partage fourni par le serveur SMB dispose également d'une section, dont le nom est le nom utilisé pour ce partage. Ces sections sont beaucoup plus simples que la section global, comme le montre l'exemple suivant :

```
[Données]
# Donne le chemin sur le répertoire utilisé par ce partage :
    path = /usr/share/samba/données

# Description du partage :
    comment = Disque de données

# Nom de volume Windows :
    volume = SMB-LNX

# Options d'accès :
    writeable = yes
    guest ok = yes
```

Comme vous pouvez le constater, l'option `path` permet de définir le répertoire utilisé pour stocker les fichiers du partage. Ce répertoire est un répertoire du serveur, et le chemin doit donc obligatoirement être un chemin Unix valide. En particulier, il faut tenir compte ici de la casse des noms de fichiers. L'option `comment` donne la description du partage, telle qu'elle apparaîtrait dans le voisinage réseau de Windows. Le nom de volume quant à lui est celui qui sera donné si un client Windows attache une lettre de lecteur à ce partage. Il est spécifié à l'aide de l'option `volume`. Enfin, les options d'accès utilisées dans cet exemple permettent d'autoriser l'écriture dans les fichiers de ce partage, et d'autoriser les accès en tant qu'invité (c'est à dire sans mot de passe).

Samba dispose d'une fonctionnalité intéressante afin de définir automatiquement un partage pour chacun des répertoires personnels des utilisateurs déclarés dans le serveur. Lorsqu'un client cherche à accéder à un partage qui n'est pas défini par une section qui lui est propre, Samba tente de le localiser un répertoire personnel d'utilisateur portant ce nom. S'il le trouve, il utilisera ce répertoire pour effectuer le partage automatiquement. Les paramètres de ce partage automatique sont définis dans la section `[homes]`, dont vous trouverez un exemple ci-dessous :

```
[homes]
    comment = Répertoires personnels
    browsable = no
    writable = yes
```

L'utilisation de l'option `browsable = no` ici sert simplement à indiquer qu'il ne doit pas apparaître de partage nommé `homes`. En effet, ce qui est désiré ici, c'est d'exposer les noms des répertoires personnels des utilisateurs, pas le partage `homes` lui-même. Les autres options sont les mêmes que pour les partages normaux.

La section `[homes]` peut poser quelques problèmes, puisqu'elle partage également le compte root. Ceci n'est certainement pas le comportement désiré. Il est même fortement recommandé d'interdire les connexions pour les utilisateurs privilégiés du système. Ceci peut être réalisé en utilisant l'option `invalid users` :

```
    invalid users = root bin daemon adm sync shutdown \
        halt mail news uucp operator gopher
```

En fait, il est conseillé de placer cette option dans la section `[global]`, afin d'interdire les connexions de ces utilisateurs sur tous les partages.

Enfin, Samba fournit une fonctionnalité comparable à la section `[homes]` pour les imprimantes partagées. Il est en effet capable d'analyser le fichier de configuration `/etc/printcap` pour déterminer la liste des imprimantes installées, et les partager pour les clients Windows. Ces imprimantes apparaîtront donc comme des imprimantes partagées classiques. Pour cela, il suffit d'ajouter les lignes suivantes dans la section `[global]` du fichier `smb.conf` :

```
[global]
# Définit le type d'impression utilisé (BSD sous Linux) :
    printing = bsd

# Lit la définition des imprimantes installées dans /etc/printcap
# (type de commande BSD uniquement) :
    printcap name = /etc/printcap
    load printers = yes
```

La première ligne indique à Samba le type de commande à utiliser pour réaliser une impression. Sous Linux, les commandes BSD sont utilisées, aussi fixe-t-on à `bsd` l'option `printing`.

Les options concernant toutes les imprimantes que Samba trouvera peuvent être précisées dans la section `[printers]`. Cette section joue donc le même rôle pour les imprimantes partagées que la section `[homes]` joue pour les répertoires personnels des utilisateurs. La section donnée ci-dessous pourra vous servir d'exemple :

```
[printers]
    comment = Imprimantes
    browseable = no
    printable = yes
    writeable = no
    guest ok = no
    path = /tmp
    create mode = 0700
```

L'option `browseable` a ici la même signification que dans la section `[homes]`. On ne désire en effet pas qu'un partage portant le nom `printers` apparaisse dans les voisinages réseau des machines Windows ! En revanche, l'option `printable` permet d'indiquer clairement que ces partages sont

des imprimantes. Il est évident que l'on ne peut pas écrire sur une imprimante, l'option `writeable` est donc fixée à `no`. Enfin, les impressions ne sont autorisées que pour les utilisateurs identifiés, et l'utilisateur invité n'a pas le droit de soumettre un travail d'impression.

Le mécanisme utilisé par Samba pour imprimer un document est le suivant. Lorsqu'un client demande une impression sur une des imprimantes partagées, Samba copie le fichier que le client lui envoie en local. Ce fichier est ensuite communiqué au gestionnaire d'impression Unix, en l'occurrence lpr sous Linux. Celui-ci imprime le fichier et le supprime du disque. Dans la section `[printers]`, l'option `path` permet d'indiquer dans quel répertoire les fichiers temporaires envoyés par les clients doivent être stockés. Il est tout à fait cohérent de les placer dans le répertoire `/tmp` comme dans l'exemple donné ci-dessus. Enfin, il est logique de protéger ces fichiers contre les autres utilisateurs. Ceci est réalisé à l'aide de l'option `create mode`, qui fixe les droits de ces fichiers à 0 pour tout le monde, sauf pour le compte root, à leur création. Ainsi, seul les programmes fonctionnant sous le compte root (donc Samba et la commande **lpr**) peuvent lire, écrire et effacer ces fichiers.

Il est bien entendu possible de définir les imprimantes manuellement, à raison d'une section par imprimante. Ce type de configuration ne sera toutefois pas décrit ici.

Toute modification du fichier de configuration de Samba implique de la signaler aux démons smbd et nmbd. Ceci peut être réalisé avec la méthode habituelle, en leur envoyant un signal `SIGHUP`. Vous devrez donc taper les commande suivantes :

```
killall -HUP smbd
killall -HUP nmbd
```

Comme on peut le voir, le fichier de configuration `smb.conf` n'est pas très compliqué, mais utilise un grand nombre d'options. Heureusement Samba fournit l'outil de configuration « SWAT » (abréviation de l'anglais « Samba Web Administration Tool »), qui permet d'effectuer la configuration de Samba graphiquement. Comme son nom l'indique, SWAT joue le rôle de serveur Web, et peut être utilisé avec n'importe quel navigateur en se connectant sur le port 901 de la machine sur laquelle il tourne.

Pour que ceci puisse fonctionner, il faut ajouter le service réseau « swat » dans votre fichier de configuration `/etc/services` :

```
swat    901/tcp
```

Vous devrez également ajouter une ligne dans votre fichier `/etc/inetd.conf`, afin que le démon inetd puisse lancer SWAT automatiquement lorsqu'un client cherche à l'utiliser sur le réseau. Vous devrez donc ajouter la ligne suivante :

```
swat  stream  tcp  nowait.400  root  /usr/local/samba/bin/swat swat
```

et signaler à inetd la modification de `inetd.conf` à l'aide de la commande :

```
killall -HUP inetd
```

Vous pourrez alors accéder à la configuration graphique de SWAT à l'aide de Netscape, en utilisant simplement l'URL suivante :

```
http ://localhost :901
```

SWAT demandera bien entendu le nom de l'utilisateur root et son mot de passe pour permettre la modification du fichier `smb.conf`. Notez qu'il est fortement déconseillé de laisser la possibilité de réaliser un connexion Web sur le compte root, et que vous ne devriez pas autoriser les connexions non locales sur le port TCP 901. L'utilisation de SWAT ne pose pas vraiment de problème et ne sera donc pas décrite plus en détail ici.

7.7.4. Configuration d'un client SMB

L'utilisation des volumes partagés par l'intermédiaire du protocole SMB sous Linux est très simple. Elle se fait exactement comme pour NFS : il suffit simplement de monter le volume dans un répertoire vide de l'arborescence de votre système de fichiers.

Pour que ceci puisse être réalisable, il faut bien entendu que le noyau supporte le système de fichiers SMB. Ce système de fichiers est considéré par le noyau exactement comme le système de fichiers NFS : il s'agit d'un système de fichiers réseau. Vous pourrez donc activer le support de ce système de fichiers à l'aide de l'option de configuration « `SMB filesystem support (to mount WfW shares etc.)` », que vous trouverez normalement dans le menu « `Network File Systems` » (ce menu est un sous-menu du menu « `Filesystems` » si vous utilisez le programme de configuration en mode texte). La compilation du noyau et son installation seront décrites en détail dans le Chapitre 9.

Lorsque vous aurez configuré et installé ce noyau, vous pourrez monter les volumes partagés par des serveurs de fichiers SMB (qu'il s'agisse de serveurs Linux ou Windows) exactement comme n'importe quel système de fichiers, et accéder à leurs ressources. Malheureusement, la commande `mount` du système ne supporte pas encore complètement le système de fichiers SMB. Elle utilise donc un programme complémentaire nommé `smbmount`, fourni avec la distribution de Samba. Notez bien que ce programme ne fait pas officiellement partie de Samba, il est seulement fourni en tant qu'utilitaire complémentaire pour Linux. Bien qu'il fasse également partie d'un paquetage indépendant, il est recommandé d'utiliser la version fournie avec Samba. Ceci signifie que vous devrez installer Samba sur les postes clients, même si vous ne lancez pas les démons smbd et nmbd. La commande standard **mount** du système sera peut-être modifiée un jour pour intégrer toutes les fonctionnalités de la commande **smbmount**, ce qui rendra inutile l'installation de Samba sur les postes clients.

Quoi qu'il en soit, la syntaxe de la commande **smbmount** est très simple :

```
smbmount partage répertoire
```

où `partage` est le nom de partage du volume à monter, et `répertoire` est son point de montage (par exemple, `/mnt/`). Le nom de partage utilisé est exactement le même nom que celui utilisé par Windows, à ceci près que les antislash ('\') sont remplacés par des slash ('/'). Ainsi, pour monter dans

`/mnt/` le partage `Données` du serveur de fichiers `SMBSERVER`, vous devrez utiliser la commande suivante :

```
smbmount //smbserver/données /mnt
```

Remarquez que le protocole SMB ne fait pas la distinction entre les majuscules et les minuscules (tout comme Windows d'une manière générale), et que vous pouvez utiliser indifféremment les majuscules ou les minuscules.

L'utilisation des slash à la place des antislash dans le nom du partage est due au fait que l'antislash est un caractère spécial que shell interprète comme étant la poursuite de la commande courante sur la ligne suivante. Vous pouvez utiliser des antislash si vous le désirez, mais dans ce cas, vous devrez mettre le nom de partage entre guillemets, pour empêcher le shell de les interpréter. Malheureusement, l'antislash est également un caractère d'échappement dans les chaînes de caractères, et est utilisé introduire des caractères spéciaux. Et il est lui-même un caractère spécial, qui doit donc être précédé d'un antislash d'échappement ! Il faut donc doubler tous les antislash, et la commande précédente devient particulièrement longue et peu pratique avec cette syntaxe :

```
smbmount "\\\\smbserver\\données" /mnt
```

Il est très facile de faire des erreurs avec de telles commandes, aussi je ne vous la conseille pas.

En fait, il est possible de faire en sorte que la commande mount classique du système puisse être utilisée pour réaliser les montages de partages SMB à partir de la version 2.0.6. ou plus de Samba. Pour cela, il suffit de créer dans le répertoire `/sbin/` un lien symbolique `mount.smbfs` vers la commande **smbmount**. Lorsque l'on utilisera la commande **mount** avec le type de système de fichiers `smbfs`, celle-ci utilisera ce lien et appellera ainsi automatiquement **smbmount**. Les paramètres fournis à **mount** seront transmis tels quels à smbmount. En particulier, vous pourrez utiliser une commande de ce type :

```
mount -t smbfs partage répertoire
```

où `partage` et `répertoire` sont toujours les noms de partage et de répertoire devant servir de point de montage.

Une fois le partage monté, vous pouvez l'utiliser comme un système de fichiers classique. Lorsque vous aurez fini de l'utiliser, vous pourrez simplement le démonter, avec la commande **smbumount** :

```
smbumount /mnt
```

Il est possible également d'utiliser la commande système classique **umount**, mais cela nécessite de repasser sous le compte root, ou de fixer le bit setuid sur l'exécutable de **umount**, ce qui est un énorme trou de sécurité. La commande **smbumount** quant à elle peut être setuid, car elle vérifie que l'utilisateur ne cherche à démonter que les partages qu'il a lui-même monté.

Le protocole SMB ne prend pas en charge la notion d'utilisateur et de groupe d'utilisateurs des systèmes de fichiers Unix classiques. Par conséquent, le propriétaire et le groupe des fichiers accédés par un partage SMB sont fixés par défaut à l'utilisateur qui a effectué le montage et à son groupe. Ceci est naturel et ne pose pas de problème pour la plupart des utilisateurs, mais peut être gênant lorsque l'on monte un partage en tant que root pour le compte d'un autre utilisateur. En effet, celui-ci n'aura pas les droits d'écriture sur les fichiers du partage. Pour résoudre ce problème, il est possible de préciser le nom de l'utilisateur et son groupe à l'aide des options `uid` et `gid` de la commande **smbmount** :

```
smbmount partage répertoire -o uid=utilisateur,gid=groupe
```

où `utilisateur` et `groupe` sont respectivement les noms ou les numéros de l'utilisateur et du groupe auquel les fichiers devront être attribués. Les paramètres `uid` et `gid` peuvent également être utilisés avec la commande **mount**. Dans ce cas, ils sont passés tels quels à **smbmount**.

Vous l'aurez sans doute remarqué, la commande **smbmount** vous demande de taper un mot de passe pour accéder au partage. Ce mot de passe est le mot de passe attendu par le serveur de fichiers SMB. Pour Windows, il s'agit du mot de passe attribué au partage. Pour les serveurs de fichiers Samba, il s'agit du mot de passe de l'utilisateur au nom duquel se fait le partage, ou du mot de passe SMB enregistré dans le fichier de mots de passe `smbpasswd` de Samba. Dans tous les cas, ce mot de passe est demandé, même s'il est vide. La commande **smbmount** peut accepter une option `password`, qui permet de préciser ce mot de passe. Cependant, il est fortement déconseillé de l'utiliser, et ce pour deux raisons. Premièrement, le mot de passe apparaît en clair lorsqu'il est saisi, et la ligne de commande peut être vue par n'importe quel utilisateur avec la commande **ps**. Ceci pose donc un problème de sécurité évident. Deuxièmement, l'option `password` n'est pas reconnue par la commande **mount**, et ne peut donc être utilisée qu'avec **smbmount**. C'est pour cela qu'une autre solution a été proposée, bien qu'encore imparfaite : définir le nom d'utilisateur et le mot de passe dans la variable d'environnement USER. Pour cela, il suffit d'utiliser la commande suivante :

```
export USER=utilisateur%secret
```

où `utilisateur` est le nom d'utilisateur classique, et `secret` son mot de passe sur le partage à monter. Cette technique n'est pas non plus très sûre, et n'est en aucun cas pratique. Aussi est-il recommandé de toujours utiliser la commande **smbmount** directement, et de ne taper le mot de passe que lorsque celle-ci le demande.

Chapitre 8. Notions de compilation et de sources

Ce chapitre présente les notions de base de compilation et de fichiers sources de programmes. Il est certain que le lecteur de ce document n'a pas forcément l'intention de programmer sous Linux, cependant, il est nécessaire de connaître les notions qui vont être décrites ici. En effet, il n'est pas rare, voire il est même courant, d'avoir à recompiler une application lorsqu'on travaille avec Linux. Ceci n'est pas étonnant, quand on sait que toute bonne installation passe par le recompilation du noyau de Linux ! La raison de cet état de fait provient sans nul doute du fait que les licences GNU et BSD imposent de fournir les fichiers sources aux utilisateurs d'une part, et que le langage C et Unix sont historiquement fortement liés. Nous allons commencer par donner un peu de vocabulaire. Nous verrons ensuite comment installer (et compiler !) le compilateur C/C++ de GNU.

8.1. Vocabulaire

Le *langage naturel* est le langage que les êtres humains utilisent pour communiquer, soit oralement, soit par écrit. Le langage naturel est très riche : il utilise des constructions grammaticales et syntaxiques complexes, et il dispose d'un vocabulaire très étendu. Il permet donc d'exprimer la plupart des idées humaines, et c'est bien sa fonction. En revanche, il n'est pas rigoureux, dans le sens où il laisse la porte à un grand nombre d'ambiguïtés. Ce qui n'est pas dit est souvent sous-entendu, et c'est donc pratiquement entre les mots que se trouve le sens du discours. Il n'est ainsi pas rare de pouvoir interpréter une même phrase différemment, selon le contexte socioculturel, géographique ou temporel dans lequel elle est dite. Les jeux de mots utilisent cette caractéristique à merveille. La conséquence est que l'on ne peut pas utiliser facilement le langage naturel pour décrire rigoureusement, voire mathématiquement, quelque chose.

Un *langage formel* est, au contraire, un langage restreint, qui ne dispose que de très peu de constructions syntaxiques et d'un vocabulaire très limité. La caractéristique principale des langages formels est qu'ils sont rigoureux, une expression ou une phrase donnée peut être validée selon les règles de syntaxe qui doivent être respectées. Tout texte écrit dans un langage formel est donc soit *valide*, soit *invalide*.

En définissant une association entre les constructions d'un langage formel et un jeux de concepts limité, il est possible de donner une sémantique à un langage formel. Cette sémantique donne la possibilité de qualifier de « *vraie* » ou de « *fausse* » les assertions écrites dans le langage formel (on notera qu'une expression peut être valide du point de vue de la syntaxe mais sémantiquement fausse).

Un langage formel permet donc d'exprimer avec précision, sans aucune ambiguïté possible, une idée basée sur les concepts qu'il utilise. Les notations mathématiques constituent un langage formel par excellence.

> **Note:** Par exemple, l'expression mathématique « `x=1` » est toujours valide, c'est une simple équation. En revanche, elle n'est pas toujours vraie, cela dépend de la valeur de `x`. En particulier, si x représente 3, on a « `3 = 1` », ce qui est valide, mais faux. Notez bien la différence.

Un *langage de programmation* est un langage formel qui permet de définir les tâches qu'un ordinateur doit effectuer, et de décrire les objets informatiques sur lesquels il doit travailler. Un langage de programmation est donc un code, et tout programme doit être écrit dans un tel langage. Pratiquement, les langages de programmation sont des langages très simples, disposant de constructions du type « `si ... alors ...` » ou « `pour chaque ... fait ...` ». Les programmes étant écrits avec de tels langages, il est clair qu'un programme ne fait que ce qui a été écrit : ni plus, ni moins. Il faut donc tout dire à l'ordinateur quand on écrit un programme, ce qui en général est relativement fatigant et compliqué. C'est le prix à payer pour la rigueur de l'informatique : l'ordinateur ne se trompe jamais, parce qu'il ne fait qu'obéir aux ordres donnés dans un langage formel (donc précis). Celui qui se trompe, c'est en général le programmeur, et assez couramment l'utilisateur.

> **Note:** Notez qu'un programme est un texte valide s'il vérifie la grammaire du langage formel dans lequel il est écrit. Ceci ne l'empêchera pas de faire n'importe quoi si on l'exécute. Un programme « *vrai* » est donc un programme syntaxiquement correctement écrit et qui en plus fait ce que l'on désire qu'il fasse. Il n'y en a pas beaucoup...

Le C est un langage de programmation relativement simple, qui se trouve sur tous les types d'ordinateurs et de systèmes d'exploitation. Les programmes sont plus difficile à écrire en C que dans d'autres langages de programmation, mais ils sont plus performants. Le C est très utilisé pour écrire les systèmes d'exploitation : le noyau de Linux lui-même est écrit en C. Le C++ est un langage plus évolué, qui est dérivé du C. Il est beaucoup plus riche, et il permet d'écrire des programmes et de les faire évoluer plus facilement. Bien que plus lents que ceux écrits en C, les programmes écrits en C++ sont toujours très performants par rapport à ceux écrits dans d'autres langages.

Les programmes sont en général écrits dans un certain nombre de fichiers. Ces fichiers sont appelés les *fichiers sources*, du fait qu'ils sont à l'origine du programme. Le texte de ces fichiers est appelé le *code source*, ou plus simplement le *code*.

Il va de soi que le code source est écrit dans un langage de programmation, qui n'est pas compréhensible tel quel par le matériel de l'ordinateur. Pour exécuter un programme à partir de son code source, il n'y a que deux solutions :

- disposer d'un autre programme capable de lire le code source et de l'exécuter directement ;
- disposer d'un autre programme capable de traduire le code source en langage binaire, directement compréhensible par le matériel.

Les programmes de la première catégorie sont appelés des *interpréteurs*, car ils traduisent le code source à la volée, lors de l'exécution du programme. Les programmes interprétés sont fatalement relativement lents, puisque l'interpréteur doit analyser en permanence les fichiers sources pour les exécuter. En revanche, les programmes de la deuxième catégorie sont appelés *compilateurs*. Les programmes compilés sont beaucoup plus rapides à l'exécution, puisqu'ils ne sont traduit qu'une seule fois en langage machine et s'adressent directement au matériel.

> **Note:** Si vous avez bien suivi, on se trouve face au problème de l'½uf et de la poule. En effet, les interpréteur et les compilateurs sont eux-même des programmes, écrits dans un langage informatique. Quel est donc le premier compilateur ou interpréteur ? Ce problème a effectivement dû

> être résolu au début de l'informatique, de la manière la plus simple : les premiers programmeurs entraient directement le langage machine en binaire dans les ordinateurs (via les câblages ou les cartes perforées...). Quand on sait le travail que cela représentait, et le nombre d'erreurs que cela a pu générer, on comprend facilement pourquoi les compilateurs font partie des tous premiers programmes qui ont été développés...

Dans les systèmes Unix, les deux types de programmes existent. Les programmes interprétés constituent ce que l'on appelle des *scripts*. Le système utilise le shell pour les interpréter. La plupart des opérations d'administration du système utilisent des scripts, car il est toujours plus facile d'écrire et de tester un script que d'écrire un programme en C. Les programmes compilés sont notamment le noyau lui-même, le shell, les commandes de base et les applications. Ce sont eux qui en fin de compte effectuent le travail, et ils sont souvent appelés à partir de scripts. Nous avons déjà vu des exemples de scripts lors de la configuration du système. Pour l'instant, nous allons nous intéresser au langage C et au processus de compilation.

La *compilation* est l'opération qui consiste à lire un fichier source et à le traduire dans le langage binaire du processeur. Ce langage est absolument incompréhensible par les êtres humains, cependant, il existe un langage de programmation qui permet de coder directement le binaire : il s'agit de l'*assembleur*.

En général, le processus de compilation génère un fichier binaire pour chaque fichier source. Ces fichiers binaires sont nommés fichiers *objets*, et porte de ce fait l'extension « .o » (ou « .obj » dans les systèmes Microsoft). Comme un programme peut être constitué de plusieurs fichiers sources, il faut regrouper les fichiers objets pour générer le fichier exécutable du programme. Cette opération s'appelle l'*édition de liens*, et elle est réalisée par un programme nommé le *linker* (*éditeur de liens* en français). Ce programme regarde dans tous les fichiers objets les références partielles aux autres fichiers objets, et pour chaque « lien » ainsi trouvé, il complète les informations nécessaires pour en faire une référence complète. Par exemple, un fichier source peut très bien utiliser une fonctionnalité d'un autre fichier source. Comme cette fonctionnalité n'est pas définie dans le fichier source courant, une référence partielle est créée dans le fichier objet lors de sa compilation, mais il faudra tout de même la terminer en indiquant exactement comment accéder à la fonctionnalité externe. C'est le travail de l'éditeur de liens, lorsqu'il regroupe ces deux fichiers.

Certains fichiers objets sont nécessaires pour tous les programmes. Ce sont notamment les fichiers objets définissant les fonctions de base, et les fonctions permettant d'accéder au système d'exploitation. Ces fichiers objets ont donc été regroupés dans des *librairies*, que l'on peut ainsi utiliser directement lors de la phase d'édition de liens. Les fichiers objets nécessaires sont alors lus dans la librairie et ajoutés au programme en cours d'édition de liens. Les librairies portent souvent l'extension « .a » (ou « .lib » dans les systèmes Microsoft).

Malheureusement, cette solution souffre de la duplication du code contenu dans les librairies dans tous les programmes, d'où une perte de place considérable. De plus, la mise à jour d'une librairie nécessite de refaire l'édition de liens de tous les programmes qui l'utilisent, ce qui n'est pas réalisable en pratique. C'est pour cela que les librairies partagées ont été crées : une librairie partagée n'est pas incluse dans les fichiers des exécutables qui l'utilisent, mais reste dans un répertoire bien défini du système de fichiers. De cette manière, le code de la librairie est effectivement partagé, et la mise à jour de la librairie peut se faire sans avoir à toucher tous les programmes qui l'utilisent. Le

problème est cependant que l'édition de lien reste incomplète, parce que les références aux objets des librairies partagées sont toujours externes. Il existe donc un programme spécial, l'*éditeur de liens dynamique* (« ld », pour « Link Dynamically »), qui résout les dernières références incomplètes lors du chargement de chaque programme. Les librairies partagées portent l'extension « .so » (pour « Shared Object »), ou « .dll » dans les systèmes Microsoft (pour « Dynamic Link Library »).

Évidemment, le chargement des programmes est un peu plus lent avec les librairies partagées, puisqu'il faut réaliser l'édition de lien dynamique lors de leur lancement. Cependant, ce processus a été optimisé, et les formats de fichiers binaires utilisés contiennent toutes les informationsprécalculées pour faciliter la tâche de l'éditeur de lien. Ainsi, la différence de performances est devenue à peine décelable.

Classiquement, les fichiers sources C et C++ se divisent en deux grande catégories :

- les *fichiers d'en-tête*, qui contiennent les déclarations des symboles du programme ;
- les fichiers C (ou C++), qui contiennent leurs définitions.

Le fait de faire cette distinction entre la déclaration (qui consiste à dire : « telle chose existe ») et la définition (qui consiste à décrire la chose précédemment déclarée) permet de faire en sorte que l'on peut utiliser les fichiers objets sans avoir les fichiers sources. En effet, la déclaration permet de réaliser les références partielles dans les programmes, tandis que les fichiers objets contiennent bien entendu la définition binaire de ce qui a été déclaré.

Pour compiler un programme, on n'a donc réellement besoin que de trois types de fichiers :

- les fichiers de déclaration du système et des librairies de base ;
- les fichiers des librairies de base eux-mêmes ;
- les fichiers source (de déclaration et de définition) du programme à compiler.

En C et C++, les fichiers sources de déclaration portent l'extension « .h » (plus rarement « .hpp » pour les fichiers de déclaration C++), et les fichiers de définition portent l'extension « .c » pour les programmes écrits en C et « .C » (ou « .cpp », ou encore « .c++ ») pour les programmes écrits en C++.

Il va de soi que la réalisation d'un programme peut nécessiter la création d'un grand nombre de fichiers, tous plus ou moins dépendants les uns des autres. Pour pouvoir s'y retrouver plus facilement, les programmeurs utilisent un programme spécial : **make**. Ce programme est capable de réaliser toutes les opérations nécessaires à la création des fichiers exécutables d'un programme à partir de ses fichiers sources. Pour cela, il utilise un fichier contenant la définition des opérations à réaliser, ainsi que la description des dépendances entre les différents fichiers sources. Ce fichier est appelé le *fichier makefile*. Grâce à make, l'utilisateur de base que vous êtes va pouvoir compiler la plupart des programmes sans avoir la moindre notion de programmation.

En général, la compilation d'un programme passe par les étapes suivantes :

- installation des fichiers sources du programme ;

- configuration du programme pour l'environnement courant ;
- appel à **make** pour la compilation ;
- appel à **make** pour l'installation.

La première étape est élémentaire et va de soi.

La deuxième étape se fait souvent en appelant un script dans le répertoire d'installation des fichiers sources. Ce script se nomme souvent **configure**, et peut être appelé avec la ligne de commande suivantes :

```
./configure
```

à partir du répertoire où se trouvent les fichiers sources. Ce script effectue tous les tests sur le système et l'environnement de développement, et génère le fichier makefile. Le programme **configure** est en général fourni avec les logiciels GNU. Il permet de déterminer l'environnement logiciel et système et de générer le fichier makefile et des fichiers d'en-têtes contenant la définition de quelques options du programme. Dans quelques cas particuliers, vous aurez à utiliser des options de configure afin de préciser certains paramètres. L'option `--host` permet d'indiquer le type de la machine cible, dans le cas où **configure** ne parvient pas à le déterminer automatiquement :

```
--host=machine
```

où `machine` est la description de votre machine. Pour les PC fonctionnant sous Linux, cette description est de la forme « `ix86-pc-linux-gnu` », où le '`x`' représente le numéro de la génération du processeur du PC. Par exemple, pour un Pentium ou un AMD K6, la description sera « `i586-pc-linux-gnu` ». L'option `--enable-shared` permet d'utiliser les librairies dynamiques pour la compilation des librairies. Ceci qui procure un gain de place évident. Enfin, l'option `--prefix` permet de préciser le répertoire de base dans lequel le programme sera installé a priori. Cette option s'utilise de la manière suivante :

```
--prefix=répertoire
```

où `répertoire` est le répertoire de base de l'installation. La connaissance de ce répertoire est utile aux autres programmes pour localiser les composants qui vont être installés. Dans la plupart des cas, la valeur par défaut du préfixe convient, mais il est parfois nécessaire de le modifier. C'est en particulier le cas lorsque vous désirez mettre à jour un programme déjà installé et que votre distribution n'utilise pas le répertoire par défaut. Dans ce cas, il convient d'indiquer le répertoire de base dans lequel ce programme a été installé, afin de le remplacer au lieu de le dupliquer d'une part, et afin que les autres programmes puissent trouver la nouvelle version à la place de l'ancienne d'autre part.

La troisième étape est très simple aussi. Il suffit de taper la commande suivante :

```
make
```

toujours dans le répertoire où se trouvent les fichiers sources.

Enfin, la dernière étape se fait en tapant la commande suivante :

```
make install
```

Bien entendu, ces différentes étapes varient parfois avec le logiciel à installer. Cependant, il existe quasiment toujours un fichier texte indiquant comment effectuer ces opérations dans le répertoire des fichiers sources. Ce fichier porte souvent le nom de « `readme` », ou quelque chose de similaire.

Avec les notions que vous venez de voir, vous devriez pouvoir compiler quasiment tous les programmes que vous pourrez récupérer sous la forme de fichiers sources. Un dernier détail cependant : la compilation est une opération très gourmande en ressources. Ceci signifie qu'elle peut consommer beaucoup de temps processeur, de mémoire, de disque et de temps. Pour des gros programmes, il n'est pas rare de passer jusqu'à une heure de compilation, même sur une machine récente. Notez également que le facteur limitant dans les compilations sont souvent la rapidité du disque dur, et la quantité de la mémoire vive disponible.

8.2. Compilation de GCC

La plupart des distributions sont fournies avec le compilateur GNU C/C++. Cependant, la version fournie est souvent la version 2.7.2., qui est assez âgée, pour ne pas dire complètement obsolète. La raison de cet état de fait est que la compilation du noyau de Linux pose quelques problèmes avec les versions ultérieures de GCC. Les distributeurs s'assurent ainsi de ne pas avoir de problèmes relatifs aux dernières versions de GCC en n'installant que la version 2.7.2. par défaut.

Cependant, il n'est pas raisonnable de continuer à utiliser cette version, car la version courante, à savoir GCC 2.95.2. (novembre 1999), est nettement meilleure sur tout les points de vue. Elle optimise mieux le code et elle comprend bien mieux le C++ que la version 2.7.2. Pour tout dire, il s'est écoulé deux versions majeures entre la 2.7.2. et la 2.95.2. !

Sachez que l'on n'installe pas GCC pour le plaisir de le faire. C'est une étape quasiment nécessaire. Cependant, il est légitime que vous vous posiez quelques questions. La première est : « Quel est l'intérêt d'installer la dernière version de GCC pour celui qui ne veut pas programmer ? ». Eh bien tout simplement de ne pas avoir de problèmes avec des programmes récents dont seuls les sources sont disponibles. Il est probable qu'un jour ou un autre vous trouviez des sources qui ne compilent tout simplement pas avec GCC 2.7.2. Dans ce cas, vous serez obligé d'installer la dernière version de GCC, alors autant le faire tout de suite. Une autre question peut survenir, surtout si vous avez entendu parler de rumeurs peu rassurantes à propos du noyau de Linux : « N'y a-t-il pas un risque si l'on compile le noyau de Linux avec une version récente de GCC ? ». La réponse est non, pour deux raisons. Premièrement, il y avait bien un problème avec les versions 2.0. du noyau, mais ce problème a été résolu à partir de la version 2.2. Deuxièmement, les difficultés que l'on rencontre lors de la compilation du noyau avec les nouvelles versions de GCC proviennent de quelques fonctions mal écrites dans le noyau, qui ne compilent pas correctement avec une option d'optimisation qui a été ajoutée récemment à GCC. Par conséquent, il suffit de désactiver cette option pour ne plus avoir de problèmes. Les fichiers makefile des noyaux récents le font automatiquement. Ce problème est décrit plus en détail dans le chapitre traitant de la compilation du noyau.

Certaines distributions fournissent également la version 1.1.2. du système de compilation expérimental de GNU (« EGCS » en anglais, abréviation de « Experimental GNU Compilation System »). Cet outil est une version de développement de GCC, et a deux inconvénients majeurs :

- primo, elle n'est pas officielle ;
- secondo, elle est elle aussi obsolète, puisque la version 2.95.2. de GCC est ultérieure à EGCS 1.1.2 (EGCS et GCC ont été fusionné en 1999, après la diffusion des nouvelles distributions de Linux avec le noyau 2.2.).

En clair, si la version 1.1.2. d'EGCS est actuellement fournie avec votre distribution, c'est parce que celle-ci a été diffusée avant la sortie de GCC 2.95.2. Il faut donc choisir ce dernier de préférence.

> **Note:** Il est également assez courant de voir distribuées des versions 2.96 et 2.97 de GCC. Ces versions n'ont rien d'officiel, et sont des versions de développement qui doivent normalement conduire à la version finale 3.0 de GCC. Il ne faut pas les utiliser, elles ne sont pas stables, et aucun programme ne requiert actuellement les fonctionnalités du futur GCC 3.0. Celui-ci sera disponible avant que cet état de fait ne change.

8.2.1. Prérequis

À moins que vous ne réussissiez à mettre la main sur une version de GCC déjà compilée pour Linux, vous ne disposerez que des sources. Les sources de GCC peuvent être récupérées sur Internet sur le site de GCC (http ://gcc.gnu.org). Ces sources sont fournies sous la forme d'une archive au format tar-gzip. Le nom de cette archive est « `gcc-2.95.2.tar.gz` ». Il est inutile, avec cette version, de récupérer la librairie standard C++, car elle est fournie avec les sources de GCC. Il va de soi que si, au moment où vous lisez ce document, la version 3.0 de GCC est diffusée, il faut récupérer les sources de cette version.

Si votre système utilise la librairie C GNU 2.2., vous devrez également récupérer un correctif afin de modifier les sources de GCC pour les en-têtes de cette librairie. Ce correctif peut être récupéré sur Internet (http ://clisp.cons.org/~haible/gcc-glibc-2.2-compat.diff). Vous pouvez déterminer la version de la librairie C installée dans votre système à l'aide de la commande **ldd —version**.

Un autre prérequis est bien entendu que vous disposiez également d'un compilateur C sur votre système. Ce compilateur peut très bien être une ancienne version de GCC, mais il est impératif de disposez d'un compilateur C correct. Le code de GCC a été écrit pour pouvoir être compilé de manière minimale avec la plupart des compilateurs C existants. Il faut également que ce compilateur porte le nom de « **cc** » ou « **gcc** » et soit dans l'un des répertoires de la variable d'environnement PATH, ou que la variable CC ait été définie avec le chemin absolu du compilateur.

8.2.2. Installation des sources

Lorsque vous aurez installé le compilateur C natif de votre système, vous devrez décompresser les sources de GCC. Ceci doit se faire avec la commande suivante :

```
tar xvfz archive
```

où `archive` est le nom de l'archive compressée contenant ces sources. Dans notre cas, ce doit être `gcc-2.95.2.tar.gz`. Cette commande a pour conséquence de créer l'arborescence des répertoires des sources de GCC dans le répertoire où elle a été exécutée.

Vous devrez ensuite modifier les sources ainsi extraits si vous utilisez la version 2.2. de la librairie C GNU. Pour cela, il faut se placer dans le sous-répertoire `libio/` du répertoire des sources de GCC, et taper la commande suivante :

```
patch -p1 < gcc-glibc-2.2-compat.diff
```

Une fois cette opération effectuée, il est conseillé de créer un autre répertoire, ailleurs que dans le répertoire des sources de GCC, dans lequel la compilation aura lieu. Dans toute la suite, nous supposerons que le répertoire des sources se nomme `srcdir`, et que le répertoire de compilation se nomme `objdir`.

8.2.3. Configuration

L'étape suivante est d'aller dans le répertoire de compilation et de lancer le programme de configuration de GCC dans ce répertoire. Ceci peut être réalisé avec les commandes suivantes :

```
cd objdir
srcdir/configure [options]
```

Comme on l'a vu ci-dessus, le programme de configuration **configure** peut recevoir des options en ligne de commande. Il est fortement conseillé de passer l'option `--enable-shared`, qui permet d'utiliser les librairies partagées. Il est également recommandé de fournir l'option `--enable-threads`, qui permet d'activer le support du multithreading pour le langage Objective C (c'est un langage objet dérivé du C, géré par GCC en marge du C++ et qui n'est pas très courant, mais suffisamment utilisé pour qu'on ne puisse pas s'en passer). Ces fonctionnalités étant disponibles sous Linux, autant les utiliser.

Normalement, le programme de configuration détecte le type de machine et le système utilisé. Cependant, cette détection peut échouer si les types de machines indiqués pour les différents composants du système ne sont pas identiques. Dans ce cas, il vous faudra spécifier la machine hôte à l'aide de l'option `--host`. La ligne de commande pour la configuration sera alors la suivante :

```
srcdir/configure --enable-shared --enable-threads --host=hôte
```

où `hôte` est de la forme « `ix86-pc-linux-gnu` ».

Enfin, si votre distribution utilise le compilateur GCC comme compilateur de base du système, et que vous désirez remplacer la version de votre distribution par la version que vous allez compiler, il faut changer le répertoire d'installation de GCC. Normalement, GCC s'installe dans le répertoire

`/usr/local/` ce qui fait qu'il ne remplace pas la version de votre distribution. Vous devez donc spécifier les répertoires de base pour l'installation. Pour cela, il suffit d'ajouter les options suivantes :

```
--prefix=/usr --with-local-prefix=/usr \
    --with-gxx-include-dir=/usr/include/g++
```

à la ligne de commande de **configure**.

8.2.4. Compilation

La compilation de GCC peut ensuite être réalisée. Autant vous prévenir tout de suite : c'est une opération très longue, qui de plus demande beaucoup d'espace disque. Il faut au moins prévoir 150 Mo d'espace disque, et une bonne heure sur une machine à 250 Mhz. Ceci est dû à la technique employée par GCC lors de la compilation. La plupart des compilateurs C ne sont pas capables de compiler GCC avec toutes ses fonctionnalités. C'est pour cela que la compilation se passe en trois étapes :

- une version allégée de GCC est compilée avec le compilateur natif dans une première passe ;
- cette version est utilisée pour compiler la version complète de GCC ;
- la version complète est utilisée pour se recompiler, afin de tester les différences entre les deux versions complètes.

Ainsi, GCC se compile lui-même !

Ces trois opérations peuvent être exécutées à l'aide d'une seule commande :

```
make bootstrap
```

Cette commande doit être exécutée à partir du répertoire de compilation `objdir`.

8.2.5. Installation de GCC

Lorsque la compilation s'est terminée, vous pouvez installer GCC. Il est recommandé de supprimer le compilateur que vous avec utilisé pour compiler GCC, sauf si, bien entendu, il s'agissait déjà de GCC. En effet, il n'est pas nécessaire de le conserver, puisque vous utiliserez désormais GCC. L'installation de GCC est, encore une fois, très simple :

```
make install
```

Cette commande installe GCC ainsi que la librairie standard C++.

Une fois GCC installé, il se peut que vous ayez des problèmes avec un certain fichier d'en-tête de la librairie C de votre distribution, si celle-ci est assez ancienne. Ce fichier d'en-tête ne se compile

pas correctement avec GCC 2.95.2. (il utilise une construction incorrecte, qui était acceptée avec les anciennes versions de GCC mais que GCC 2.95.2. refuse à présent). Si vous vous trouvez dans cette situation, il vous faudra modifier ce fichier d'en-tête, qui se trouve normalement dans le répertoire `/usr/include/`. Il porte le nom `selectbits.h` et contient la définition de quelques macros pour le compilateur GCC et pour les autres compilateurs, dont `__FD_ZERO`. La modification du fichier est très simple : il suffit de supprimer le test sur le compilateur (ce test contient le mot « `__GNUC__` ») et toute la partie de code correspondante (cette partie s'étend jusqu'au mot « `#else` »). De cette manière, il ne reste plus que le code générique, qui compile sur toutes les machines. Il faut également supprimer la ligne « `#endif` » qui allait de pair avec les lignes « `#if` » et « `#else` » qui ont été supprimées. Une autre solution est simplement de récupérer une version de la librairie C GNU supérieure ou égale à la version 2.1.2., de la compiler et de l'installer. Toutefois, cette opération est assez délicate, et nécessite de bien connaître le système Linux. Elle n'est donc vraiment pas recommandée pour les débutants. Cependant, la manière de procéder est décrite dans l'Annexe A de ce document pour ceux qui sont vraiment téméraires.

Chapitre 9. Compilation du noyau de Linux

La compilation du noyau est une spécificité des systèmes libres, qui n'est possible que parce que l'on dispose des sources du noyau. Cependant, même pour certains Unix commerciaux, il est possible d'effectuer une édition de lien, les modules du noyau étant fournis sous la forme de fichiers objets. La compilation ou l'édition de lien du noyau est une opération technique qui peut surprendre un habitué des systèmes fermés que sont par exemple Windows 95 ou OS/2. Cependant, elle permet d'obtenir un noyau très petit, optimisé pour la machine sur laquelle il tourne, et donc à la fois économe en mémoire et performant. Il est donc recommandé d'effectuer cette compilation : pourquoi conserver un monstre capable de gérer des périphériques qui ne sont pas et ne seront jamais installé sur votre système ?

La compilation du noyau de Linux nécessite de disposer des dernières sources du noyau (version 2.4.0. au 5/01/2001) et d'un compilateur. Il est évident que le compilateur idéal est le compilateur GNU C/C++ GCC. On utilisera une version 2.7.2. ou supérieure de GCC. On prendra garde au fait que les versions de GCC supérieures à 2.95.2. ne peuvent pas être utilisées pour compiler les noyaux de version 2.2.10. ou moins. En effet, les compilateurs récents utilisent des optimisations qui sont incompatibles avec certaines constructions utilisées par ces noyaux (en réalité, ces constructions sont fausses, mais ne posent de problèmes qu'avec les compilateurs récents). Je supposerais dans la suite de ce document que vous disposez de la dernière version du noyau, à savoir la version 2.4.0.

La compilation du noyau n'est pas très difficile, cependant, elle nécessite de répondre correctement aux questions de configuration. Les erreurs peuvent être multiples, et seront fatales. Il est donc fortement conseillé de disposer d'une disquette de démarrage afin de réparer le système en cas d'erreur. Par ailleurs, il faut toujours conserver le dernier noyau utilisable en sauvegarde dans le répertoire `/boot/`. Il faut également ajouter une entrée spécifiant ce noyau dans le programme de démarrage (**lilo**), afin de pouvoir sélectionner l'ancien noyau en cas d'erreur. Ces opérations seront également décrites en détail plus loin.

La compilation du noyau se passe en quatre étapes :

- installation des fichiers sources ;
- réponse aux questions de configuration ;
- compilation proprement dite ;
- installation du nouveau noyau.

9.1. Installation des sources de Linux

Les sources du noyau peuvent être trouvées sur le site kernel.org (http ://www.kernel.org). Il est possible de récupérer les sources complètes, sous la forme d'une archive compressée d'environ 17 Mo. Toutefois, si l'on dispose déjà d'une version complète des fichiers sources, il est envisageable de ne télécharger que les patches.

Il est recommandé d'installer les sources du noyau dans un répertoire `/usr/src/linux<version>/` et d'éditer un lien symbolique vers ce répertoire sous le nom `/usr/src/linux/`. Ceci permet de

conserver plusieurs jeux de sources de versions différentes, et de travailler sur la version courante sous `/usr/src/linux/`. Les commandes suivantes permettront d'extraire les sources dans le répertoire dédié au sources de Linux. Elles supposent qu'il existe déjà un lien symbolique `/usr/src/linux/` vers le répertoire des fichiers sources actuels de Linux :

```
cd /usr/src
rm linux
mkdir linux-2.4.0
ln -s linux-2.4.0 linux
tar xvfz linux-2.4.0.tar.gz
```

Si l'on dispose déjà d'une version complète des fichiers sources, et que l'on désire appliquer un patch, il faut décompresser le fichier de patch avec la commande suivante :

```
gunzip fichier.gz
```

où `fichier.gz` représente le fichier de patch compressé (en supposant qu'il ait été compressé à l'aide de gzip). L'application du patch se fait de la manière suivante :

```
patch -p0 < fichier
```

Cette commande doit être lancée à partir du répertoire `/usr/src/`. Elle suppose que les fichiers sources de Linux pourront être trouvées dans le répertoire `./linux/`. Dans cette ligne de commande, `fichier` représente le nom du fichier de patch précédemment décompressé, et l'option `-p0` indique au programme **patch** d'utiliser les noms de répertoires relatifs au répertoire courant (à savoir `./linux/`). Si les sources sont installées dans un autre répertoire, il faudra modifier l'option `-px` passée en paramètre au programme **patch**. Consultez la page de manuel `patch` pour plus de détails sur cette option.

Il est recommandé de télécharger au moins une fois les sources complètes du noyau, et de ne pas utiliser les sources fournies avec la distribution que vous utilisez. En effet, certaines distributions modifient les sources et on ne peut donc pas leur appliquer les patches standard.

9.2. Lancement du programme de configuration

La configuration du noyau peut se faire à l'ancienne avec la commande suivante :

```
make config
```

Cette commande pose une série de questions auxquelles il faut pouvoir répondre correctement du premier coup. On n'a pas le droit à l'erreur ici, faute de quoi il faut tout reprendre à zéro.

Il est nettement plus convivial d'utiliser la version Tcl/Tk sous X11. Cette version donne l'accès aux différentes options sans un ordre quelconque, et ne présente que les options réalisables étant données celles déjà fixées. Cette méthode est évidemment la méthode conseillée. Pour l'utiliser, il suffit de taper la commande suivante :

```
make xconfig
```

Si l'on ne dispose pas encore de X11, ce qui est probablement le cas lors d'une première installation, on peut utiliser la version texte avec menu en tapant la commande suivante :

```
make menuconfig
```

Quelle que soit la méthode utilisée, il faut répondre par 'Y' (pour « Yes »), 'N' (pour « No ») ou 'M' (pour « Module ») lorsque c'est possible. 'Y' et 'M' incluent la fonctionnalité courante dans le noyau, 'N' la supprime. 'M' permet d'utiliser la fonctionnalité en tant que module du noyau. En général, l'utilisation des modules permet d'alléger le noyau car les fonctionnalités sont chargées et déchargées dynamiquement. Cependant, les fonctionnalités nécessaires au démarrage de Linux, comme les gestionnaires de disques et systèmes de fichiers par exemple, ne doivent en aucun cas être placées dans des modules, car alors le système ne pourrait pas démarrer.

9.3. Choix des options de configuration

Les questions posées sont récapitulées ci-dessous. Les réponses recommandées ont été choisies pour correspondre à la plupart des cas courants. Il ne s'agit pas des options recommandées pour installer un serveur ou pour un machine contenant des périphériques exotiques (carte vidéo, port infrarouge, etc…). Avec ce jeu d'options, un PC standard est supposé démarrer sans poser de problème. Vous devrez cependant certainement les adapter selon vos besoins. Je tiens également à préciser que certaines de ces options m'ont laissé dubitatif, étant dans l'incapacité absolue de les comprendre et de les tester. Ces options sont en général les options concernant des fonctionnalités avancées ou des périphériques rarement utilisés. Afin de se prémunir contre tout mauvais choix, il est impératif de suivre pas à pas les recommandations données dans les paragraphes concernant l'installation du nouveau noyau et suivants. Ces recommandations stipulent simplement qu'il faut conserver le noyau actuel tant que l'on n'est pas sûr que le noyau que l'on installe fonctionne parfaitement.

9.3.1. Menu « Code maturity level options »

Ce menu donne accès aux fonctionnalités en cours de développement. Il est déconseillé de répondre 'Y', sauf si des drivers particuliers sont nécessaires. Si l'on répond 'N', les questions de configuration concernant ces fonctionnalités ne seront pas posées.

9.3.2. Menu « Loadable module support »

L'option « `Enable loadable module support` » permet l'emploi des modules par le système. Il est recommandé de répondre 'Y' à cette question.

L'option « `Set version information on all module symbols` » permet d'enregistrer des informations de version dans les modules du noyau. Cette fonctionnalité autorise l'emploi des modules des versions précédentes, par l'intermédiaire du programme modprobe. En particulier, ceci est

nécessaire si vous désirez utiliser des modules non fournis dans les sources du noyau (par exemple des drivers commerciaux). Par ailleurs, si vous désactivez cette fonctionnalité, il faudra recompiler tous les modules à chaque fois que vous changerez de noyau. Il est donc recommandé de répondre par 'Y' à cette question.

L'option « Kernel module loader » donne la possibilité au noyau de charger lui-même les modules du noyau. Cette fonctionnalité est nécessaire pour la bonne marche du système, aussi est-il recommandé de répondre par 'Y' à cette question.

9.3.3. Menu « Processor type and features »

L'option « Processor family » vous permet de spécifier le type de processeur sur lequel le noyau fonctionnera. Choisissez le processeur dont vous disposez.

L'option « Toshiba Laptop support » permet d'activer la gestion d'énergie pour les portables de marque Toshiba. Vous pouvez répondre 'Y' à cette question si vous désirez utiliser votre noyau sur un portable Toshiba. La réponse recommandée est 'N'.

L'option « /dev/cpu/microcode - Intel P6 CPU microcode support » permet d'activer la reprogrammation du microcode des processeurs Intel postérieurs aux Pentium Pro. À l'aide de cette fonctionnalité, vous pourrez mettre à jour le microcode de votre processeur (cette mise à jour n'est pas permanente, le microcode doit être rechargé à chaque démarrage de la machine). Notez que cette option impose d'utiliser le système de fichiers virtuel /dev/ du noyau. Il est donc nécessaire d'activer l'option « /dev filesystem support (EXPERIMENTAL) » du menu « File systems ». En général, il n'est pas nécessaire de mettre à jour le microcode de son processeur, à moins que celui-ci ne contienne un bug incontournable ou que vous n'ayez à optimiser spécialement votre machine pour une utilisation particulière. La réponse recommandée est donc 'N'.

L'option « /dev/cpu/*/msr - Model-specific register support » permet d'activer la gestion du registre MSR sur les processeurs x86. Il existe un registre de ce type pour chaque processeur, que l'on trouvera dans les sous-répertoires portant comme nom les numéros de chaque processeur. La réponse recommandée est 'N'.

L'option « /dev/cpu/*/cpuid - CPU information support » permet d'obtenir les informations fournies par l'instruction CPUID pour chaque processeur de la machine. Il existe un registre de ce type pour chaque processeur, que l'on trouvera dans les sous-répertoires portant comme nom les numéros de chaque processeur. La réponse recommandée est 'N'.

L'option « High Memory Support » vous permet d'indiquer le mode de gestion de la mémoire que le noyau utilisera. Pour la plupart des gens, dont les machines disposent de moins d'un Go de mémoire, la réponse recommandée est « off ». Si votre machine dispose de plus d'un Go de mémoire, mais moins de quatre Go, vous pouvez changer le mode d'adressage du noyau pour gérer la totalité de votre mémoire, en choisissant l'option « 4GB ». Les processeurs x86 ne permettent pas d'adresser plus de 4Go de mémoire simultanément en raison de leur bus d'adresses 32 bits. Cependant, ils peuvent gérer jusqu'à 64Go de mémoire par tranches de 4Go, à l'aide d'une extension spécifique aux processeurs de type Pentium Pro. Si vous disposez d'une telle machine, vous devez choisir l'option « 64GB ». La réponse recommandée est « off ».

L'option « `Math emulation` » permet d'activer l'émulateur d'unité de calcul en virgule flottante, pour les processeurs qui n'en disposent pas (386 et 486 SX). Si vous avez un ordinateur récent, choisissez '`N`'.

L'option « `MTRR (Memory Type Range Register) support` » permet d'activer le support des plages mémoires du processeur. Celui-ci peut permettre l'accélération des transferts de donnés dans les plages mémoires des périphériques, en particulier pour les cartes graphiques. Cette fonctionnalité n'est disponible que pour les processeurs de type Pentium Pro et postérieurs. Si vous avez un ordinateur récent, choisissez '`Y`'.

L'option « `Symetric multi-processing support` » permet d'activer le support des cartes mères multi-processeurs. La plupart des gens n'en disposant pas, vous pouvez répondre '`N`' à cette question.

L'option « `APIC and IO-APIC support on uniprocessors` » permet d'activer la gestion des contrôleurs d'interruption programmables avancés. Ces contrôleurs sont utilisés sur les machines multi-processeurs, mais certaines cartes mères monoprocesseurs les utilisent. Si c'est le cas de votre carte mère, vous pouvez répondre par '`Y`' à cette question.

9.3.4. Menu « General setup »

L'option « `Networking support` » permet d'activer le support réseau. Les systèmes Unix étant profondément basés sur les réseaux, il faut répondre par '`Y`'.

L'option « SGI Visual Workstation support » permet de générer un noyau pour les stations de travail Silicon Graphics de type 320 ou 540. Ces stations utilisent des processeurs x86, mais sont basés sur des composants systèmes différents. À moins que vous n'ayez une telle machine, répondez par '`N`'.

L'option « `PCI support` » permet d'activer le support des ordinateurs à base de PCI. Si vous avez un ordinateur récent, répondez par '`Y`'.

L'option « `PCI access mode` » permet de sélectionner le composant qui initialisera les bus PCI. Si vous avez un ordinateur récent, avec un BIOS récent, choisissez « `BIOS` ». Sinon, choisissez « `Direct` ». Le choix « `Any` » permet de demander à Linux d'essayer l'initialisation par le BIOS, et de faire le travail lui-même si ce dernier est défaillant. Vous pouvez donc toujours choisir l'option « `Any` ».

L'option « `PCI device name database` » permet d'inclure dans le noyau une liste de noms de périphériques PCI. Cette table est utilisée pour fournir des noms humainement lisibles pour les périphériques PCI dans le système de fichiers virtuels `/proc/`. Elle accroît sensiblement la taille du noyau, mais ne consomme pas de mémoire supplémentaire une fois que la phase d'amorçage terminée. Il est donc recommandé d'activer cette fonctionnalité, à moins que vous ne cherchiez à faire un système embarqué ou une disquette d'amorçage. La réponse recommandée est donc '`Y`'.

L'option « `EISA support` » permet d'activer la gestion des bus EISA. Ce bus a désormais complètement été remplacé par le bus PCI, aussi devriez-vous répondre par '`N`', à moins que vous ne disposiez d'une machine très ancienne.

L'option « `MCA support` » permet d'activer la gestion des bus MCA (pour les PS/2 d'IBM). À moins que vous n'ayez un PS/2, répondez par '`N`'.

L'option « `Support for hot-pluggable devices` » permet d'activer la gestion des périphériques connectables à chaud (c'est à dire pendant que le système fonctionne). Parmi ces périphériques, on recontre couramment les cartes PCMCIA des portables, mais également les périphériques USB. Cette option est nécessaire à l'utilisation des cartes PCMCIA sur les portables. Elle vous donnera l'accès à l'option « `PCMCIA/CardBus support` », qui permet d'activer la gestion des cartes PCMCIA 32 bits (ceci n'est pas nécessaire pour utiliser les cartes PCMCIA 16 bits). Cette option est facultative pour l'utilisation des périphériques USB, toutefois, ces périphériques ne seront pas configurés automatiquement lorsque vous les connecterez à chaud si vous ne l'activez pas. Pour que la configuration des périphériques USB connectés à chaud fonctionne, vous devez également activer la gestion des modules du noyau, ainsi que l'option « `Kernel module loader` ». Lorsqu'un périphérique sera connecté, le noyau appelera alors le programme de configuration `/sbin/hotplug` pour charger le gestionnaire de périphérique approprié. La réponse recommandée est '`Y`'.

L'option « `System V IPC` » permet d'activer la gestion des communications inter-processus compatibles System V. C'est quasiment un standard sur tous les systèmes Unix, il faut donc répondre par '`Y`'.

L'option « `BSD Process Accounting` » permet d'activer le monitoring des applications utilisé sur les système BSD. Ce monitoring peut être utilisé par quelques applications, aussi est-il recommandé de répondre par '`Y`'.

L'option « `Sysctl support` » permet de modifier dynamiquement certains paramètres du noyau sans recompilation ni redémarrage. Il est recommandé de répondre par '`Y`' à cette question.

L'option « `Kernel core (/proc/kcore) format` » permet de choisir le format de fichier binaire pour le fichier `kcore` du système de fichiers virtuels `/proc/`. Ce fichier contient la représentation de la mémoire vive du système telle qu'elle est vue par le noyau, et peut être utilisé pour le débogage du noyau. Les différents formats de fichiers binaires sont « A.OUT », un ancien format de fichier désormais obsolète, ou « ELF » (abréviation de l'anglais « Executable and Linking Format »), le format de fichier actuel. La réponse recommandée est « `ELF` ».

L'option « `Kernel support for a.out binaries` » permet d'utiliser les programmes dont le format de fichier binaire est le format « a.out ». Bien que ce format de fichier binaire soit obsolète, il se peut que vous rencontriez de vielles applications qui n'ont pas été recompilées et qui l'utilisent encore. Il est donc recommandé d'activer la gestion de ce format : répondez par '`Y`'.

L'option « `Kernel support for ELF binaries` » permet d'utiliser les programmes dont le format de fichier binaire est le format ELF. Ce format de fichier étant devenu un standard, il faut répondre par '`Y`' à cette question.

L'option « `Kernel support for MISC binaries` » permet d'activer la gestion de formats de fichiers binaires enregistrables dans le noyau. On peut alors utiliser des chargeurs spécifiques directement au niveau du noyau, et ainsi utiliser ces fichiers binaires directement en ligne de commande. Il est recommandé de répondre par '`Y`' à cette question.

L'option « `Power Management support` » permet d'activer la gestion d'énergie sur votre machine. Il existe actuellement deux protocoles de gestion d'énergie sur les PCI : le protocole APM (abréviation de l'anglais « Advanced Power Management », relativement ancien mais fiable, et le protocole ACPI (abréviation de l'anglais « Advanced Configuration and Power Interface »), plus récent mais

pouvant poser quelques problèmes de stabilité du système. Ces deux protocoles peuvent être paramétrés respectivement avec les deux jeux d'options suivantes. La réponse recommandée est 'Y'.

L'option « ACPI support » permet d'activer la gestion du protocole de gestion d'énergie ACPI. Ce protocole permet au système d'exploitation de contrôler finement la consommation d'énergie du système, mais nécessite que votre carte mère le gère ainsi que tous les périphériques connectés dessus. Si ce n'est pas le cas, les périphériques qui seront mis en veille ne se réinitialiseront pas correctement lors du réveil de l'ordinateur, ce qui peut provoquer des plantages inédits. La réponse recommandée est donc 'N', à moins que vous ne soyez sûr que votre matériel est 100% compatible ACPI. Notes que si vous choisissez d'activer le protocole de gestion d'énergie ACPI, celui-ci sera utilisé en priorité par rapport au protocole APM. Il sera donc inutile d'activer la gestion de ce dernier. Les options suivantes permettent d'activer certaines fonctionnalités du protocole ACPI.

L'option « `Advanced Power Management BIOS support` » permet d'activer la gestion d'énergie APM par l'intermédiaire du BIOS. Cette méthode de gestion d'énergie est beaucoup plus sûre que l'ACPI, car elle est plus ancienne et en général parfaitement supporté par le matériel actuel. Il faut activer le support de la gestion d'énergie par APM si l'on veut que l'ordinateur s'éteigne tout seul lors de l'arrêt du système. Il est donc recommandé de répondre par 'Y' à cette question. Notez toutefois que cette fonctionnalité n'est disponible que pour les ordinateurs disposant d'un boîtier au format ATX. On remarquera que ce n'est pas la gestion d'énergie APM qui gère l'arrêt des disques durs et la veille des moniteurs « Green », il est possible d'avoir ces fonctionnalités même si l'on n'a pas activé la gestion d'énergie APM. On notera également que l'horloge logiciel de Linux prend du retard lorsque l'ordinateur passe en veille. Les options suivantes dépendent fortement de la configuration APM des machines. Les options de ce menu ne seront pas décrites plus en détails ici, car elles sont trop spécifiques à chaque modèle de machine. Pour la plupart des gens, il faut répondre 'N' à toutes ces questions.

9.3.5. Menu « Memory Technology Devices (MTD) »

Ce menu vous permet d'activer la gestion des puces de mémoire Flash ou autre périphérique de mémoire persistante. Ce genre de périphérique est généralement utilisé pour réaliser des systèmes de fichiers sur les systèmes embarqués, qui ne disposent pas nécessairement de disques magnétiques. L'option « `Memory Technology Device (MTD) support` » vous donnera accès aux choix de drivers pour les différents matériels supportés. La réponse recommandée est 'N'.

9.3.6. Menu « Parallel port support »

L'option « `Parallel port support` » permet d'activer la gestion du port parallèle au niveau du noyau. Il est recommandé de le faire sous la forme de module, car le port parallèle n'est pas souvent utilisé. De toutes façons, il est recommandé d'activer cette fonctionnalité, en répondant par 'Y' ou par 'M'. Quelle que soit la réponse donnée, on pourra ainsi utiliser le port parallèle pour plusieurs fonctionnalités différentes (imprimante, lecteurs externes, etc...). Répondez par 'Y' ou par 'M'.

L'option « `PC-style hardware` » permet d'indiquer au noyau que le port parallèle est compatible PC. Si vous compilez le noyau pour un PC ou un Alpha, répondez par 'M' à cette question.

L'option « `Use FIFO/DMA if available` » permet de demander au noyau d'utiliser une interruption et un canal DMA pour accéder au port parallèle, si le chipset de la carte mère le supporte. Ceci permet généralement d'accélérer les entrées / sorties sur le port parallèle, en évitant au noyau d'utiliser un mécanisme de consultation périodique de l'état du port afin de savoir s'il est capable d'accepter des données ou si des données doivent être lues. Notez que par défaut, le noyau n'utilisera aucune ligne d'interruption pour accéder au port parallèle, et ce même si vous avez activé cette option. Pour changer le comportement par défaut, vous devrez passer le paramètre `parport` au noyau lors de son démarrage, suivi de l'adresse du port et de la ligne d'interruption à utiliser, séparés par une virgule. Si vous avez demandé la compilation du driver de port parallèle sous forme de module, vous devrez spécifier ces options dans le fichier `/etc/modules.conf`. La réponse recommandée est '`Y`'.

L'option « `SuperIO chipset support (EXPERIMENTAL)` » permet d'activer la gestion des chipset SuperIO présents sur certaines cartes mères. La réponse recommandée est '`N`'.

L'option « `Support for PCMCIA management for PC-styles ports` » permet d'utiliser les outils PCMCIA pour les périphériques sur port parallèles. La réponse recommandée est '`N`'.

L'option « `Sparc hardware (EXPERIMENTAL)` » permet de prendre en charge les ports parallèles des vieilles stations Sparc de Sun. Les nouvelles stations Ultra Sparc utilisent à présent des ports parallèles de PC, et la peu de gens ont à utiliser cette option. La réponse recommandée est '`N`'.

L'option « `Support foreign hardware` » permet d'activer la gestion de matériels exotiques pour le port parallèle. Il est recommandé de répondre '`N`' à cette question.

L'option « `IEEE 1284 transfer modes` » permet de paramétrer le driver du port parallèle pour utiliser les communications bidirectionnelles du port. Cette option est utile si l'on utiliser une imprimante capable d'indiquer son état à l'ordinateur. Notez que pour cette fonctionnalité soit disponible, vous devez également paramétrer votre BIOS pour que le port parallèle soit en mode EPP ou ECP. La réponse recommandée est '`Y`'.

9.3.7. Menu « Plug and Play configuration »

L'option « `Plug and Play support` » permet d'activer la gestion du plug and play au niveau du noyau. La réponse recommandée est '`Y`'.

L'option « `ISA Plug and Play support` » permet d'activer les fonctionnalités Plug and Play pour les périphériques ISA. Il est ainsi possible d'éviter d'avoir à utiliser les outils isapnp et pnpdump, et d'éviter la compilation des drivers de ces périphériques en modules. La configuration de ces périphériques ISA Plug and Play en est donc grandement simplifiée. Il est donc recommandé de répondre par '`Y`' à cette question. Toutefois, si vous désirez affecter manuellement les lignes d'interruptions, canaux DMA et ports d'entrée/sortie pour vos cartes, vous devez répondre par '`N`'.

9.3.8. Menu « Block devices »

L'option « `Normal PC floppy disk support` » permet d'activer la gestion des lecteurs de disquettes sous Linux. Vous pouvez répondre par '`Y`' à cette question.

L'option « `PS/2 ESDI hard disk support` » permet d'activer la gestion des disques ESDI sur les ordinateurs de type PS/2. Cette option n'est disponible que si vous avez activé la gestion du bus MCA à l'aide de l'option « `MCA support` » du menu « General setup ». La réponse recommandée est '`N`'.

L'option « `XT hard disk support` » permet d'activer la gestion des disques durs XT. Ce sont de très vieux disques durs, que plus personne n'utilise. Répondez par '`N`' à cette question.

L'option « `Parallel port IDE device support` » permet d'activer la gestion des périphériques IDE connectés sur port parallèles. Il est recommandé de placer ce pilote en module, car les lecteurs sur port parallèles ne sont pas toujours connectés à l'ordinateur. Répondez par '`M`' à cette question.

L'option « `Parallel port IDE disks` » permet d'activer le support des disques IDE connectés sur port parallèles. Si vous disposez d'un tel disque, répondez par '`M`' à cette question, et choisissez le protocole de communication sur port parallèle correspondant dans l'une des options suivantes. La réponse recommandée est '`N`'.

L'option « `Parallel port ATAPI CD-ROMs` » permet d'activer la gestion des CD-ROM ATAPI connectés sur port parallèle. Si vous disposez d'un tel lecteur, répondez par '`M`' à cette question, et choisissez le protocole de communication sur port parallèle correspondant dans l'une des options suivantes. La réponse recommandée est '`N`'.

L'option « `Parallel port ATAPI disks` » permet d'activer la gestion des disques ATAPI connectés sur port parallèle. Si vous disposez d'un tel disque, répondez par '`M`' à cette question, et choisissez le protocole de communication sur port parallèle correspondant dans l'une des options suivantes. La réponse recommandée est '`N`'.

L'option « `Parallel port ATAPI tapes` » permet d'activer la gestion des lecteurs de bandes connectés sur port parallèle. Si vous disposez d'un tel lecteur, répondez par '`M`' à cette question, et choisissez le protocole de communication sur port parallèle correspondant dans l'une des options suivantes. La réponse recommandée est '`N`'.

L'option « `Parallel port generic ATAPI devices` » permet la gestion de périphériques ATAPI non standards connectés au port parallèle. Les logiciels utilisateurs peuvent envoyer des commandes ATAPI spécifiques à ces périphériques par l'intermédiaire de ce driver. En particulier, les graveurs de CD connectés sur ports parallèles utilisent cette fonctionnalité. Si vous disposez d'un graveur de CD sur port parallèle, répondez par '`M`' à cette question, et choisissez le protocole de communication sur port parallèle correspondant dans l'une des options suivantes. La réponse recommandée est '`N`'.

Les options qui suivent permettent de choisir les protocoles de communication sur port parallèle adaptés à votre matériel. Vous devez en choisir au moins un si vous comptez utiliser un périphérique IDE connecté sur le port parallèle. Les réponses recommandées sont '`N`' pour les protocoles que votre matériel ne comprend pas.

L'option « `Compaq SMART2 support` » permet d'activer la gestion des cartes contrôleurs Smart Array de Compaq. À moins que vous ne disposiez d'une telle carte, la réponse recommandée est '`N`'.

L'option « `Compaq CISS Array support` » permet d'activer la gestion des cartes contrôleurs CISS de Compaq. À moins que vous ne disposiez d'une telle carte, la réponse recommandée est '`N`'.

L'option « `Mylex DAC960/DAC1100 PCI RAID Controler support` » permet d'activer la gestion des contrôleurs RAID Mylex. La réponse recommandée est '`N`'.

L'option « `Loopback device support` » permet d'utiliser des fichiers classiques pour y placer un système de fichier. Ceci est essentiellement utilisé pour créer des images de CD et les tester avant de les graver. Si vous disposez d'un graveur de CD-ROM, il est recommandé de répondre '`Y`' à cette question. Sinon, répondez '`M`', pour vous réserver la possibilité d'utiliser des systèmes de fichiers stockés dans des fichiers classiques.

L'option « `Network block device support` » permet d'accéder aux fichiers spéciaux de périphériques de type bloc d'un ordinateur distant par l'intermédiaire du réseau. Cette fonctionnalité est peu utilisée, la réponse recommandée est donc '`N`'.

L'option « `RAM disk support` » permet d'activer la gestion des disques virtuels. Ces disques sont souvent utilisés pour créer des disquettes de réparation, qui chargent les utilitaires systèmes sur un disque en mémoire. La réponse recommandée est '`N`' pour un système normal, et '`Y`' pour un système destiné à être placé sur une disquette de réparation.

L'option « `Default RAM disk size` » permet de fixer la taille par défaut des disques virtuels. La taille recommandée est de 4096 Ko.

L'option « `Initial RAM disk (initrd) support` » permet d'activer la possibilité de monter le système de fichier racine sur un disque virtuel au démarrage. La réponse recommandée est '`N`' pour un système normal, '`Y`' pour un système destiné à être placé sur une disquette de réparation.

9.3.9. Menu « Multi-device support (RAID and LVM)

L'option « `Multiple devices driver support (RAID and LVM)` » permet d'activer la gestion de la redondance de données RAID et des disques logiques. La technologie RAID permet de stocker de réaliser des agrégats de disques, soit afin de simuler des disques de grande capacité, soit afin de stocker les données de manière redondante sur plusieurs disques afin d'obtenir une sécurité accrue de ces données. La technologie LVM permet quant à elle uniquement de regrouper plusieurs volumes physiques afin de simuler la présence d'un volume logique de très grande capacité. Vous pouvez répondre par '`Y`' à cette question si vous avez besoin de l'un de ces fonctionnalités. Dans le cas contraire, répondez par '`N`'. La réponse recommandée est '`N`'.

L'option « `RAID support` » permet d'activer la prise en charge des technologies RAID au niveau logiciel. Cette option n'est pas nécessaire pour utiliser les technologies RAID avec un matériel spécifique. La réponse recommandée est '`N`'.

L'option « `Linear (append) mode` » permet de concaténer plusieurs partitions pour ne former qu'une seule zone de données plus grande. Cette fonctionnalité n'assure pas la redondance des données. La réponse recommandée est '`N`'.

L'option « `RAID-0 (striping) mode` » permet de répartir les données sur plusieurs partitions de manière équilibrée. Ceci permet de simuler des disques de très grande capacité, et également d'augmenter les performances en minimisant les temps d'accès, si ces partitions sont sur des disques différents. Cette fonctionnalité n'assure cependant pas la redondance des données. La réponse recommandée est '`N`'.

L'option « `RAID-1 (mirroring) mode` » permet de dupliquer les données sur plusieurs disques à la volée. Cette redondance des données permet d'assurer une grande sécurité. La réponse recommandée est '`N`'.

L'option « `RAID-4/RAID-5 mode` » permet d'activer le support logiciel RAID-4 ou RAID-5. Dans le mode de fonctionnement RAID-4, un des disques est utilisé pour contrôler la validité des données sur les autres disques. Dans le mode de fonctionnement RAID-5, ces données de contrôle sont réparties sur tous les disques, pour une capacité toujours diminuée de la capacité de l'un des disques. La réponse recommandée est '`N`'.

L'option « `Boot support` » permet de placer les fichiers de démarrage sur des disques RAID. Le démarrage de Linux demandera alors de passer des paramètres complémentaires lors de l'amorçage soit en ligne de commande, soit avec LILO. Une alternative à cette option est l'option « `Auto Detect support` », qui permet de faire la détection des volumes RAID automatiquement lors du démarrage. La réponse recommandée pour ces deux options est '`N`'.

L'option « `Logical volume manager (LVM) support` » permet d'activer la gestion des volumes virtuels. Avec cette option, vous pourrez faire des agrégats de plusieurs disques, périphériques RAID ou périphériques loopback afin de simuler un périphérique de type bloc de très grande capacité. La réponse recommandée est '`N`'.

L'option « `LVM information in proc filesystem` » permet d'ajouter des informations sur les volumes virtuels LVM dans le système de fichiers virtuel `/proc/`. La réponse recommandée est '`N`'.

9.3.10. Menu « Networking options »

L'option « `Packet socket` » permet d'autoriser la manipulation directe des trames réseau par des applications clientes. L'option recommandée est '`N`'.

L'option « `Packet socket : mmapped IO` » n'est disponible que si l'option précédente a été activée. Elle permet d'utiliser un mécanisme de communication optimisé pour la communication des trames réseau aux applications clientes, basé sur des segments de mémoire partagés (ce qui évite une copie des données transférées). La réponse recommandée est '`Y`'.

L'option « `Kernel/User netlink socket` » permet d'activer un mécanisme de communication interprocessus basé sur les sockets. Quelques applications peuvent utiliser cette interface, et certaines fonctionnalités du noyau utilisent elles-mêmes cette interface pour publier des informations utilisables par les programmes clients. La manière dont ce protocole fonctionne est assez compliquée. Elle utilise des fichiers spéciaux de périphériques placés dans le répertoire `/dev/`. Les fichiers spéciaux utilisés par cette interface de communication disposent tous du même code majeur, qui vaut 36. La nature des informations qui transitent par ces fichiers spéciaux est dépendante de leurs codes mineurs. La réponse recommandée est '`N`'.

L'option « `Routing messages` » permet d'obtenir des informations sur le routage réseau par l'intermédiaire du fichier spécial de code majeur 36 et mineur 0. La réponse recommandée est '`N`'.

L'option « `Netlink device emulation` » est obsolète et n'est plus utilisée. Répondez par '`Y`' à cette question pour assurer la compatibilité avec d'éventuels vieux programmes.

L'option « `Network packet filtering (replace ipchains)` » permet d'activer les fonctions de filtrage des paquets réseau du noyau. Ces fonctions peuvent être utilisées pour plusieurs raisons, les plus courantes étant sans doute la réalisation de Firewall et de partages de connexions à Internet. Pour information, un Firewall est un programme qui filtre les informations en provenance et à destination du réseau, selon des règles de sécurité prédéfinies. Ces règles permettent d'effectuer le filtrage en fonction des adresses et du type des paquets émis et reçus. Les actions qui peuvent être prise sur les paquets ainsi filtrés peuvent être variées, allant de l'élimination du paquet à sa modification ou son transfert vers une autre adresse que celle vers laquelle il devait aller initialement. Cette fonctionnalité permet donc également de réaliser la translation d'adresses des paquets TCP/IP. Vous devez activer cette option si vous désirez réaliser un partage de connexion à Internet ou un Firewall. La réponse recommandée est '`N`'.

L'option « `Network packet filtering debugging` » active la gestion des messages de débogage des fonctionnalités de filtrage des paquets. Ces messages peuvent être très utiles pour le débogage des règles de filtrage des noyaux et du masquerading, aussi la réponse recommandée est-elle '`Y`'.

L'option « `Socket Filtering` » active la fonctionnalité permettant aux programmes utilisateurs d'enregistrer un filtre pour chaque socket. Ce filtre indique au noyau si les paquets correspondants doivent être transmis ou rejetés. Ce type de filtres est très utilisé pour la réalisation de Firewall évolués, ou d'outil d'analyse de trafic du réseau. La réponse recommandée est '`N`'.

L'option « `Unix domain sockets` » permet d'activer les sockets de types Unix. Comme la plupart des programmes Unix utilisent ce paradigme de communication, il faut répondre '`Y`' à cette question.

L'option « `TCP/IP networking` » permet d'activer le protocole de communication réseau TCP/IP. Le système utilisant intensivement ce protocole de communication, il faut répondre '`Y`' à cette question.

L'option « `IP multicasting` » permet d'autoriser l'envoi des données à plusieurs ordinateurs en mode multicast (un paquet pour plusieurs destinations). Cette fonctionnalité permet de réduire le trafic réseau dans certaines applications, mais elle est très peu utilisée. La réponse recommandée est donc '`N`'.

L'option « `IP: advanced router` » permet de configurer le système pour être un routeur (ordinateur qui transfère des informations d'un réseau à un autre). La réponse recommandée est '`N`'.

L'option « `IP: policy routing` » permet d'activer le routage des paquets en fonction des adresses sources en plus des adresses destinations des paquets. La réponse recommandée est '`N`'.

L'option « `IP : use netfilter MARK value as routing key` » permet d'activer le routage en fonction du champ MARK des paquets en plus des adresses destination. Le champ MARK est utilisé par les fonctionnalités de filtrage du noyau, et permet d'identifier certains paquets pour leur faire subir des traitements ultérieurs. Parmi ces traitements, on peut utiliser un routage spécifique, ce que cette option permet de réaliser. La réponse recommandée est '`N`'.

L'option « `IP: fast network address translation` » permet de convertir les adresses sources et destination des paquets qui passent par l'ordinateur selon une méthode prédéterminée. Cette fonctionnalité n'est pas nécessaire pour réaliser le masquerading, aussi la réponse est-elle recommandée '`N`'.

L'option « `IP: equal cost multipath` » permet de choisir une route possible parmi plusieurs routes de manière non déterministe pour un paquet donné. La réponse recommandée est '`N`'.

L'option « `IP: use TOS value as routing key` » permet de prendre en compte le type de service auquel appartient le paquet courant dans la détermination de la route. La réponse recommandée est '`N`'.

L'option « `IP: verbose route monitoring` » permet d'activer les traces du sous-système de routage. La réponse recommandée est '`N`'.

L'option « `IP: large routing tables` » permet d'accroître la taille des tables de routage et d'augmenter ainsi leur rapidité d'exécution pour les grands réseaux. La réponse recommandée est '`N`'.

L'option « `IP: kernel level autoconfiguration` » permet de réaliser la configuration du protocole réseau IP au niveau du noyau, lors de la phase de démarrage. Cette option est utilisée notamment lorsque l'on désire monter le système de fichier root par NFS. La réponse recommandée est '`N`'.

L'option « `IP : BOOTP support` » permet de demander au noyau de déterminer automatiquement l'adresse IP lors du démarrage grâce au protocole « BOOTP ». Cette option n'est valide que lorsque l'option « `IP : kernel level autoconfiguration` » a été activée. La réponse recommandée est '`N`'.

L'option « `IP : RARP support` » permet de demander au noyau de déterminer automatiquement l'adresse IP lors du démarrage grâce au protocole « RARP ». Ce protocole est un protocole plus ancien que le protocole BOOTP, il est en passe de devenir obsolète. Cette option n'est valide que lorsque l'option « `IP : kernel level configuration` » a été activée. La réponse recommandée est '`N`'.

L'option « `IP: tunneling` » permet d'activer l'encapsulation des paquets d'un protocole dans les paquets d'un autre protocole. La réponse recommandée est '`N`'.

L'option « `IP: GRE tunnels over IP` » permet d'autoriser l'encapsulation des protocoles IPv4 et IPv6 avec la méthode « GRE » (abréviation de l'anglais « Generic Routing Encapsulation »). Cette méthode d'encapsulation est destinée aux routeurs Cisco. La réponse recommandée est '`N`'.

L'option « `IP: broadcast GRE over IP` » permet de créer un réseau Ethernet virtuel sur IP par l'intermédiaire de la méthode d'encapsulation GRE, qui permet d'effectuer des broadcasts d'IP dans le réseau virtuel. La réponse recommandée est '`N`'.

L'option « `IP: multicast routing` » permet de configurer le système pour le routage des paquets ayant plusieurs destinations (c'est à dire les paquets IP envoyés en multicast). La réponse recommandée est '`N`'.

L'option « `IP: PIM-SM version 1 support` » permet d'activer la gestion du protocole de routage « PIM » des paquets envoyés en multicast. Ce protocole est géré par Cisco. La réponse recommandée est '`N`'.

L'option « `IP: PIM-SM version 2 support` » permet d'activer la gestion de la version 2 du protocole de routage PIM. La réponse recommandée est '`N`'.

L'option « `IP: ARP daemon support (EXPERIMENTAL)` » permet de limiter à 256 la table d'adresses physiques utilisées pour les requêtes ARP. Cette option est utile pour limiter la consommation mémoire du noyau dans les grands réseau. Les requêtes ne pouvant être satisfaites directement sont transférées à un démon, repoussant ainsi le reste de la table hors de la mémoire du noyau. La réponse recommandée est '`N`'.

L'option « `IP: TCP Explicit Congestion Notification support` » n'est pas encore documentée. Elle ne sera donc pas traitée dans ce document. La réponse recommandée est '`N`'.

L'option « `IP: TCP syncookie support (disabled per default)` » permet de protéger la machine d'une certaine forme d'attaque réseau. Le fait de répondre par '`Y`' à cette question inclut le support de cette protection, mais ne l'active pas par défaut. L'activation doit se faire par configuration dynamique du noyau via `/proc/`. La réponse recommandée est '`N`'.

L'option « `The IPv6 protocol (EXPERIMENTAL)` » permet d'activer la gestion du protocole IPv6. La réponse recommandée est '`N`'.

L'option « `IPv6: enable EUI-64 token format` » permet d'utiliser le format d'adresse EUI-64, qui est le nouveau format d'adresse utilisé par le protocole IPv6. Il faut répondre '`Y`' à cette question si l'environnement réseau utilise ce nouveau format, et '`N`' dans le cas contraire. La réponse recommandée est '`N`'.

L'option « `IPv6: disable provider based addresses` » permet de désactiver complètement l'ancien adressage d'IPv6. Cette option doit être choisie lorsque l'environnement réseau a complètement été basculé vers le nouveau mode d'adressage, ou si des problèmes apparaissent en raison de conflits entre les deux modes d'adressages dans un réseau qui n'a été que partiellement migré en IPv6. La réponse recommandée est '`N`'.

L'option « `IPv6: routing messages via old netlink` » permet d'obtenir les informations de routage de manière compatible avec l'ancien protocole (IPv4 ?). La réponse recommandée est '`N`'.

L'option « `Kernel httpd acceleration (EXPERIMENTAL)` » est une option permettant d'intégrer un petit serveur Web très optimisé au sein d'un thread du noyau. Ce serveur n'est capable de délivrer que des pages statiques, mais d'une manière extrêmement efficace car toutes les opérations sont effectuées au sein du noyau. Lorsqu'une requête HTTP ne peut pas être exécutée par le thread du noyau, celui-ci peut transférer la requête à un serveur Web classique, comme Apache par exemple, afin que celui-ci y réponde. Cette fonctionnalité est expérimentale et intègre du code classiquement située dans les applicatifs utilisateurs au sein du noyau. Aussi est-il vivement recommandé de répondre '`N`' à cette question et de laisser de côté cette fonctionnalité.

L'option « `Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)` » permet d'activer la gestion des réseaux ATM. ATM est un type de réseau travaillant en mode connecté (c'est à dire qu'une connexion permanente est établie entre les deux machines), ce qui permet d'effectuer une négociation initiale des ressources à allouer à cette connexion. Les réseaux ATM sont donc relativement adaptés aux transferts de données temps réel, comme la voix ou la vidéo. Les paquets transférés sont tous de taille fixe, ce qui permet de simplifier leur traitement et d'obtenir des débits très grands. ATM est utilisé aussi bien pour les réseaux de grande échelle que pour les réseaux locaux. Vous pouvez répondre par '`Y`' à cette question si vous êtes connectés à un réseau ATM. La réponse recommandée est '`N`'.

L'option « `Classical IP over ATM` » permet d'activer le support du protocole IP encapsulé dans un réseau virtuel ATM. Une alternative à cette option est l'option « `LAN Emulation (LANE) support` ». La réponse recommandée est '`N`'.

L'option « `Do NOT send ICMP if no neighbour` » permet d'éviter l'envoi de paquet ICMP signalant l'inaccessibilité d'une machine lorsque le noyau supprime temporairement la connexion à cette machine de ses tables internes pendant certaines opérations de maintenance. La réponse recommandée est '`N`'.

L'option « `LAN Emulation (LANE) support` » permet de simuler un réseau local classique sur un réseau ATM et éventuellement d'établir un pont entre ce réseau local virtuel et d'autre réseaux Ethernet réels. La réponse recommandée est '`N`'.

L'option « `Multi-Protocol Over ATM (MPOA) support` » permet l'établissement de canaux virtuels ATM au travers des limites des réseaux afin d'optimiser le routage des données. La réponse recommandée est '`N`'.

L'option « `The IPX protocol` » permet de prendre en charge le protocole réseau « IPX » de Novell. La réponse recommandée est '`N`'.

L'option « `IPX: Full internal IPX network` » permet de configurer le serveur pour qu'il apparaisse comme un réseau IPX à part entière, en lui assignant un numéro de réseau Novell. Toutes les requêtes seront redirigées vers des n½uds gérés en interne pour ce réseau virtuel. La réponse recommandée est '`N`'.

L'option « `Appletalk protocol support` » permet d'activer la gestion des réseaux AppleTalk. La réponse recommandée est '`N`'.

L'option « `DECnet support` » permet d'activer la gestion des réseaux DECnet, initialement créé par Digital (et repris maintenant par Compaq). La réponse recommandée est '`N`'.

L'option « `DECnet: SIOCGIFCONF support` » permet de paramétrer les interfaces réseaux via un appel système spécial. La validation de cette option peut provoquer des problèmes avec certains utilitaires, aussi la réponse recommandée est-elle '`N`'.

L'option « `DECnet: router support (EXPERIMENTAL)` » permet d'activer les fonctionnalités de reoutage sur les réseaux DECnet. La réponse recommandée est '`N`'.

L'option « `DECnet: use FWMARK value as routing key (EXPERIMENTAL)` » permet d'établir des règles de routages pour les réseaux DECnet qui se basent sur la valeur du champ MARK de ces paquets. Cette valeur peut être modifiée par le code de filtrage des paquets du noyau à l'aide des règles de firewalling. La réponse recommandée est '`N`'.

L'option « `802.1d Ethernet Bridging` » permet de configurer le système comme un pont Ethernet (un pont permet de regrouper physiquement plusieurs réseaux en un seul réseau physique). La réponse recommandée est '`N`'.

L'option « `CCITT X.25 Packet Layer (EXPERIMENTAL)` » permet d'activer la gestion du protocole de bas niveau pour X.25. La réponse recommandée est '`N`'.

L'option « `LAPB Data Link Driver (EXPERIMENTAL)` » permet d'activer la gestion du protocole de communication de haut niveau de X.25. La réponse recommandée est '`N`'.

L'option « `802.2 LLC (EXPERIMENTAL)` » permet d'encapsuler le protocole X.25 sur Ethernet. La réponse recommandée est '`N`'.

L'option « `Frame Diverter (EXPERIMENTAL)` » active une fonctionnalité permettant de détourner les trames réseaux qui arrivent sur l'interface réseau, même si ces trames ne sont pas destinées à l'interface réseau courante. Grâce à cette option, il est facile de réaliser des analyseurs réseau ou d'intercaler une machine Linux entre deux réseaux et de la configurer comme un pont afin de réaliser un cache transparent pour certains protocoles. La réponse recommandée est '`N`'.

L'option « `Acorn Econet/AUN protocols (EXPERIMENTAL)` » permet d'activer la gestion des réseaux Econet. La réponse recommandée est '`N`'.

L'option « `AUN over UDP` » permet d'encapsuler les paquets Econet dans UDP. La réponse recommandée est '`N`'.

L'option « `Native Econet` » permet de prendre en charge les cartes réseau Econet. La réponse recommandée est '`N`'.

L'option « `WAN rooter` » permet d'effectuer le routage sur un réseau « WAN » (abréviation de l'anglais « Wide Area Network »). La réponse recommandée est '`N`'.

L'option « `Fast switching (read help!)` » permet d'activer la communication entre les machines directement par l'intermédiaire des interfaces réseau. Cette option est incompatible avec les fonctionnalités de filtrage du noyau. Vous ne devez donc pas l'activer si vous désirez réaliser un Firewall ou partager une connexion à Internet. La réponse recommandée est '`N`'.

L'option « `Forwarding between high speed interfaces` » permet de configurer les drivers réseaux pour attendre avant de réémettre les paquets en cas de congestion extrême du réseau. La réponse recommandée est '`N`'.

9.3.11. Menu « IP: Netfilter Configuration »

L'option « `Connection tracking (required for masq/NAT)` » permet de suivre la connexions courantes afin de déterminer à quelle connexion un paquet appartient. Ceci peut être utile si l'on désire créer des règles de filtrage des paquets se basant sur les informations de gestion de connexions pour les protocoles comme TCP. Cette option est également nécessaire pour utiliser les mécanismes de translation d'adresses, vous devrez donc l'activer si vous désirez effectuer un partage de connexion à Internet. La réponse recommandée est '`N`'.

L'option « `FTP protocol support` » active la gestion du suivi des connexions FTP. Ces connexions nécessitent en effet un traitement particulier, et vous devrez activer cette option si vous voulez utiliser des connexions FTP avec un partage de connexion à Internet. La réponse recommandée est '`Y`'.

L'option « `Userspace queueing via NETLINK (EXPERIMENTAL)` » permet de mettre à disposition de programmes clients les paquets traités par le code de filtrage, par l'intermédiaire de l'interface Netlink. La réponse recommandée est '`N`'.

L'option « `IP tables support (required for filtering/masq/NAT)` » permet d'activer la gestion des tables au sein du code de filtrage du noyau. Une table est en réalité un ensemble de fonctionnalité cohérent permettant d'appliquer des traitements aux paquets selon des règles organisées

en groupes. Ces traitements peuvent intervenir à différents endroits dans la gestion des paquets par le code réseau du noyau. Les deux tables les plus importantes sont celles qui permettent de réaliser le filtrage des paquets et les translations d'adresses. Vous devez donc activer cette fonctionnalité si vous désirez réaliser un Firewall ou un partage de connexion à Internet. La réponse recommandée est 'N'.

L'option « `limit match support` » active la gestion de la limitation du nombre de fois par seconde qu'une règle peut être vérifiée par un paquet. Cette limitation est utile lorsque l'on enregistre des messages pour chaque paquet qui vérifie certaines règles, afin d'éviter l'engorgement des fichiers de traces du système. La réponse recommandée est 'N'.

L'option « `MAC address match support` » active la gestion du critère de sélection des paquets basé sur leur adresse Ethernet source. Cette règle n'est utilisable que pour les paquets provenant d'une interface réseau de type Ethernet. La réponse recommandée est 'N'.

L'option « `netfilter MARK match support` » active la gestion du critère de sélection des paquets basé sur le champ MARK de leurs en-têtes. Ce champ peut être modifiépar certaines règles des chaînes précédemment traversées par les paquets, afin de les marquer pour un traitement ultérieur. Cette option doit donc obligatoirement être activée si l'on désire détecter ces paquets pour effectuer ce traitement. La réponse recommandée est 'N'.

L'option « `Multiple port match support` » permet d'utiliser des plages de valeurs pour les ports TCP et UDP dans les critères de sélection des règles pour ces deux protocoles. Sans cette option, les ports doivent être spécifiés un à un, ce qui peut rendre relativement peu pratique la définition de certaines règles. La réponse recommandée est 'Y'.

L'option « `TOS match support` » active la gestion du critère de sélection des paquets basé sur le champ TOS de leurs en-têtes. Ce champ permet de définir le type de service des paquets, principalement afin de distinguer les paquets prioritaires des paquets normaux. La réponse recommandée est 'N'.

L'option « `Connection state match support` » permet de sélectionner les paquets selon leur rôle dans la gestion des connexions réseaux. Par exemple, cette option permet de distinguer les paquets qui établissent une connexion réseau des autres paquets. La réponse recommandée est 'Y'.

L'option « `Unclean match support (EXPERIMENTAL)` » permet de sélectionner les paquets dont les en-têtes IP ne sont pas corrects ou contiennent des valeurs incohérentes. Ces paquets doivent d'abord avoir traversé le code de validation de l'intégrité des paquets des interfaces réseau pour pouvoir être filtrés selon ce critère. La réponse recommandée est 'N'.

L'option « `Owner match support (EXPERIMENTAL)` » permet de sélectionner les paquets créés par les processus locaux en utilisant comme critère les identifiants de groupe, de processus et d'utilisateur de celui qui les a créé. Ces critères de sélection ne sont pas utilisables sur les paquets provenant de l'extérieur. La réponse recommandée est 'N'.

L'option « `Packet filtering` » active la gestion de la table `filter`, couramment utilisée pour filtrer les paquets autorisés et réaliser un Firewall. La réponse recommandée est 'N'.

L'option « `REJECT target support` » active la gestion de la cible `REJECT` dans les règles des chaînes de la table `filter`. Cette cible se distingue de la cible `DROP`, gérée nativement par la table

`filter`, par le fait qu'un message d'erreur est renvoyé à la machine source du paquet. La réponse recommandée est '`Y`'.

L'option « `MIRROR target support (EXPERIMENTAL)` » active la gestion de la cible `MIRROR`, qui permet de renvoyer à l'émetteur les paquets qui sont dirigés vers cette cible dans une règle de filtrage. La réponse recommandée est '`N`'.

L'option « `Full NAT` » active les fonctionnalités de translation d'adresse, au travers de la table `nat`. Cette option doit être activée si vous désirez réaliser un partage de connexion à Internet. Dans le cas contraire, la réponse recommandée est '`N`'.

L'option « `MASQUERADE target support` » active la gestion du masquerading des paquets réseau. Vous devez activer cette option si vous désirez réaliser un partage de connexion à Internet. La réponse recommandée est '`N`'.

L'option « `REDIRECT target support` » active la gestion de la translation d'adresses destination pour rediriger les paquets vers la machine locale. Cette option est très utilisée pour réaliser des proxies qui devront fonctionner de manière transparente pour les clients. La réponse recommandée est '`N`'.

L'option « `Packet mangling` » active la gestion de la table `mangle`. Cette table a pour but de donner les moyens d'effectuer diverses modifications des en-têtes des paquets, afin de les marquer pour un traitement ultérieur spécifique. La réponse recommandée est '`N`'.

L'option « `TOS target support` » permet d'utiliser la cible `TOS` dans les règles des chaînes de la table `mangle`. Cette cible autorise la modification du champ « Type Of Service » des paquets, principalement dans le but de modifier leur priorité. La réponse recommandée est '`N`'.

L'option « `MARK target support` » active la gestion de la cible `MARK`, qui permet de marquer les paquets avec un traceur afin de pouvoir les identifier ultérieurement. Par exemple, il est possible de modifier le routage de certains paquets, selon qu'ils sont marqués ou non. On peut ainsi réaliser des liaisons prioritaires pour certaines catégories de paquets. La réponse recommandée est '`N`'.

L'option « `LOG target support` » active la gestion de la cible `LOG`. Cette cible permet d'enregistrer des messages de débogage dans les fichiers de traces du système. On veillera à activer également la gestion des limites sur les règles de filtrages afin d'éviter d'engorger les fichiers de traces si l'on désire utiliser cette option. La réponse recommandée est '`N`'.

L'option « `ipchains (2.2-style) support` » n'est disponible que si l'on utilise les modules du noyau pour Netfilter. Elle permet d'utiliser les outils de configuration des Firewall et des translations d'adresses **ipchains**, qui étaient utilisés avec les noyaux 2.2. de Linux. Il s'agit donc d'une option de compatibilité. On préférera dorénavant la commande **iptables**, aussi la réponse recommandée est-elle '`N`'.

L'option « `ipfwadm (2.0-style) support` » n'est disponible que si l'on utilise les modules du noyau pour Netfilter. Elle permet d'utiliser les outils de configuration des Firewall et des translations d'adresses **ipfwadm**, qui étaient utilisés avec les noyaux 2.0. de Linux. Il s'agit donc d'une option de compatibilité. On préférera dorénavant la commande **iptables**, aussi la réponse recommandée est-elle '`N`'.

9.3.12. Menu « IPv6: Netfilter Configuration »

L'option « `IP6 tables support (required for filtering/masq/NAT)` » permet d'activer la gestion de Netfilter pour le protocole IPv6. La réponse recommandée est '`N`'.

L'option « `limit match support` » permet d'activer la gestion des limites sur le nombre de fois par seconde qu'une règle peut être vérifiée. Cette option permet par exemple d'éviter d'engorger les fichiers de traces du système lorsqu'un grand nombre de règles sont vérifiées. La réponse recommandée est '`N`'.

L'option « `netfilter MARK match support` » permet d'activer la gestion du critère de sélection des paquets basé sur le champ MARK de leur en-tête. Ce champ peut être modifié par les règles de la chaîne `mangle`, comme pour le protocole IPv4. La réponse recommandée est '`N`'.

L'option « `Packet filtering` » permet d'activer la gestion de la table `filter`, afin de réaliser par exemple un Firewall. La réponse recommandée est '`N`'.

L'option « `Packet mangling` » permet d'activer la gestion de la table `mangle`, qui autorise la modification de certains champs des en-têtes des paquets IP. La réponse recommandée est '`N`'.

L'option « `MARK target support` » permet de prendre en charge la gestion de la cible MARK, afin de marquer les paquets vérifiant certaines règles, par exemple pour effectuer un traitement ultérieur. La réponse recommandée est '`N`'.

9.3.13. Menu « QoS and/or fair queueing »

L'option « `QoS and/or fair queuing (EXPERIMENTAL)` » permet d'activer les options de configuration des algorithmes qui fixent les priorités sur les paquets à envoyer sur le réseau. La réponse recommandée est '`N`'.

Les options qui suivent permettent de choisir les algorithmes à utiliser pour la détermination des priorités d'émission. Parmi ces algorithmes, on retrouve les algorithmes basés sur la notion de qualité de service, qui peuvent être activés grâce à l'option « `QoS support` ». Les options suivantes permettent de paramétrer les notions attachées à la qualité de service.

9.3.14. Menu « Telephony Support »

L'option « `Linux telephony support` » permet d'activer la gestion des cartes téléphoniques. Ces cartes permettent de transférer les communications téléphoniques sur des supports numériques, par exemple en encapsulant les informations sonores sur un réseau TCP/IP. Cette option n'a rien à voir avec un modem, qui fait exactement l'inverse (transfert des données informatiques sur un réseau téléphonique analogique). La réponse recommandée est '`N`'.

L'option « `QuickNet Internet LineJack/PhoneJack support` » active la prise en charge des cartes téléphoniques Quicknet. Vous pouvez répondre '`Y`' ou '`M`' si vous possédez ce matériel, sinon la réponse recommandée est '`N`'.

9.3.15. Menu « ATA/IDE/MFM/RLL support »

L'option « `ATA/IDE/MFM/RLL support` » permet d'activer la gestion des disques durs et autres périphériques IDE. La réponse recommandée est '`Y`'. Les options spécifiques à chaque type de contrôleur sont accessibles dans le sous-menu « IDE, ATA and ATAPI Block devices ».

9.3.16. Menu « IDE, ATA and ATAPI Block devices »

L'option « `Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support` » permet d'activer la gestion des disques IDE. À moins que votre ordinateur ne soit complètement SCSI, répondez par '`Y`' ou '`M`' à cette question. Si vous répondez par '`M`', il faut que Linux soit installé sur un disque SCSI. Il est recommandé de répondre par '`Y`' à cette question.

L'option « `Use old disk-only driver on primary interface` » permet d'utiliser un vieux driver pour les disques IDE connectés à la première interface IDE et qui poseraient quelques problèmes avec le driver général. Seule la première interface sera concernée par cette option, les autres interfaces utiliseront le nouveau driver. En général, la réponse à cette question est '`N`'.

L'option « `Include IDE/ATA-2 DISK support` » permet d'utiliser les disques durs IDE et ATAPI. Si l'ordinateur possède un disque IDE, il est fortement recommandé de répondre '`Y`' à cette question. On ne doit répondre '`N`' que si l'ordinateur est complètement SCSI. De plus, on prendra garde au fait que l'on ne peut pas mettre les fonctionnalités nécessaires au démarrage de l'ordinateur dans des modules. Ceci signifie que si le disque de démarrage est un disque IDE, il ne faut pas mettre cette fonctionnalité dans un module. Donc, pour la plupart des gens, il faut répondre '`Y`' à cette question.

L'option « `Use multi-mode by default` » permet d'activer la gestion du mode de transfert multiple par défaut. Cette option n'est nécessaire que pour quelques cas particuliers, aussi la réponse recommandée est-elle '`N`'.

Il suit un certain nombre d'options permettant d'activer des fonctionnalités spécifiques aux principales grandes marques de disque dur. Ces options ne sont pas encore disponibles, et ne sont pas encore documentées. Elles ne seront donc pas décrites ici.

L'option « `PCMCIA IDE support` » permet d'activer la gestion des périphériques IDE connectés par un port PCMCIA sur les portables. Cette option n'est accessible que si vous avez activé la gestion des cartes PCMCIA avec l'option « `CardBus support` » du menu « PCMCIA/CardBus support ». La réponse recommandée est '`N`'.

L'option « `Include IDE/ATAPI CDROM support` » permet d'utiliser les CD-ROM IDE et ATAPI. La plupart des ordinateurs possèdent un lecteur de CD-ROM ATAPI actuellement, il est donc recommandé d'activer cette fonctionnalité. On essaiera d'utiliser cette fonctionnalité sous forme de module de préférence, parce que cette fonctionnalité n'est pas nécessaire en permanence. Cependant, si vous disposez d'un graveur de CD-ROM, il est recommandé de ne pas activer cette fonctionnalité. L'accès aux périphériques IDE ATAPI se fera alors par l'intermédiaire de l'émulateur de périphériques SCSI. Si vous disposez d'un lecteur de CD-ROM IDE et que ne voulez pas utiliser l'émulation SCSI, vous pouvez répondre '`Y`' à cette question. Répondez par '`N`' si vous ne possédez pas de lecteurs de CD-ROM IDE et si vous ne voudrez pas en installer un ultérieurement. La réponse recommandée est '`M`'.

L'option « `Include IDE/ATAPI TAPE support` » permet d'inclure la gestion des périphériques de sauvegarde à bande IDE et ATAPI (streamers). Peu de gens disposent de tels périphériques, et ceux qui en ont un ne l'utilisent que pour les sauvegardes, il est donc conseillé de répondre par '`M`' à cette question si vous possédez un tel périphérique, et par '`N`' sinon.

L'option « `Include IDE/ATAPI FLOPPY support` » permet d'inclure la gestion des lecteurs de disques amovibles IDE et ATAPI. C'est en particulier le cas pour les lecteurs LS120 et ZIP. Comme les lecteurs de disques amovibles IDE sont assez rares, la réponse recommandée est '`N`'.

L'option « `SCSI emulation support` » permet d'émuler la présence d'un périphérique SCSI par un périphérique IDE en convertissant les requêtes SCSI en requêtes ATAPI. Cette fonctionnalité est utile lorsque l'on veut utiliser certains logiciels bas niveau qui ne peuvent travailler qu'avec des périphériques SCSI. C'est en particulier le cas des logiciel de gravage de CD. On notera que cette fonctionnalité ne sera pas activable si le pilote IDE est actif. Il est donc nécessaire d'utiliser celui-ci sous la forme de module, ou de ne pas l'utiliser du tout. La réponse recommandée est '`N`' si vous ne disposez pas d'un graveur de CD-ROM ATAPI, et '`Y`' dans le cas contraire.

L'option « `CMD640 chipset bugfix/support` » permet de contourner une bogue des chipsets IDE CMD640. Vous ne devez activer cette option que si vous disposez d'un tel chipset. La réponse recommandée est '`Y`' si vous avez un tel chipset, et '`N`' sinon.

L'option « `CMD640 enhanced support` » permet l'autodétection des paramètre idéaux pour les chipset IDE CMD640. Cette option n'est disponible que si vous avez activé le support des chipsets CMD640 dans la question précédente. La réponse recommandée est '`Y`' si vous avez un tel chipset, et '`N`' sinon.

L'option « ISA-PNP EIDE support » permet de prendre en charge la gestion des cartes ISA Plug and Play prenant en charge des disques EIDE additionnels. Cette option peut être nécessaire si ces cartes doivent être initialisées avant de chercher à utiliser les disques qui y sont connectées. La réponse recommandée est '`N`'.

L'option « `RZ1000 chipset bugfix/support` » permet de contourner une bogue des chipsets RZ1000. Vous ne devez activer cette option que si vous disposez d'un tel chipset. La réponse recommandée est '`Y`' si vous avez un tel chipset, et '`N`' sinon.

L'option « `Generic PCI IDE chipset support` » permet d'activer la gestion des périphériques IDE sur bus PCI. Vous devez répondre par '`Y`' à cette question si vous disposez d'une carte mère PCI et de contrôleurs IDE.

L'option « `Sharing PCI IDE interrupts support` » permet d'activer le partage des lignes d'interruptions utilisées par les contrôleurs IDE avec les cartes présentes sur le bus PCI. Cette fonctionnalité nécessite un support matériel particulier de la part du contrôleur IDE, support qui n'est présent que sur certaines cartes mères. En général, l'activation de cette option ne perturbe pas le fonctionnement du système sur les ordinateurs incapables d'effectuer le partage des lignes d'interruptions des contrôleurs IDE, mais la réponse recommandée reste '`N`'.

L'option « `Generic PCI bus-master DMA support` » permet d'activer la gestion des disques Ultra DMA. La réponse recommandée est '`Y`' si vous disposez d'une carte mère PCI gérant l'Ultra DMA et de disques IDE.

L'option « `Boot off-board chipsets first support` » permet d'inverser la numérotation des disques IDE connectés aux contrôleurs de la carte mère et des disques IDE connectés aux contrôleurs additionnels que l'on peut avoir sur une carte fille. Il faut répondre par '`N`' à cette question.

L'option « `Use PCI DMA by default when available` » permet d'activer la gestion de l'Ultra DMA au démarrage. La réponse recommandée est '`Y`', sauf si l'on veut désactiver le support de l'Ultra DMA.

L'option « `ATA Work(s) In Progress (EXPERIMENTAL)` » donne l'accès à des fonctionnalités extrêmement expérimentales concernant les disques IDE. La réponse recommandée est '`N`'.

Il suit un certain nombre d'options qui permettent d'activer un support étendu pour un différents chipsets. Il est recommandé de répondre par '`N`' à ces questions, sauf à celles correspondant aux chipsets effectivement présents sur votre carte mère.

L'option « `IGNORE word93 Validation BITS` » permet d'éviter une inconsistance dans les spécifications du protocole matériel ATAPI. Ces spécifications n'ont pas été claires à un endroit, et il existe maintenant des différences mineures entre les différents chipsets présents sur le marché. Cette option permet de désactiver cette fonctionnalité ambiguë. Bien qu'il ne soit pas dangereux de répondre par l'affirmative à cette question, la réponse recommandée reste '`N`'.

9.3.17. Menu « SCSI support »

L'option « `SCSI support` » permet d'activer la prise en charge des périphériques SCSI. Il faut répondre par '`Y`' ou '`M`' à cette question si vous disposez de tels périphériques. De plus, si le noyau se trouve sur un disque SCSI ou s'il a besoin de composants se trouvant sur le disque SCSI pour s'amorcer, il ne faut pas mettre en module cette fonctionnalité. Ceux qui n'ont pas de périphériques SCSI et ne comptent pas en utiliser peuvent répondre '`N`' à cette question. En revanche, ceux qui disposent d'un périphérique ATAPI pour lequel ils n'ont pas de drivers, et pour lequel ils utilisent la couche d'émulation SCSI, doivent activer cette fonctionnalité. C'est en particulier le cas si vous avez un graveur de CD. Dans ce cas, il est recommandé de répondre '`Y`' à cette question. Même dans les autres cas, il est conseillé de répondre par '`M`' à cette question, afin de se réserver la possibilité de connecter ultérieurement un lecteur ZIP connecté sur port parallèle, qui nécessitera alors le support SCSI.

L'option « `SCSI disk support` » permet d'activer la gestion des disques durs SCSI. Il ne faut pas répondre par '`M`' si votre système a besoin d'un disque SCSI pour démarrer. Cette option permet également de gérer les lecteurs ZIP connectés sur port parallèle. La réponse recommandée est donc '`M`'.

L'option « `Maximum number of SCSI disks that can be loaded as modules` » permet de fixer le nombre maximum de disques SCSI qui pourront être utilisés si les drivers SCSI du noyau sont chargés en tant que modules. Il est recommandé de laisser la valeur par défaut dans ce champ.

L'option « `SCSI tape support` » permet d'activer la gestion des périphériques à bande SCSI. La réponse recommandée est '`N`'.

L'option « `SCSI CD-ROM support` » permet d'activer la gestion des CD-ROM SCSI. Vous devez activer cette fonctionnalité également si vous utilisez l'émulation SCSI pour un graveur de CD ATAPI.

Si vous êtes dans l'un de deux ces cas, il est recommandé de répondre 'Y' à cette question. Dans tous les autres cas, la réponse recommandée est 'N'.

L'option « Enable vendor-specific extensions (for SCSI CDROM) » permet d'activer la gestion de CD-ROM SCSI disposant de commandes SCSI spécifiques. C'est notamment le cas pour les CD multisessions NEC/TOSHIBA et les graveurs de CD HP. Pour la plupart des utilisateurs, la réponse recommandée est 'N'.

L'option « Maximum number of CDROM devices that can be loaded as modules » permet de fixer le nombre maximum de lecteurs de CDROM SCSI qui pourront être utilisés si les drivers du noyau sont chargés en tant que modules. Il est recommandé de laisser la valeur par défaut dans ce champ.

L'option « SCSI generic support » permet d'activer la gestion des périphériques SCSI non standards. Pour ces périphériques, il faut utiliser un programme capable d'envoyer les commandes SCSI appropriées à votre matériel. Il faut activer cette fonctionnalité pour la plupart des périphériques SCSI qui ne sont ni des disques, ni des lecteurs de CD-ROM, ni des lecteurs de bandes. C'est en particulier le cas si vous utilisez un graveur de CD-ROM. Dans ce cas, la réponse recommandée est 'Y'. Pour la plupart des utilisateurs, la réponse recommandée est 'N'.

L'option « Enable extra checks in new queueing code » permet d'activer des contrôles supplémenaires dans les drivers SCSI. Si une erreur est détectée, le système s'arrêtera en catastrophe immédiatement, limitant ainsi les risques de pertes de données ultérieurs. Si vous n'activez pas cette option, le système continuera à fonctionner normalement, mais les dégâts risquent d'être beaucoup plus importants à terme si une erreur se produit. La réponse recommandée est 'Y'.

L'option « Probe all LUNs on each SCSI device » permet d'effectuer la détection de tous les numéros logiques d'unités SCSI de chaque périphérique. Comme la plupart des périphériques SCSI ne disposent que d'un seul numéro d'unité logique, la réponse recommandée est 'N'.

L'option « Verbose SCSI error reporting (kernel size +=12K) » permet d'utiliser un jeu de messages d'erreurs alternatif pour le SCSI. Ces messages sont plus lisibles, mais prennent plus de place dans le noyau. La réponse recommandée est 'N'.

L'option « SCSI logging facility » permet d'activer les traces du sous-système SCSI. La réponse recommandée est 'N'.

9.3.18. Menu « SCSI low-level drivers »

Ce jeux d'options permet de sélectionner le driver bas niveau SCSI adapté à votre matériel. Il faut connaître la marque et le modèle de votre adaptateur SCSI, ainsi que ses paramètres pour répondre à ces questions. Si vous utilisez l'émulation SCSI pour les périphériques ATAPI, il n'est pas nécessaire de choisir un driver bas niveau pour ceux-ci.

9.3.19. Menu « PCMCIA SCSI adapter support »

L'option « `PCMCIA SCSI adapter support` » permet la prise en charge les périphériques SCSI au format PCMCIA. Les options qui suivent correspondent aux drivers des différents types de matériels SCSI au format PCMCIA que Linux est capable de gérer. Vous devez donc choisir les drivers correspondants à votre matériel. La réponse recommandée est '`N`'.

9.3.20. Menu « IEEE 1394 (FireWire) support »

L'option « `IEEE 1394 (FireWire) support (EXPERIMENTAL)` » permet d'activer la gestion des cartes FireWire. Ces cartes fournissent l'accès à un bus de données extrêmement rapide, que l'on utilise généralement pour connecter des périphériques exigeant une bande passante très élevée, comme les caméra vidéo par exemple. La réponse recommandée est '`N`'.

L'option « `Texas Instruments PCILynx support` » permet d'activer la gestion des drivers pour les cartes FireWire PCILynx de Texas Instruments. La réponse recommandée est '`N`'.

L'option « `Use PCILynx local RAM` » permet d'utiliser la mémoire vive embarquée dans certaines cartes PCILynx de Texas Instruments afin d'améliorer les performances. Ce type de carte est très rare, et ce driver ne fonctionne pas du tout avec les cartes qui ne disposent pas de mémoire vive embarquée, aussi faut-il répondre '`N`' à cette question.

L'option « `Support for non-IEEE1394 local ports` » permet d'accéder aux ressources matérielles des cartes PCILynx par l'intermédiaire de fichiers spéciaux de périphériques dédiés, au lieu de passer par l'interface FireWire standard. La réponse recommandée est '`N`'.

L'option « `OHCI (Open Host Controller Interface) support` » active la prise en charge des contrôleurs IEEE 1394 respectant les spécifications OHCI (il s'agit d'un standard de communication pour les contrôleurs). Ce driver n'a été testé qu'avec un contrôleur de Texas Instruments, mais ce contrôleur est l'un des plus utilisés du marché. La réponse recommandée est '`Y`'.

L'option « `Video 1394 support` » n'est pas documentée et ne sera pas décrite dans ce document. La réponse recommandée est '`N`'.

L'option « `Raw IEEE 1394 I/O support` » permet aux programmes de communiquer directement avec le matériel IEEE 1394. C'est en général le mode de fonctionnement désiré, aussi la réponse recommandée à cette question est-elle '`Y`'.

L'option « `Excessive debugging support` » est réservée pour les développeurs de drivers. Elle permet de stocker sur disque toutes les informations transitant sur le bus FireWire, ce qui sature généralement les disques durs extrêmement rapidement. Il faut répondre par '`N`' à cette question.

9.3.21. Menu « I2O support »

L'option « `I2O support` » permet d'activer la gestion des cartes d'entrée / sorties I2O. Ces cartes prennent en charge la gestion des entrées / sorties de manière uniforme, pour tous les périphériques matériels, et permettent donc d'écrire un driver uniforme au niveau du système d'exploitation pour toute une classe de périphériques. Ce driver uniforme (« OSM », abréviation de « Operating System Module ») communique avec des drivers spécifiques à chaque type de matériel, qui ne font aucune

hypothèse sur le système d'exploitation utilisé. Ainsi, il est possible d'écrire des drivers communs à tous les systèmes d'exploitations. La réponse recommandée est 'N'. Si vous répondez par 'Y', vous devrez choisir le type de carte I2O installée et la liste des drivers I2O OSM que vous voulez utiliser.

9.3.22. Menu « Network device support »

L'option « Network device support » permet d'activer la gestion des diverses interfaces réseau que Linux peut prendre en charge. On notera que les choix faits pour les protocoles de communication dans le menu « Networking options » sont nécessaires (la plupart des composants des systèmes Unix utilisent les fonctions réseau du système pour communiquer) mais non suffisants pour accéder au monde extérieur. Si l'on n'active pas cette option, seuls les processus de la machine locale seront accessibles (par l'intermédiaire de l'interface réseau « loopback »). Pour accéder aux autres ordinateurs, que ce soit par une carte réseau, un câble parallèle ou série, ou par modem via un fournisseur d'accès à Internet, il faut en plus répondre 'Y' à cette question. La réponse recommandée est 'Y'.

L'option « Dummy net driver support » permet d'activer la fonctionnalité de réseau virtuel. Cette fonctionnalité fournit une interface virtuelle sur un réseau ne contenant qu'une seule machine. Cette interface peut être utilisée pour faire croire à des programmes réseaux que la machine est effectivement connectée à un réseau. La réponse recommandée est 'N'.

L'option « Bonding driver support » permet de réunir plusieurs connexions Ethernet entre deux machines pour simuler une connexion unique, dont la bande passante est la somme des bandes passantes des connexions ainsi regroupées. La réponse recommandée est 'N'.

L'option « EQL (serial line load balancing) support » permet d'activer la gestion de connexions multiples sur plusieurs connexions séries (par exemple par l'intermédiaire de deux modems). Cette fonctionnalité nécessite également le support de cette fonctionnalité du côté de la machine distante. La réponse recommandée est 'N'.

L'option « Universal TUN/TAP device driver support » permet d'activer la gestion d'interfaces réseau virtuelles tunX et tapX (où 'X' est le numéro de chaque interface), dont les données peuvent être lues et écrites directement par des applications normales, par l'intermédiaire de fichiers spéciaux de périphériques /dev/tunX et /dev/tapX. Les interfaces de type tun se comportent exactement comme des interfaces réseau point à point (elles ne permettent donc de communiquer qu'avec qu'une seule autre machine), alors que les interfaces de type tap simulent le fonctionnement d'une carte Ethernet classique. Cette fonctionnalité permet donc aux programmes classiques d'accéder aux paquets routés sur leurs interfaces, et de recevoir et d'envoyer des données brutes sur le réseau. Les interfaces tunX et tapX sont créées dynamiquement, lorsque les programmes qui désirent les utiliser s'enregistrent au niveau du système en ouvrant le fichier spécial de périphérique /dev/net/tun, de type caractère et de codes majeur et mineur égaux respectivement à 10 et 200. La réponse recommandée est 'N'.

L'option « Ethertap network tap (OBSOLETE) » n'est activée que si l'option « Kernel/User netlink socket » a été activée. Elle permet de gérer un fichier spécial de type block et de numéro majeur 36 et de numéro mineur 16, qui fournit les données brutes d'une interface Ethernet. Ce fichier peut être utilisé par un programme pour envoyer et recevoir des trames Ethernet directement sur ce pseudo périphérique au plus bas niveau. Cette fonctionnalité est devenue obsolète depuis le noyau

2.4.0, car celui-ci propose le mécanisme plus général de l'option « `Universal TUN/TAP device driver support` ». La réponse recommandée est donc '`N`'.

L'option « `General Instruments Surfboard 1000` » permet d'activer la gestion des cartes Sufboard 1000, qui sont généralement utilisées pour connecter un modem câble à l'ordinateur. Ces modems permettent de recevoir des informations en provenance d'Internet à haut débit, mais ne permettent pas d'envoyer d'informations à destination d'Internet. Le canal montant doit donc toujours être réalisé par une liaison téléphonique classique. La réponse recommandée est '`N`'.

L'option « `FDDI driver support` » permet d'activer la gestion des cartes Ethernet FDDI. Les options qui suivent permettent de sélectionner les drivers pour les cartes FDDI gérées par Linux. La réponse recommandée est '`N`'.

L'option « `HIPPI driver support (EXPERIMENTAL)` » permet la gestion des cartes HIPPI. Les options qui suivent permettent de sélectionner les drivers pour les cartes de ce type. La réponse recommandée est '`N`'.

L'option « `PLIP (parallel port) support` » permet d'activer les connexions par port parallèle. Ce type de connexion peut être utilisé pour transférer des fichiers par un câble parallèle entre deux ordinateurs. La réponse recommandée est '`N`'.

L'option « `PPP (point-to-point protocol) support` » permet d'activer la gestion des connexions PPP. Ces connexions sont utilisées pour accéder à Internet via un modem, ou pour établir une connexion de manière plus générale via un câble série. Il est conseillé d'activer le support de ce type de connexion, à moins que vous ne soyez sûr de ne jamais vous connecter à Internet. La réponse recommandée ici est '`Y`'.

L'option « `PPP multinlink support (EXPERIMENTAL)` » permet d'utiliser plusieurs lignes PPP pour augmenter la bande passante en les agrégeant. Par exemple, si vous disposez de plusieurs lignes téléphoniques, vous pouvez vous connecter plusieurs fois à votre fournisseur d'accès et accroître ainsi votre bande passante. Bien entendu, pour que cette technique fonctionne, il faut que le fournisseur d'accès l'autorise. La réponse recommandée est '`N`'.

L'option « `PPP support for async serial ports` » permet de prendre en charge les communications PPP sur les câbles série classique. C'est en général l'option qui convient lorsque l'on veut se connecter à Internet à l'aide d'un modem classique. Notez que cette option n'est pas utilisable pour les connexions à l'aide d'un modem Numéris. La réponse recommandée est '`Y`'.

L'option « `PPP support for sync tty ports` » permet de prendre en charge les communications PPP avec les adaptateurs synchones, tels que les cartes SyncLink. La réponse recommandée est '`N`'.

L'option « `PPP Deflate compression` » permet de prendre en charge l'algorithme de compression Deflate (le même algorithme que celui utilisé par le programme **gzip**) pour compresser les données transmises sur les connexions PPP. La réponse recommandée est '`Y`'.

L'option « `PPP BSD-Compress compression` » permet de prendre en charge l'algorithme de compression LZW pour compresser les données transmises sur les connexions PPP. Cet algorithme est soumis à un brevet logiciel, mais ces brevets sont illégaux en Europe et il est tout à fait légal de l'uti-

liser. Cependant, les taux de compressions obtenus sont inférieurs à ceux de la méthode Deflate, aussi la réponse recommandée est-elle 'N'.

L'option « PPP over Ethernet (EXPERIMENTAL) » permet de réaliser des connexions point à point sur un réseau de type Ethernet. La réponse recommandée est 'N'.

L'option « SLIP (serial line) support » permet de connecter deux ordinateurs par une ligne série ou par un modem. Les options suivantes permettent de fixer les paramètres des connexions SLIP. Le protocole SLIP est en train de tomber en désuétude et est remplacé par PPP. La réponse recommandée à cette question est donc 'N'.

L'option « CSLIP compressed headers » permet d'activer la compression des en-têtes IP pour le protocole SLIP. La réponse recommandée est 'N'.

L'option « Keepalive and linefill » permet d'activer la surveillance de la connexion SLIP. Cette option est utilisée sur les connexions SLIP passant par des lignes analogiques de mauvaise qualité. La réponse recommandée est 'N'.

L'option « Six bit SLIP encapsulation » permet de faire en sorte que le protocole SLIP n'envoie que des données codées sur 6 bits. Ceci permet d'améliorer la qualité des transmissions sur les réseaux peu fiables, ou qui ne peuvent pas transférer plus 7 bits de données. La réponse recommandée est 'N'.

L'option « Fibre Channel driver support » permet d'activer la gestion des adaptateurs Fiber Channel. L'option qui suit donne la possibilité de sélectionner le driver pour les cartes Interphase à base de chipset Tachyon. La réponse recommandée est 'N'.

L'option « Red Creek Hardware VPN (EXPERIMENTAL) » permet d'activer la gestion des cartes permettant de créer des réseaux privés virtuels. La réponse recommandée est 'N'.

L'option « Traffic Shaper (EXPERIMENTAL) » permet d'activer la possibilité de contrôler le débit maximal de données à travers une interface réseau. La réponse recommandée est 'N'.

9.3.23. Menu « ARCnet devices »

L'option « ARCnet support » permet d'activer la gestion des cartes de type ARCnet. Les options qui suivent correspondent aux différents drivers pour les différentes cartes de ce type. Si vous avez une carte de ce type, répondez par 'M' ou 'Y' à cette question et choisissez le driver correspondant à votre matériel dans la suite du menu. La réponse recommandée est 'N'.

9.3.24. Menu « AppleTalk devices »

L'option « Appletalk interfaces support » active la gestion des interfaces réseau Appletalk. Ce sont des interfaces réseau permettant de communiquer sur les réseau Apple d'ancienne génération. La réponse recommandée est 'N'.

Les options « Apple/Farallon LocalTalk PC support » et « COPS LocalTalk PC support » permettent de gérer les adaptateurs LocalTalk pour PC. Les options qui suivent permettent de choisir les drivers appropriés à votre matériel. La réponse recommandée est 'N'.

L'option « `AppleTalk-IP driver support` » permet de se connecter à un réseau utilisant le protocole IP via un adaptateur réseau AppleTalk. Cette fonctionnalité est expérimentale. La réponse recommandée est '`N`'.

L'option « `IP to Appletalk-IP Encapsulation support` » permet d'encapsuler les paquets IP dans des paquets AppleTalk. Cette fonctionnalité est utilisée par les machines Linux qui n'ont accès qu'à un réseau Appletalk. La réponse recommandée est '`N`'.

L'option « `Appletalk-IP to IP Decapsulation support` » permet d'extraire les paquets IP encapsulés dans des paquets IP. Cette fonctionnalité est utilisée par les machines Linux qui font office de passerelles vers Internet pour les machines d'un réseau Appletalk. La réponse recommandée est '`N`'.

9.3.25. Menu « Ethernet (10 or 100Mbit) »

L'option « `Ethernet (10 or 100Mbit)` » permet d'activer la gestion des cartes de type Ethernet. Les options qui suivent correspondent aux différents drivers pour les différentes cartes de ce type. Si vous avez une carte de ce type, répondez par '`M`' ou '`Y`' à cette question et choisissez le driver correspondant à votre matériel dans la suite du menu. La réponse recommandée est '`N`'. La plupart des cartes réseau sont compatibles NE2000. Il existe deux types de drivers pour ces cartes, selon qu'elles sont ISA ou non. Le driver ISA peut être choisi en activant les options « `Other ISA cards` » et « `NE2000/NE1000 support` ». Le driver pour les cartes PCI peut être sélectionné en activant les options « `EISA, VLB, PCI and on board controllers` » et « `PCI NE2000 support` ».

9.3.26. Menu « Ethernet (1000 Mbit) »

Les options disponibles dans ce menu permettent de choisir les drivers à utiliser pour les cartes Ethernet Gigabit. Choisissez l'option qui correspond à votre matériel si vous en disposez d'une.

9.3.27. Menu « Wireless LAN (non-hamradio) »

L'option « `Wireless LAN (non-hamradio)` » permet d'activer la gestion des connexions sans fil (mais qui ne sont pas gérées par les fonctionnalités radio-amateurs). Les options qui suivent permettent de choisir les drivers disponibles. La réponse recommandée est '`N`'.

L'option « `STRIP (Metricom starmode radio IP)` » permet d'activer la gestion des adaptateurs réseau radio Metricom. La réponse recommandée est '`N`'.

L'option « `AT&T WaveLAN & DEC RoamAbout DS support` » permet d'activer la gestion des adaptateurs réseau radio WaveLAN. La réponse recommandée est '`N`'.

L'option « `Aironet Arlan 655 & IC2200 DS support` » permet d'activer la gestion des adaptateurs réseau radio Aironet 655 et IC2200 DS. La réponse recommandée est '`N`'.

L'option « `Aironet 4500/4800 series adapters` » permet d'activer la gestion des adaptateurs réseau radio Aironet de la série 4500/4800. Les options qui suivent correspondent aux différents types

de cartes de cette série. L'option « `Aironet 4500/4800 PROC interface` » donne la possibilité de configurer ces cartes par l'intermédiaire du système de fichiers virtuels `/proc/`. La réponse recommandée est '`N`'.

9.3.28. Menu « Token ring devices »

L'option « `Token Ring driver support` » permet d'activer la gestion des réseaux Token Ring. Les options qui suivent permettent de choisir les drivers pour ces adaptateurs. La réponse recommandée est '`N`'.

9.3.29. Menu « Wan interfaces »

L'option « `Comtrol Hostess SV-11 support` » permet d'activer la gestion des cartes Comtrol. La réponse recommandée est '`N`'.

L'option « `COSA/SRP sync serial boards support` » permet d'activer la gestion des cartes COSA et SRP. La réponse recommandée est '`N`'.

L'option « `MultiGate (COMX) synchronous serial boards support` » permet d'activer la gestion des drivers des cartes MultiGate. La réponse recommandée est '`N`'. Les options qui suivent permettent de paramétrer les fonctionnalités supportées par le driver.

L'option « `LanMedia Corp. SSI/V.35, T1/E1, HSSI, T3 boards` » permet d'activer la gestion des drivers des cartes LanMedia. La réponse recommandée est '`N`'.

L'option « `Sealevel Systems 4021 support` » permet d'activer la gestion des drivers des cartes Sealevel ACB 56. La réponse recommandée est '`N`'.

L'option « `SyncLink HDLC/SYNCPPP support` » n'est pas documentée à l'heure actuelle. La réponse recommandée est '`N`'.

L'option « `Frame relay DLCI support` » permet d'activer la gestion du protocole Frame Relay. Les options qui suivent permettent de configurer le driver. La réponse recommandée est '`N`'.

L'option « `WAN router drivers` » permet d'activer les options de menu suivantes, qui donnent la possibilité de faire la configuration des adaptateurs WAN. La réponse recommandée est '`N`'.

L'option « `Sangoma WANPIPE(tm) multiprotocol cards` » permet d'activer la gestion des cartes WAN Sangoma. Les options qui suivent permettent de configurer les paramètres pour ce driver. La réponse recommandée est '`N`'.

L'option « `LAPB over Ethernet driver` » permet d'activer un driver pour une carte X.25 virtuelle. La réponse recommandée est '`N`'.

L'option « `X.25 async driver` » permet d'activer la gestion du protocole X.25 sur une ligne série normale ou sur un modem. La réponse recommandée est '`N`'.

L'option « `SBNI12-xx support` » permet d'activer la gestion des cartes SBNI12-xx. La réponse recommandée est '`N`'.

9.3.30. Menu « PCMCIA network device support »

L'option « `PCMCIA network device support` » permet d'activer la gestion des cartes réseau au format PCMCIA. Les options qui suivent correspondent aux divers drivers pris en charge par Linux. La réponse recommandée est '`N`'.

9.3.31. Menu « ATM drivers »

L'option « `ATM over TCP` » permet d'encapsuler ATM sur un réseau TCP/IP. Ce type d'encapsulation n'est utile qu'à des fin de test et de développement. La réponse recommandée est '`N`'.

L'option « `Efficient Networks ENI155P` » active la gestion des cartes ATM ENI155P d'Efficient Networks et Power155 de SMC. Les options qui suivent permettent de configurer le driver. La réponse recommandée est '`N`'.

L'option « `Fujitsu FireStream (FS50/FS155)` » active la gestion des cartes ATM FireStream de Fujitsu. La réponse recommandée est '`N`'.

L'option « `ZeitNet ZN1221/ZN1225` » active la gestion des cartes ATM ZN1221 et ZN1225 de ZeitNet. Les options qui suivent permettent de configurer le driver. La réponse recommandée est '`N`'.

L'option « `IDT 77201 (NICStAR) (ForeRunnerLE)` » active la gestion des cartes ATM basées sur le chipset NICStAR. Les options qui suivent permettent de configurer le driver. La réponse recommandée est '`N`'.

L'option « `Madge Ambassador (Collage PCI 155 Server)` » active la gestion des cartes ATM basées sur le chipset ATMizer. L'option qui suit permet d'activer les informations de débogage du driver. La réponse recommandée est '`N`'.

L'option « `Madge Horizon [Ultra] (Collage PCI 25 and Collage PCI 155 Client)` » active la gestion des cartes ATM basées sur le chipset Horizon. L'option qui suit permet d'activer les informations de débogage du driver. La réponse recommandée est '`N`'.

L'option « `Interphase ATM PCI x575/x525/x531` » active la gestion des cartes ATM ChipSAR d'Interphase. L'option qui suit permet d'activer les informations de débogage du driver. La réponse recommandée est '`N`'.

L'option « `FORE Systems 200E-series` » active la gestion des cartes ATM FORE Systems 200E. Les options qui suivent permettent de configurer le driver. La réponse recommandée est '`N`'.

9.3.32. Menu « Amateur Radio support »

L'option « `Amateur Radio support` » permet d'activer la gestion des communications Radio de Linux. Les options qui suivent permettent de préciser les options du protocole réseau utilisé pour ces communications. La réponse recommandée est '`N`'.

9.3.33. Menu « AX.25 network device drivers »

Ce menu permet de choisir les drivers bas niveau à utiliser pour le support des communications Radio. Il faut choisir le driver correspondant à votre matériel.

9.3.34. Menu « IrDA subsystem support »

L'option « `IrDA subsystem support` » permet d'activer la gestion des périphériques infrarouges. La réponse recommandée est '`N`'.

Il faut choisir le protocole de communication que vous désirez dans les options qui suivent. Les dernière options permettent de choisir les options de ces protocoles.

9.3.35. Menu « Infrared-port device drivers »

Ces options permettent de choisir les drivers bas niveau pour les périphériques infrarouges. Vous devez choisir le driver correspondant à votre matériel.

9.3.36. Menu « ISDN subsystem »

L'option « `ISDN support` » permet d'activer la gestion des interfaces « ISDN » (« Numéris » en France). Si vous disposez d'une telle interface, la réponse recommandée est '`M`', sinon, c'est '`N`'.

L'option « `Support synchronous PPP` » permet d'utiliser une version de PPP qui ne gère pas les synchronisations dans la communication entre les deux ordinateurs. Cette fonctionnalité n'est en effet plus nécessaire avec ISDN, puisque c'est un protocole numérique. La réponse recommandée est '`Y`'.

L'option « `Use VJ-compression with synchronous PPP` » permet d'utiliser l'algorithme de Van Jacobson pour la compression des en-têtes IP dans le protocole PPP synchrone. La réponse recommandée est '`Y`'.

L'option « `Support generic MP (RFC 1717)` » permet de regrouper plusieurs connexions ISDN avec le protocole PPP synchrone, et ce afin d'accroître la bande passante. La réponse recommandée est '`N`'.

L'option « `Support BSD compression (module only)` » active la gestion de la compression de donnée avec l'algorithme BSD. Cette fonctionnalité n'est disponible que sous la forme de module pour des raisons de licences logicielles. La réponse recommandée est '`N`'.

L'option « `Support audio via ISDN` » permet d'activer la gestion des commandes vocales dans l'émulateur de modem du sous-système ISDN. La réponse recommandée est '`N`'.

L'option « `Support AT-Fax Class 1 and 2 commands` » permet d'activer la gestion des commandes vocales des FAX. La réponse recommandée est '`N`'.

L'option « `X.25 PLP on top of ISDN` » permet d'activer l'encapsulation du protocole X.25 dans le protocole ISDN. La réponse recommandée est '`N`'.

9.3.37. Menu « ISDN feature submodules »

L'option « `isdnloop support` » permet d'activer la gestion d'une interface ISDN virtuelle, afin de tester la configuration sans effectuer d'appel réel. La réponse recommandée est '`N`'.

L'option « `Support isdn diversion services` » permet d'activer la gestion de services ISDN supplémentaires. La réponse recommandée est '`N`'.

9.3.38. Menu « Passive ISDN cards »

L'option « `HiSax SiemensChipSet driver support` » permet d'activer les options de configuration pour les cartes à base de chipsets Siemens. Ce sont les cartes les plus courantes. Les options qui suivent permettent de sélectionner le type de matériel correspondant. La réponse recommandée est '`N`'.

9.3.39. Menu « Active ISDN cards »

L'option « `ICN 2B and 4B support` » permet d'activer le support pour des cartes ISDN fabriquées par la société ICN. La réponse recommandée est '`N`'.

L'option « `PCBIT-D support` » permet d'activer la gestion des cartes ISDN fabriquées par la société Octal. La réponse recommandée est '`N`'.

L'option « `Spellcaster support` » permet d'activer la gestion des cartes ISDN Spellcaster. La réponse recommandée est '`N`'.

L'option « `IBM Active 2000 support` » permet d'activer la gestion des cartes ISDN IBM Active 2000. La réponse recommandée est '`N`'.

L'option « `Eicon active card support` » permet d'activer la gestion des cartes Eicon. Les options qui suivent permettent de choisir le driver approprié au matériel dont vous disposez. La réponse recommandée est '`N`'.

L'option « `CAPI2.0 support` » permet d'activer la gestion de l'interface de programmation CAPI (« Common ISDN Application Programming Interface », permettant aux programmes clients d'utiliser de manière uniforme toutes les cartes ISDN. La réponse recommandée est '`Y`'.

L'option « `Verbose reason code reporting (kernel size +=7K` » permet d'indiquer le degré de traces utilisé par le driver des cartes AVM. La réponse recommandée est '`N`'.

L'option « `CAPI2.0 Middleware support (EXPERIMENTAL)` » n'est pas encore documentée et ne sera pas décrite dans ce document.

L'option « `CAPI2.0 /dev/capi support` » n'est pas documenté et ne sera pas décrite ici. La réponse recommandée est '`N`'.

L'option « `CAPI2.0 filesystem support` » n'est pas encore documentée et ne sera pas décrite dans ce document.

L'option « `CAPI2.0 capidrv interface support` » n'est pas documentée et ne sera pas décrite ici. La réponse recommandée est '`N`'.

Les options qui suivent permettent de prendre en charge les cartes AVM. La réponse recommandée est '`N`'.

L'option « `Hypercope HYSDN cards (Champ, Ergo, Metro) support (module)` » permet de prendre en charge les cartes ISDN actives d'Hypercope. La réponse recommandée est '`N`'.

L'option « `HYSDN CAPI2.0 support` » permet de gérer l'interface de programmation CAPI 2.0 pour les cartes Hypercope. La réponse recommandée est '`Y`'.

9.3.40. Menu « Old CD-ROM drivers (not SCSI, not IDE) »

L'option « `Support non-SCSI/IDE/ATAPI CDROM drives` » permet d'activer la gestion des lecteurs de CD-ROM non SCSI et non IDE. Ce sont essentiellement des vieux lecteurs de CD-ROM, parfois connectés sur une carte son par une interface propriétaire. Les options qui suivent permettent de choisir le driver qui correspond au matériel dont vous disposez. La réponse recommandée est '`N`'.

9.3.41. Menu « Input Core Support »

L'option « `Input core support` » permet d'activer un nouveau mécanisme de gestion des périphériques de saisie et d'entrée interactives. Ce mécanisme permettra à terme de gérer tous les périphériques d'entrée de manière uniforme, et donc de simplifier la gestion des données provenant de l'utilisateur. Les périphériques déjà capables d'utiliser ce nouveau mécanisme comprennent les joysticks et certains périphériques USB. Vous devez activer cette fonctionnalité si vous désirez connecter un joystick, un clavier ou une souris USB sur votre ordinateur. La réponse recommandée est '`N`' pour les gens qui ne disposent pas de tels périphériques.

L'option « `Keyboard support` » permet de prendre en charge les claviers USB en redirigeant les données provenant de ces claviers vers le traitement classique du clavier de Linux. Si vous disposez d'un tel clavier, il n'y a normalement rien à faire pour l'utiliser, puisque le BIOS assure la compatibilité ascendante des claviers USB connectés sur les ports USB de la carte mère. Cependant, cette technique ne fonctionne pas pour les claviers connectés sur des cartes USB filles, et vous pouvez avoir besoin d'activer la gestion de ces claviers ici. La réponse recommandée est '`N`', sauf si vous désirez connecter un clavier USB sur votre ordinateur et qu'il ne fonctionne pas nativement. Notez que cette option est incompatible avec l'option « `USB HIDBP Keyboard (basic) support` » du menu « `USB support` », qui permet de gérer les claviers USB d'une manière plus légère afin de réduire la taille du noyau dans les systèmes embarqués.

De la même manière, l'option « `Mouse support` » permet de prendre en charge les souris USB en simulant une souris PS/2 classique, via le fichier spécial de périphérique `/dev/input/mice`, de codes majeur et mineur 13 et 63. Ce driver permet également d'accéder aux souris USB via les fichiers spéciaux de périphériques `/dev/input/mouseN`, où `N` est le numéro de la souris, et dont les numéros de codes majeur et mineurs sont respectivement 13 et 32 à 63. Ce driver est également incompatible avec le driver simplifié que l'on peut activer avec l'option « `USB HIDBP Mouse (basic) support` » du menu « `USB support` ». La réponse recommandée est '`N`', sauf pour ceux qui disposent d'une souris USB.

Les options « `Horizontal screen resolution` » et « `Vertical screen resolution` » permettent de définir la résolution horizontalement et verticalement de l'écran, afin de permettre l'utilisation d'une tablette de digitalisation USB comme une souris. Ces deux données permettent de faire la conversion entre les coordonnées de la tablette et les coordonnées de l'écran. La réponse recommandée est '`N`'.

L'option « `Joystick support` » active la prise en charge des joysticks USB et permet aux applications d'y accéder au travers des fichiers spéciaux de périphériques `/dev/input/jsN`, où `N` est le numéro du joystick, et dont les numéros de codes majeur et mineurs sont respectivement 13 et 0 à 31. Notez que cette option n'est pas nécessaire pour la prise en charge des joystick non-USB, bien que l'activation de l'option « `Input core support` » restent obligatoire. La réponse recommandée est '`N`'.

L'option « `Event interface support` » active la gestion des fichiers spéciaux `/dev/input/eventN`, où `N` est le numéro du périphérique, qui permettent de lire les événements provenant des périphériques d'entrée USB de manière générique. Ces fichiers peuvent être utilisés par plusieurs périphériques distinct, et permettront à terme de connecter plusieurs périphériques d'entrée sur la même machine, tout en permettant une gestion uniforme de la part des applications. Actuellement, très peu de programmes utilisent cette fonctionnalité, et il est encore rare de voir des machines à plusieurs claviers et plusieurs souris. La réponse recommandée est donc '`N`'. À terme cependant, cette fonctionnalité sera certainement utilisée pour permettre l'utilisation d'une même machine par plusieurs personnes (il suffit pour cela de développer un driver pour les serveurs X de XFree86 capable de lire les données provenant de ces fichiers spéciaux de périphériques).

9.3.42. Menu « Character devices »

L'option « `Virtual terminal` » permet d'activer la gestion des terminaux virtuels, accessibles avec les combinaisons de touche `ALT+Fx` (ou `CTRL+ALT+Fx` sous XWindow). Il faut au moins un terminal virtuel pour pouvoir utiliser le clavier et l'écran de la machine, vous devez donc répondre '`Y`' à cette question.

L'option « `Support for console on virtual terminal` » permet de placer la console système (les flux standard du noyau, et le login en mode mono utilisateur) sur un terminal virtuel. Si vous répondez '`N`' à cette question, la console ne sera pas accessible avec le clavier et l'écran, aussi faut-il répondre '`Y`' à cette question.

L'option « `Standard/generic (8250/16550 and compatible UARTs) serial support` » permet d'activer la gestion des ports série classiques. Vous pouvez vous en passer si vous n'avez ni modem ni souris série, et si vous ne désirez pas utiliser le port série du tout. Il est recommandé de répondre '`Y`' à cette question.

L'option « `Support for console on serial port` » permet d'activer la redirection de la console système sur un port série. En général, la console est redirigée vers un terminal virtuel, aussi la réponse recommandée à cette question est-elle '`N`'.

L'option « `Extended dumb serial driver options` » permet d'activer les options de configurations étendues pour les ports série classiques. La réponse recommandée est '`N`'.

L'option « `Support more than 4 serial ports` » permet d'activer la gestion de ports série additionnels. La réponse recommandée est '`N`'.

L'option « `Support for sharing serial interrupts` » permet d'activer la gestion des interruptions de plusieurs ports série d'une même carte par un seul canal d'interruption. Il faut que la carte série gère le partage des interruptions pour pouvoir utiliser cette option. La réponse recommandée est '`N`'.

L'option « `Autodetect IRQ on standard ports (unsafe)` » permet de demander au noyau de tenter une détection automatique de la ligne d'interruption utilisée par les ports série. Cette fonctionnalité n'est pas très sûre et la réponse recommandée est '`N`'.

L'option « `Support special multiport boards` » permet d'activer de certaines cartes série, qui peuvent indiquer quel port série est en attente de traitement de la part du driver. La réponse recommandée est '`N`'.

L'option « `Support the Bell Technologies HUB6 card` » permet d'activer la gestion des cartes HUB6. La réponse recommandée est '`N`'.

L'option « `Non-standard serial port support` » permet d'activer les options correspondantes à des ports série non standard. Les options qui suivent permettent de configurer les drivers correspondants. La réponse recommandée est '`N`'.

L'option « `Unix98 PTY support` » permet d'activer les pseudo terminaux compatibles Unix 98. Les pseudo terminaux sont des composants logiciels permettant d'émuler des terminaux. Il y a deux composants par pseudo terminal : la partie maître est le programme qui cherche à accéder au pseudo terminal, et la partie esclave est le programme qui simule le terminal physique. Les composants maîtres peuvent ouvrir le pseudo terminal par l'intermédiaire des fichiers spéciaux `/dev/ptyxx`. En revanche, il existe deux protocoles de nommage des fichiers spéciaux utilisés par les composants esclaves. L'ancienne méthode utilisait les fichiers spéciaux `/dev/ttyxx`, mais elle a été remplacée par un autre protocole, dans lequel le composant esclave doit d'abord ouvrir le fichier spécial `/dev/ptmx`, qui lui donne le numéro `n` de pseudo terminal qu'il doit simuler, et il ouvre ensuite le fichier `/dev/pts/n`. Ce dernier fichier spécial est créé à la volée, grâce à un système de fichier virtuel. Pour que ce mécanisme fonctionne, il faut que le système de fichiers `/dev/pts/` ait été également choisi parmi les systèmes de fichiers, et qu'il soit monté (voir la configuration du fichier `/etc/fstab`). Si vous répondez '`Y`', l'option suivante vous permet de choisir le nombre de pseudo terminaux que vous désirez que le système gère. La valeur par défaut est 256. La réponse recommandée est '`Y`'.

L'option « `Parallel printer support` » permet d'activer la gestion des imprimantes connectées sur port parallèle. Le fait d'activer cette option n'empêche pas d'utiliser le port parallèle pour d'autres périphériques. Il est recommandé d'activer cette option sous forme de module parce qu'ainsi, les fonctionnalités d'impressions ne seront chargées que lorsque c'est nécessaire. La réponse recommandée est donc '`M`'.

L'option « `Support for console on line printer` » permet de rediriger les messages destinés à la console sur une imprimante connectée au port parallèle. Ces messages sont alors imprimés au fil de l'eau. La réponse recommandée est '`N`'.

L'option « `Support for user-space parallel port device drivers` » active la gestion du fichier spécial de périphérique `/dev/parport`, au travers duquel les programmes peuvent accéder de manière uniforme au port parallèle. Cette fonctionnalité n'est pas nécessaire pour utiliser une imprimante ou des périphériques ATAPI connectés sur port parallèle, aussi la réponse recommandée est-elle '`N`'.

L'option « `QIC-02 tape support` » permet d'activer la gestion des lecteurs de cassettes non SCSI du type QIC. L'option suivante permet d'indiquer si la configuration dynamique du lecteur doit être utilisée. La réponse recommandée pour ces deux questions est '`N`'.

L'option « `Intel i8x0 Random Number Generator support` » active la prise en charge des générateurs de nombres aléatoires présents sur les cartes mères à base de chipsets i8x0 d'Intel. La réponse recommandée est '`N`'.

L'option « `/dev/nvram support` » permet d'activer l'accès à la mémoire non volatile de l'horloge temps réel. Cette mémoire peut être accédée par l'intermédiaire du fichier spécial de périphérique `/dev/nvram`, de code majeur 10 et de code mineur 144. La réponse recommandée est '`N`'.

L'option « `Enhanced Real Time Clock Support` » permet d'activer l'accès au compteurs de l'horloge temps réel par l'intermédiaire du fichier spécial de périphérique `/dev/rtc`, de code majeur 10 et de code mineur 135. Cette option doit être activée si vous disposez d'une machine multiprocesseur. La réponse recommandée est '`N`'.

L'option « `Double Talk PC internal speech card support` » permet d'activer la gestion du synthétiseur de voix Double Talk. La réponse recommandée est '`N`'.

L'option « `Siemens R3964 line discipline` » active la gestion des synchronisations entre les périphériques utilisant le protocole de communication R3964 de Siemens. La réponse recommandée est '`N`'.

L'option « `Applicom intelligent fieldbus card support` » active la gestion des cartes Applicom « intelligent fieldbus ». La réponse recommandée est '`N`'.

L'option « `/dev/agpgart support (AGP support)` » active la gestion des transferts de données accélérés par le bus AGP pour les drivers de cartes graphiques. Ces transferts peuvent être pilotés au travers du fichier spécial de périphérique `/dev/agpgart`, de code majeur 10 et de code mineur 175. Cette fonctionnalité n'est pas nécessaire pour le bon fonctionnement des cartes graphiques AGP, mais elle permet d'accroître les performances des drivers 3D. Elle est également indispensable pour permettre l'accès au registre MTRR du processeur permettant de contrôler le bus AGP. La réponse recommandée est '`Y`' si vous disposez d'une carte graphique AGP, et '`N`' sinon. Les options qui suivent permettent de sélectionner le type de chipset utilisé par la carte mère, afin de déterminer la manière dont le port AGP fonctionne. Si vous avez activé cette fonctionnalité, vous devez choisir l'option correspondant à votre chipset également.

L'option « `Direct Rendering Manager (XFree86 DRI support)` » permet d'activer le support de l'architecture DRI permettant aux pilotes graphiques d'accéder directement aux ressources des cartes graphiques. Cette fonctionnalité est à la base du fonctionnement de l'architecture 3D de Linux. Les options suivantes permettent de sélectionner le type de carte graphique installé sur le système. La réponse recommandée est '`N`', sauf si vous disposez d'une de ces cartes graphiques. Dans ce

cas, vous devriez certainement utiliser XFree86 4.0.2, afin de bénéficier des accélérations 3D de ces cartes graphiques.

9.3.43. Menu « I2C support »

L'option « `I2C support` » permet d'activer la gestion du protocole de communication I2C. Ce protocole de communicatione est utilisé par beaucoup de micro-contrôleurs, et peut être nécessaire pour accéder à certaines fonctionnalités. En particulier, les cartes d'acquisition TV basées sur les puces électroniques Bt848 nécessitent cette fonctionnalité pour être utilisable. La réponse recommandée est '`N`', sauf si vous disposez d'une telle carte. Dans ce cas, la réponse recommandée est '`M`'.

L'option « `I2C bit-banging interfaces` » active la gestion des adaptateurs « bit-banging ». Cette option est nécessaire pour faire fonctionner les cartes d'acquisition TV basées sur les puces électroniques Bt848. La réponse recommandée est '`N`', sauf si vous disposez d'une telle carte. Les options suivantes activent la gestion de périphériques utilisant cette interface. Il n'est pas nécessaire des les activer pour faire fonctionner les cartes d'acquisition TV à base de Bt848. La réponse recommandée pour ces options est '`N`'.

L'option « `I2C PCF 8584 interfaces` » active la gestion des adaptateurs PCF. Les suivantes permettent d'activer les périphériques utilisant cette interface. Les réponses recommandées à ces questions sont '`N`'.

L'option « `I2C device interface` » active la gestion des fichiers spéciaux de périphériques `/dev/i2c-*`. La réponse recommandée est '`N`'.

9.3.44. Menu « Mice »

L'option « `Bus Mouse Support` » active la prise en charge des souris bus. Ces souris sont des souris connectées sur des bus spécifiques, comme par exemple sur une carte graphique. Cette option permet d'inclure le driver générique de ce type de souris, les drivers spécifiques peuvent être sélectionnés avec les options suivantes. La réponse recommandée est '`N`', sauf si vous disposez d'une telle souris.

L'option « `ATIXL busmouse support` » permet d'activer la gestion des souris bus connectées sur certaines cartes graphiques ATI. La réponse recommandée est '`N`'.

L'option « `Logitech busmouse support` » permet de gérer les souris bus Logitech. Attention, les souris Logitech connectées sur port PS/2 ne sont pas des souris bus et ne sont pas gérées par ce driver. La réponse recommandée est '`N`'.

L'option « `Microsoft busmouse support` » permet de gérer les souris bus Microsoft. Attention, les souris Microsoft connectées sur port PS/2 ne sont pas des souris bus et ne sont pas gérées par ce driver. La réponse recommandée est '`N`'.

L'option « `Mouse Support (not serial and bus mice)` » permet d'activer la gestion des souris qui ne sont connectées ni au port série, ni sur un bus souris. Si vous disposez d'une souris série ou d'une souris bus, vous devez répondre '`N`' à cette question. Sinon, la réponse recommandée est '`Y`'.

L'option « `PS/2 mouse (aka "auxiliary device") support` » permet de gérer toutes les souris connectées sur un port PS/2, quelles que soit leur marque. La plupart des souris vendues actuellement son de ce type. La réponse recommandée est donc '`Y`'.

L'option « `C&T 82C710 mouse port support (as on TI Travelmate)` » permet de gérer les souris pour Travelmate. Ces souris sont des souris PS/2 particulières à ces ordinateurs. Il est recommandé d'essayer le driver générique PS/2 avant de choisir celui-ci, aussi faut-il répondre '`N`' à cette question en général.

L'option « `PC110 digitizer pad support` » permet d'activer la gestion de l'émulation souris pour les PC110 digitizer. La réponse recommandée est '`N`'.

9.3.45. Menu « Joystick support »

L'option « `Jostick support` » permet d'activer les options de gestion des joysticks. Les options suivantes correspondent aux drivers des différents types de joysticks supportés. La réponse recommandée est '`N`'.

9.3.46. Menu « Watchdog cards »

L'option « `Watchdog Timer Support` » permet d'activer la détection des blocages systèmes grâce à un chien de garde. Le chien de garde exige un accès en écriture sur le fichier spécial de périphérique `/dev/watchdog`, de code majeur 10 et de code mineur 130, au moins une fois par minute. Si cette condition n'est pas vérifiée, la machine est redémarrée automatiquement. La gestion de cette fonctionnalité peut être réalisée matériellement grâce à une carte spéciale, ou logiciellement. La réponse recommandée est '`N`'.

L'option « `Disable watchdog shutdown on close` » permet de maintenir l'activité du chien de garde même si le processus qui le réveille régulièrement ferme le fichier `/dev/watchdog`. Ce n'est pas le cas si la réponse '`N`' est donnée, dans ce cas, le chien de garde ne fonctionnera plus en cas d'arrêt du processus de surveillance. La réponse recommandée est '`N`'.

L'option « `Software Watchdog` » permet d'activer la gestion du chien de garde logiciel. La réponse recommandée est '`N`'.

L'option « `WDT Watchdog timer` » permet d'activer la gestion des cartes WDT. Ces cartes ne pouvant pas être configurées automatiquement, vous devrez spécifier manuellement le port et la ligne d'interruption à l'aide du paramètre du noyau `wdt` lors du démarrage du système. Si vous répondez '`Y`' ou '`M`', vous pourrez spécifier les fonctionnalités de cette carte à l'aide des options « `WDT501 features` » et « `Fan Tachometer` ». La réponse recommandée est '`N`'.

L'option « `WDT PCI Watchdog timer` » permet d'activer la gestion des cartes WDT PCI. La réponse recommandée est '`N`'.

Les options « `WDT501 features` » et « `Fan Tachometer` » permettent de spécifier les paramètres du système surveillés par les cartes WDT : température et vitesse du ventilateur. La réponse recommandée est '`N`'.

L'option « `Berkshire Products PC Watchdog` » permet d'activer la gestion des cartes Berkshire. La réponse recommandée est '`N`'.

L'option « `Acquire SBC Watchdog Timer` » permet d'activer la gestion des chiens de garde des PSC-6x86 Single Board Computer d'Acquire Inc. La réponse recommandée est '`N`'.

L'option « `SBC-60XX Watchdog Timer` » permet d'activer la gestion des chiens de garde des ordinateurs disposant d'une carte mère SBC 6010. La réponse recommandée est '`N`'.

L'option « `Mixcom Watchdog` » permet d'activer la gestion des chiens de garde Mixcom. La réponse recommandée est '`N`'.

L'option « `Intel i810 TCO timer / Watchdog` » permet d'activer la gestion des chiens de garde matériels intégrés dans les chipsets i810 et i815 d'Intel. Ces chiens de garde peuvent surveiller l'activité du système régulièrement et contrôler sa température. La réponse recommandée est '`N`'.

9.3.47. Menu « Ftape, the floppy tape device driver »

L'option « `Ftape (QIC-80/Travan) support` » permet d'activer les options de gestion des lecteurs de bande connectés sur le contrôleur de disquettes. Si vous avez un tel périphérique, répondez '`Y`' ou '`M`'. Sinon, répondez '`N`'.

L'option « `Zftape, the VFS interface` » permet d'inclure la gestion des systèmes de fichiers virtuels dans le driver de ftape. Il faut impérativement activer cette option, faute de quoi le driver sera inutilisable. Répondez par '`Y`' à cette question si vous avez activé la fonctionnalité ftape.

L'option « `Default block size` » permet de spécifier la taille par défaut des blocs utilisés par les programmes d'archivage. La valeur 10240 correspond à la taille des blocs utilisée par GNU tar, c'est la valeur recommandée.

L'option « `Number of ftape buffers (EXPERIMENTAL)` » est expérimentale et il est très déconseillé de la modifier. La valeur par défaut est 3.

L'option « `Enable procfs status report (+2kb)` » permet de générer un répertoire `ftape/` dans le système de fichiers virtuel `/proc/`. Ce répertoire contient des informations concernant l'état courant du driver ftape. La réponse recommandée à cette question est '`N`'.

L'option « `Debugging output` » permet de fixer la quantité de traces que le driver ftape génère. La valeur recommandée est « `Normal` ».

L'option « `Floppy tape controllers` » permet de sélectionner le type de contrôleur de disquettes. La valeur recommandée est « `Standard` ». Si vous choisissez un autre type de contrôleur, vous devrez spécifier les paramètres matériels pour ce contrôleur dans les trois options qui suivent.

L'option « `Default FIFO threshold (EXPERIMENTAL)` » est expérimentale et ne doit pas être modifiée. La valeur recommandée est 8.

L'option « `Maximal data rate to use (EXPERIMENTAL)` » permet de réduire la vitesse maximale de transfert utilisé par le driver ftape. Cette option est expérimentale et ne doit pas être modifiée. La valeur par défaut est 2000.

9.3.48. Menu « PCMCIA character device support »

Les options de ce menu permettent d'activer la gestion des cartes PCMCIA de type série, comme par exemple les modems, les ports série et les cartes réseau intégrant un modem. La réponse recommandée à ces options est 'N'.

9.3.49. Menu « Multimedia devices »

L'option « Video For Linux » permet d'activer les options de gestion de la vidéo et de radio sous Linux. Il faut activer cette option pour accéder aux deux menus suivants, qui permettent de choisir les drivers adaptés au type de matériel vidéo installé et les options de configuration pour certains de ces drivers. La réponse recommandée est 'N'.

9.3.50. Menu « Video For Linux »

L'option « V4L information in proc filesystem » permet d'inclure des informations complémentaire sur l'état de l'interface Vidéo pour Linux dans le système de fichiers /proc/. La réponse recommandée est 'Y'.

L'option « I2C on parallel port » permet d'utiliser le port parallèle comme un interface I2C pour les contrôleurs vidéo qui reconnaissent ce protocole. La réponse recommandée est 'N'.

Les options qui suivent permettent de choisir les drivers pour les différents types de cartes d'acquisition vidéo. Vous devez activer la fonctionnalité correspondante au matériel dont vous disposez. En général, il est recommandé d'utiliser les drivers sous forme de modules, car certains de ces drivers ne sont pas capables de faire la distinction entre les différents modèles de cartes qui utilisent la même électronique, et il faut leur communiquer ces informations sous la forme de paramètres lors du chargement des modules. C'est en particulier le cas pour les cartes vidéo basées sur la puce électronique Bt848 et les puces qui en sont dérivées.

9.3.51. Menu « Radio Adapters »

Les options proposées par ce menu vous permettront d'activer les drivers pour les différentes cartes radio supportées par Linux. Vous devez activer le driver correspondant à votre matériel, et éventuellement spécifier les paramètres matériels de cette carte. La réponse recommandée à ces questions est 'N'.

9.3.52. Menu « File systems »

L'option « Quota support » permet d'activer la gestion des quotas de disque utilisé par utilisateur. Cette fonctionnalité n'est disponible que pour les systèmes de fichiers EXT2. La réponse recommandée est 'N'.

L'option « `Kernel automounter support` » permet d'effectuer le montage des disques NFS automatiquement (c'est à dire à la demande) au niveau du noyau. La réponse recommandée est '`N`'.

L'option « `Kernel automounter version 4 support (also supports v3)` » permet d'activer la nouvelle version du montage automatique des disques NFS. La réponse recommandée est '`N`'.

Les options qui suivent permettent de prendre en compte les systèmes de fichiers de différents systèmes d'exploitation. Parmi ces systèmes, certains sont encore en cours de développement et ne fonctionnent pas très bien. Cela signifie qu'il est très déconseillé d'utiliser ces drivers pour écrire des données sur ces systèmes de fichiers : de très grosses pertes de données peuvent s'ensuivre. En particulier, il ne faut surtout pas utiliser le système de fichier NTFS en écriture : les développeurs de cette partie du noyau sont, encore actuellement, certains qu'il détruira le système de fichiers. En revanche, les systèmes de fichiers qui sont complètement gérés peuvent être utilisés sans risques. Cependant, ils peuvent ne pas être assez puissants pour stocker les informations de fichiers nécessaires à tout système Unix. Dans ce cas, le système de fichiers ne peut être utilisé que pour stocker des données d'applications, pas le système lui-même. C'est en particulier le cas de la FAT. Afin de corriger les limitations de ce système de fichiers, une extension nommée UMSDOS a été développée. Cette extension stocke dans chaque répertoire un fichier caché décrivant les droits Unix, et rend ainsi la FAT exploitable sous Linux. Cette fonctionnalité est très intéressante si l'on veut installer Linux sur un disque déjà formaté en FAT. Il faut prendre garde cependant au fait que les performances seront alors déplorables (les accès disques sont plus lents dans un facteur 2 à 3). Le problème de performances se pose d'ailleurs pour tous les systèmes de fichiers qui ont été intégré à Linux mais qui ne sont pas natifs. Ainsi, même sans utiliser UMSDOS, la FAT32 est deux fois plus lente que les systèmes de fichiers natifs. Le système de fichiers le plus utilisé actuellement sous Linux est EXT2, qui offre à la fois la sécurité, les fonctionnalités et les performances. Il est fortement recommandé de l'utiliser dès que l'on peut se passer des autres systèmes de fichiers.

Il est impératif de compiler le système de fichiers de la partition root dans le noyau. Si cela n'est pas fait, le noyau ne pourra pas monter la partition root et se terminera en affichant le message « kernel panic ». Il ne faut pas compiler ce système de fichiers en tant que module, pour les mêmes raison. La réponse recommandée pour l'option « `Second extended fs support` » est donc '`Y`' et rien d'autre, et on installera la partition root sur un tel système de fichier.

Certains systèmes de fichiers nécessitent le support de systèmes de fichiers de base. En particulier, les systèmes de fichiers VFAT (qui gère les FAT32), MSDOS fs (qui gère les partitions DOS 12 et 16 bits) et UMSDOS exigent tous trois le support du système de fichiers DOS FAT en général. Pour utiliser ces systèmes de fichiers, on devra donc activer l'option « `DOS FAT fs support` ». De même, pour lire les CD-ROM au format Juliette (l'extension Microsoft au format ISO 9660 pour supporter les noms longs sous Windows), il faudra activer l'option « `ISO 9660 CDROM file system support` ».De toutes façons, il est fortement recommandé de gérer ce système de fichiers si l'on veut utiliser des CD-ROMs.

Trois systèmes de fichiers sont virtuels. Ils ne correspondent à aucun support physique, et leur arborescence est créée uniquement en mémoire, à la volée, par le noyau. Il s'agit du système de fichiers `/proc/` (option « `/proc file system support` »), qui fournit des informations dynamiquement sur l'état du système, du système de fichiers `/dev/` (option « `/dev file system sup-`

port (EXPERIMENTAL) »), qui permet de gérer les fichiers spéciaux de périphériques à la volée, et du système de fichiers /dev/pts/ (option « /dev/pts file system for Unix98 PTYs »), qui permet de créer des fichiers spéciaux de périphériques à la demande pour les pseudo terminaux. Il est fortement recommandé d'activer la gestion des systèmes de fichiers /proc/ et /dev/pts/, car ils utilisés par beaucoup de programmes. Le système de fichiers /dev/ pourra être utilisé sur les systèmes embarqués ou les disquettes de démarrage, pour lesquels la taille prise par le répertoire /dev/ peut être gênante. Dans ce cas, on aura intérêt à réaliser le montage automatique de ce système de fichiers à l'aide de l'option « Automatically mount at boot ».

9.3.53. Menu « Network File Systems »

L'option « Coda file system support (advanced network fs) » permet d'activer le support du système de fichiers réseau Coda, qui donne accès aux périphériques par le réseau comme s'ils étaient branchés sur la machine locale. Cette option active les fonctionnalités clientes au niveau du noyau, mais il faut également des outils complémentaires pour implémenter les fonctions serveurs. La réponse recommandée est 'N'.

L'option « NFS file system support » permet d'activer le support NFS classique en tant que machine cliente. NFS est moins puissant que Coda, mais est cependant le plus répandu. Il lui est donc encore préférable pour l'instant. La réponse recommandée est 'Y'.

L'option « Provide NFSv3 client support » permet d'activer le support de la version 3 du protocole NFS en tant que machine cliente. La réponse recommandée est 'N'.

L'option « Root file system on NFS » permet d'indiquer au noyau que le système de fichiers root est placé sur un serveur NFS. Dans ce cas, le noyau se connectera automatiquement à ce serveur pour monter le système de fichiers root. Pour réaliser ce type de montage, il faut avoir répondu 'Y' à la question « IP : kernel level autoconfiguration », afin que les couches réseaux soient configurées au niveau du noyau lors du démarrage du système. La réponse recommandée est 'N'.

L'option « NFS server support » permet d'activer le support NFS en tant que machine serveur. La réponse recommandée est 'N'.

L'option « Provide NFSv3 server support » permet d'activer le support de la version 3 du protocole NFS en tant que serveur. La réponser recommandée est 'N'.

L'option « SMB file system support (to mount WfW shares, etc.) » permet d'accéder aux répertoires partagés par les machines fonctionnant sous Windows. Cette fonctionnalité n'est possible que si les postes Windows utilisent TCP/IP comme protocole de réseau de base, et non NetBIOS. La réponse recommandée est 'Y'.

Les « Use a default NLS » et « Default Remote NLS Option » permettent de spécifier une page de code par défaut pour lire les noms de fichiers et de répertoire partagées par le serveur. La réponse recommandée est 'N'.

L'option « NCP file system support (to mount NetWare volumes) » permet d'accéder aux volumes NetWare. Cette fonctionnalité nécessite la gestion d'IPX au niveau de la machine Linux. On notera qu'il est inutile d'activer cette fonctionnalité si l'on désire faire en sorte que la machine Linux

soit serveur de fichier Novell. Les options qui suivent permettent de paramétrer les accès aux volumes NetWare (sécurité, lock de fichiers, droits d'accès, etc…). La réponse recommandée est 'N'.

9.3.54. Menu « Partition Types »

Quelques systèmes utilisent un format différents pour les tables de partitions que le format utilisé par le BIOS des PC. Pour lire les données stockées sur les disques partitionnés par ces systèmes sur des machines d'architecture différente, vous devez activer une gestion spéciale des tables de partitions. Les options de ce menu permettent d'activer la gestion des tables de partitions de ces architectures ces systèmes. La réponse recommandée à ces questions est 'N'.

9.3.55. Menu « Native Language Support »

Les options fournies ici permettent de choisir les pages de codes à utiliser pour les jeux de caractères dans les systèmes de fichiers. Ces pages de codes doivent être choisies en fonction de la langue d'installation du système. Les pages de codes recommandées pour un système français sont les suivantes :

- « `Codepage 437 (United States, Canada)` », indispensable en raison de la majorité des programmes provenant des États Unis ;
- « `Codepage 850 (Europe)` », pour les caractères accentués européens ;
- « `NLS ISO 8859-1 (Latin 1; Western European Languages)` » pour les caractères accentués européens sur les CDROMs ;
- « `NLS ISO 8859-15 (Latin 9; Western European Languages with Euro)` » pour les caractères accentués européens sur les CDROMs, y compris quelques caractères additionnels non gérés par la page de code ISO 8859-1.

Pour toutes les autres pages de codes, la réponse recommandée est 'N'.

L'option « `Default NLS Option` » permet de choisir la page de code par défaut à utiliser parmi celles qui ont été choisies. La réponse recommandée est « `iso8859-1` ».

9.3.56. Menu « Console drivers »

L'option « `VGA text console` » permet d'utiliser les modes textes VGA pour l'affichage en mode texte. La réponse à cette question est 'Y'.

L'option « `Video mode selection support` » permet de choisir le mode texte à utiliser au démarrage de Linux. Ce mode peut être indiqué grâce à un paramètre passé au noyau lors du démarrage. La réponse recommandée est 'Y'.

L'option « `MDA text console (dual-headed) (EXPERIMENTAL)` » permet d'activer le support de l'affichage multiple réalisable à l'aide d'une carte MDA et d'une carte VGA. Cette option ne doit être choisie que si la carte MDA est la carte d'affichage principale. La réponse recommandée est '`N`'.

9.3.57. Menu « Frame-buffer support »

L'option « `Support for frame buffer devices (EXPERIMENTAL)` » permet d'activer le support des cartes graphiques par l'intermédiaire d'une interface unifiée et d'un buffer vidéo nommé « frame buffer », accessible via les fichiers spéciaux `/dev/fb*`. Cette interface permet aux programmes d'accéder aux fonctionnalités des cartes graphiques de manière portable. Le support de cette fonctionnalité n'est en général pas nécessaire pour les PC, puisque beaucoup de serveurs X ont été développés pour cette plate-forme, et que dans le pire des cas, un serveur X basé sur le standard VESA 2.0 est disponible. Cependant, pour les autres plates-formes, ou si vous disposez d'une carte graphique exotique, il peut être utile d'activer cette fonctionnalité et d'utiliser un serveur X basé sur cette interface. Les options qui suivent permettent d'inclure les drivers pour différentes cartes graphiques. Vous trouverez de plus amples informations sur ces options dans la huitième partie de ce document. La réponse recommandée est '`N`'.

L'option « `Virtual Frame Buffer support (ONLY FOR TESTING!)` » permet d'activer la gestion d'une carte graphique virtuelle en mémoire seulement. Ce driver consomme énormément de ressources et ne doit être utilisé que pour tester des applications basées sur cette interface. La réponse recommandée est '`N`'.

L'option « `Advanced low level driver options` » permet d'activer les options permettant de fixer certains paramètres bas niveau. Ces paramètres peuvent être fixés grâce aux options suivantes. En particulier, les formats de pixels gérés, ainsi que les polices de caractères à utiliser peuvent être précisés. Vous trouverez de plus amples informations sur ces options dans le Chapitre 10. La réponse recommandée est '`N`'.

9.3.58. Menu « Sound »

L'option « `Sound card support` » permet d'activer la gestion des cartes son. Si vous ne disposez pas de carte son, choisissez la réponse '`N`', sinon, activez cette fonctionnalité.

Les options qui suivent permettent de choisir les drivers pour les cartes son non standard. Ces drivers gère les cartes sons correctement, mais ne fournissent pas exactement la même interface de programmation que le standard actuel sous Linux qu'est « OSS » (abréviation de l'anglais « Open Sound System »). Certaines fonctionnalités ne seront donc pas accessibles de la manière standard avec ces cartes.

Vient ensuite l'option « `OSS sound modules` », qui permet d'activer la sélection des drivers compatibles avec l'interface OSS. Les options qui suivent celle-ci permettent de choisir et de configurer les drivers pour les cartes son compatibles. Il est recommandé d'utiliser les drivers OSS si possible. Notez qu'une certain nombre de drivers ont également été développés dans le cadre de l'architecture ALSA (abréviation de l'anglais « Advanced Linux Sound Architecture »). Si vous ne trouvez pas de driver approprié à votre matériel ici, vous pouvez essayer d'installer ces drivers.

Notez enfin que le driver de son pour les cartes d'acquisition TV basées sur la puce Bt848 se situe parmi les drivers OSS (option « `TV card (bt848) mixer support` ». Vous devez donc l'intégrer sous forme de module si vous disposez d'une telle carte.

9.3.59. Menu « USB support »

L'option « `Support for USB` » permet de prendre en charge les périphériques USB sous Linux. La réponse recommandée est 'Y'.

L'option « `USB verbose debug message` » permet de demander aux drivers USB d'envoyer des informations de débogage dans les fichiers de traces du système. La réponse recommandée est 'N'.

L'option « `Preliminary USB device filesystem` » active la gestion des informations sur les ports USB dans le système de fichiers virtuels `/proc/`. Ces informations comprennent en particulier la liste des périphériques connectés sur le bus USB. La réponse recommandée est 'Y'.

L'option « `Enforce USB bandwidth allocation (EXPERIMENTAL)` active la gestion de la bande passante du bus USB entre les différents périphériques qui y sont connectés. Cette option est expérimentale, et la réponse recommandées est 'N'.

Les trois options suivantes (« `UHCI (Intel PIIX4, VIA, ...) support` », « `UHCI Alternate Driver (JE) support` » et « `OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support` » permettent d'inclure les drivers pour les différents types de contrôleurs USB. On recense deux différents types de contrôleurs USB : les contrôleurs UHCI (abréviation de l'anglais « Universal Host Controller Interface ») et les contrôleurs OHCI (abréviation de « Open Host Controller Interface »). Les contrôleurs UHCI sont fabriqués par Intel et VIA essentiellement, alors que les contrôleurs OHCI sont présents sur les chipsets non Intel (Compaq, SiS, Aladdin). Vous devez choisir le driver approprié au chipset de votre carte mère. Pour les contrôleurs UHCI, vous avez le choix entre deux drivers.

L'option « `USB Audio support` » permet de prendre en charge les périphériques audio connectés sur le port USB, comme des hauts-parleurs par exemple. La réponse recommandée est 'N'.

L'option « `USB Bluetooth support (EXPERIMENTAL)` » active la gestion des périphériques USB Bluetooth (capables d'effectuer des transmissions sans fil). La réponse recommandée est 'N'.

L'option « `USB Mass Storage support` » active la gestion des périphériques USB de masse (lecteurs de CD, etc...). La réponse recommandée est 'N'.

L'option « `USB Mass Storage verbose debug` » permet de demander au driver des périphériques USB de masse de générer des messages de débogage dans les fichiers de trace du système. La réponse recommandée est 'N'.

L'option « `Freecom USB/ATAPI Bridge support` » permet de gérer les passerelles entre le bus USB et les périphériques ATAPI connectés sur ce bus. La réponse recommandée est 'N'.

L'option « `USB Modem (CDC ACM) support` » permet de prendre en charge les modems analogiques et Numéris utilisant l'interface CDC ACM (abréviation de l'anglais « Communication Device Class Abstract Control Model ») connectés sur le port USB. La réponse recommandée est 'N'.

L'option « `USB Printer support` » active la gestion des imprimantes connectées sur le port USB. La réponse recommandée est '`N`'.

L'option « `USB Human Interface Device (full HID) support` » permet d'activer la prise en charge des périphériques d'entrée et de saisie tels que les claviers, souris et tablettes de digitalisation. Cette option n'est activable que si l'on a activé également la fonctionnalité « `Input core support` » du menu « `Input core support` ». Ce driver gère complètement les périphériques d'entrée USB, et est incompatible avec les drivers simplifiés pour le clavier et la souris, que l'on peut activer avec les options « `USB HIDBP Keyboard (basic) support` » et « `USB HIDBP Mouse (basic) support` ». La réponse recommandée est '`N`'.

L'option « `USB HIDBP Keyboard (basic) support` » permet d'activer un driver simplifié pour les claviers USB. Ce driver peut être utilisé pour alléger le noyau dans les systèmes embarqués, mais ne gère pas toutes les touches des claviers étendus. Ce driver est incompatible avec le driver « `USB Human Interface Device (full HID) support` ». La réponse recommandée est '`N`'.

L'option « `USB HIDBP Mouse (basic) support` » permet d'activer un driver simplifié pour les souris USB. Ce driver peut être utilisé pour alléger le noyau dans les systèmes embarqués. Il est incompatible avec le driver « `USB Human Interface Device (full HID) support` ». La réponse recommandée est '`N`'.

L'option « `Wacom Intuos/Graphire tablet support` » active la prise en charge des tablettes graphiques Wacom USB. Cette option nécessite d'avoir également activé l'option « `Mouse support` » ou l'option « `Event interface support` » du menu « `Input core support` ». La réponse recommandée est '`N`'.

L'option « `USB Kodak DC-2xx Camera support` » active la prise en charge des caméras USB Kodak DC-2xx et de quelques appareils compatibles. La réponse recommandée est '`N`'.

L'option « `USB Mustek MDC800 Digital Camera support (EXPERIMENTAL)` » permet de prendre en charge les caméras USB Mustek MDC800 et de quelques appareils compatibles. La réponse recommandée est '`N`'.

L'option « `USB Scanner support` » active la prise en charge des scanners USB. La réponse recommandée est '`N`'.

L'option « `Microtek X6USB scanner support (EXPERIMENTAL)` » permet de prendre en charge les scanners Microtek X6USB. Ces scanners apparaîtront dans le système comme des périphériques SCSI génériques. Notez qu'il faut une version modifiée de SANE pour utiliser ce driver. La réponse recommandée est '`N`'.

L'option « `USB IBM (Xirlink) C-it Camera support` » permet de connecter une caméra USB Xirlink d'IBM à votre ordinateur. Cette caméra sera utilisable via l'interface Video4Linux, que l'on devra donc également activer. La réponse recommandée est '`N`'.

L'option « `USB OV511 Camera support` » permet de connecter une caméra USB OV511 à votre ordinateur. Cette caméra sera utilisable via l'interface Video4Linux, que l'on devra donc également activer. La réponse recommandée est '`N`'.

L'option « `D-Link USB FM radio support (EXPERIMENTAL)` » permet de connecter une radio FM USB D-Link. Cette radio sera utilisable via l'interface Video4Linux, que l'on devra donc égale-

ment activer. La réponse recommandée est 'N'.

L'option « DABUSB driver » permet la prise en charge d'un récepteur USB DAB (abréviation de l'anglais « Digital Audio Broadcasting »). La réponse recommandée est 'N'.

L'option « PLUSB Prolific USB-Network driver (EXPERIMENTAL) » permet de prendre en charge les ponts USB-USB PL-2032 de la société Prolific. Ces ponts permettent de relier en réseau deux ordinateurs par l'intérmédiaire de leurs ports USB. La réponse recommandée est 'N'.

L'option « USB ADMtek Pegasus-based ethernet device support (EXPERIMENTAL) » permet de prendre en charge les cartes Ethernet USB. La réponse recommandée est 'N'.

L'option « NetChip 1080-based USB Host-to-Host Link (EXPERIMENTAL) » permet de prendre en charge les ponts USB-USB NetChip 1080. La réponse recommandée est 'N'.

L'option « USS720 parport driver » permet d'activer les convertisseurs USB / Port parallèle permettant de connecter les périphériques utilisant un port parallèle sur le port USB de l'ordinateur. La réponse recommandée est 'N'.

L'option « USB Diamond Rio500 support (EXPERIMENTAL) » permet de prendre en charge les lecteurs MP3 USB Rio500. La réponse recommandée est 'N'.

9.3.60. Menu « USB Serial Converter support »

L'option « USB Serial Converter support » permet de prendre en charge les périphériques USB utilisant une interface série classique. Les options qui suivent activent les drivers spécifiques aux différents matériels supportés par Linux. La réponse recommandée est 'N'.

9.3.61. Menu « Kernel hacking »

L'option « Magic SysRq key » permet d'activer la gestion des séquences de touches utilisant la touche SysRq (ou Syst) en cas de plantage du noyau. Cette option permet de déboguer le noyau lorsque l'on développe une nouvelle fonctionnalité. Répondez 'N' à cette question.

9.4. Compilation du noyau

Une fois la configuration du noyau réalisée, la compilation peut être lancée. Pour cela, il suffit de lancer les trois commandes suivantes dans le répertoire /usr/src/linux :

```
make dep
make clean
make bzImage
```

La première commande génère les dépendances entre les fichiers du noyau. Ces dépendances sont utilisées par les fichiers makefile. La deuxième commande effectue le ménage nécessaire pour supprimer

tous les fichiers objets pouvant résulter d'une précédente compilation. Cette opération est nécessaire afin d'éviter de mélanger des fichiers ayant été compilés avec des options de configuration différentes. Enfin, la troisième commande lance la compilation et l'édition de lien proprement dite.

9.5. Installation du noyau

Une fois la compilation achevée, il faut installer le nouveau noyau. Cette opération nécessite beaucoup de prudence, car si le noyau nouvellement créé n'est pas bon, le système ne redémarrera plus. C'est pour cela qu'il est conseillé de conserver toujours deux versions du noyau, dont on est sûr que l'une d'entre elle fonctionne parfaitement. En pratique, cela revient à conserver la version originale du noyau installé par votre distribution. Pour cela, il faut en faire une copie de sauvegarde.

En général, le noyau est installé dans le répertoire `/boot/` (ou dans le répertoire racine pour les anciennes versions de Linux). Il porte souvent le nom de `vmlinuz`, pour le sauvegarder, il suffit donc de taper par exemple la commande suivante :

```
cp vmlinuz vmlinuz.old
```

Il faut également indiquer au gestionnaire d'amorçage qu'il faut qu'il donne maintenant la possibilité de démarrer l'ancienne version du noyau sous ce nouveau nom. Pour LILO, il suffit d'éditer le fichier `/etc/lilo.conf` et d'y ajouter une nouvelle configuration. En pratique, cela revient à dupliquer la configuration du noyau actuel et à changer simplement le nom du noyau à charger (paramètre « `image` » de la configuration dans `/etc/lilo.conf`) et le nom de la configuration (paramètre « `label` »). Vous devrez aussi rajouter l'option « `prompt` » si elle n'y est pas déjà, afin que LILO vous demande la configuration à lancer à chaque démarrage. Dans notre exemple, le nom du noyau à utiliser pour la configuration de sauvegarde sera `vmlinuz.old`. De même, si la configuration initiale de Linux porte le nom « linux », vous pouvez utiliser le nom « oldlinux » pour la configuration de sauvegarde.

Une fois le fichier `lilo.conf` mis à jour, il faut vérifier que l'on peut bien charger l'ancien système. Pour cela, il faut réinstaller LILO et redémarrer la machine. La réinstallation de LILO se fait exactement de la même manière que son installation, simplement en l'invoquant en ligne de commande :

```
lilo
```

Si LILO signale une erreur, vous devez corriger immédiatement votre fichier `lilo.conf` et le réinstaller.

Vous pourrez alors redémarrer redémarrer la machine avec la commande suivante :

```
reboot
```

LILO affiche alors son prompt (« `LILO boot :` »), et il faut taper le nom de la configuration de sauvegarde :

```
LILO boot :oldlinux
```

Le système doit alors démarrer en utilisant la copie sauvegardée du noyau. Si cela ne fonctionne pas, on peut toujours utiliser le noyau actuel en tapant « linux » à la place de « oldlinux » et corriger le fichier `lilo.conf`.

Lorsque vous vous serez assuré que le système peut démarrer avec la sauvegarde du noyau, vous pourrez installer le nouveau noyau. Son image a été créée par **make** dans le répertoire `/usr/src/linux/arch/i386/boot` sous le nom `bzImage`. L'installation se fait donc simplement par une copie dans `/boot/` en écrasant le noyau actuel `vmlinuz` :

```
cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz
```

Il faut également copier le fichier `System.map` du répertoire `/usr/src/linux/` dans le répertoire `/boot/` :

```
cp System.map /boot
```

Ce fichier contient la liste de tous les symboles du nouveau noyau, il est utilisé par quelques utilitaires systèmes.

Une fois ces deux opérations réalisées, il faut à nouveau réinstaller lilo pour qu'il prennent en compte le nouveau noyau. Ceci se fait avec la même commande que celle utilisée précédemment :

```
lilo
```

Encore une fois, il faut redémarrer la machine avec la commande suivante :

```
reboot
```

et vérifier que le nouveau noyau fonctionne bien. S'il ne se charge pas correctement, c'est que les options de configuration choisies ne sont pas correctes. Il faut donc utiliser le noyau sauvegardé, vérifier ses choix et tout recommencer. Attention cependant, cette fois, il ne faut pas recommencer la sauvegarde du noyau, puisque cette opération écraserait le bon noyau avec un noyau défectueux.

Si le nouveau noyau démarre correctement, il ne reste plus qu'à installer les modules.

9.6. Compilation des modules

Si le système a redémarré correctement, on peut compiler les modules et les installer. Il n'est pas nécessaire de prendre les mêmes précautions pour les modules que pour le noyau. Il suffit donc ici de lancer la commande suivante dans le répertoire `/usr/src/linux/` :

```
make modules
```

Les modules sélectionnés lors de la configuration sont alors compilés, il ne reste plus qu'à les installer.

9.7. Installation des modules

Avant toute installation de nouveaux modules, il est recommandé de décharger tous les modules présents en mémoire. Cette opération peut être réalisée avec les commandes suivantes :

```
modprobe -ar
lsmod
rmmod module
rmmod module
   ⋮
```

Notez que certains modules ne se déchargent pas automatiquement, il faut donc exécuter **rmmod** sur ces modules manuellement. L'installation des modules est alors très simple, puisqu'il suffit de lancer la commande suivante dans le répertoire `/usr/src/linux/` :

```
make modules_install
```

Les modules sont installés dans le répertoire `/lib/module/version/`, où `version` est le numéro de version du noyau courant. Il est possible que des modules d'autres versions du noyau existent dans leurs répertoires respectifs. Si vous n'en avez plus besoin, vous pouvez les effacer. Attention cependant si vous avez installé des modules additionnels non fournis avec le noyau dans ces répertoires, vous pourriez encore en avoir besoin.

Comme on l'a déjà vu, les modules sont utilisés par le chargeur de module du noyau, grâce à la commande **modprobe**. Cette commande a besoin de connaître les dépendances entre les modules afin de les charger dans le bon ordre. Il faut donc impérativement mettre à jour le fichier `/lib/modules/version/modules.d` à chaque fois que l'on installe les modules, à l'aide de la commande suivante :

```
depmod -a
```

> **Note:** La commande `depmod -a` est exécutée automatiquement lors de l'installation des modules du noyau. Toutefois, elle devra être exécutée manuellement si l'on installe des modules non fournis avec le noyau.

Les modules doivent être installés après avoir installé le noyau et redémarré le système, faute de quoi la commande **depmod** peut ne pas trouver trouver tous les symboles utilisés par les modules dans le noyau en court d'exécution.

Chapitre 10. Installation de XWindow

L'installation de XWindow a été pendant longtemps une tâche ardue et risquée. À présent, il est possible d'installer cet environnement relativement facilement, et en prenant beaucoup moins de risques que par le passé. En fait, les seules difficultés dans l'installation de XWindow réside en deux points stratégiques :

- il faut impérativement connaître les caractéristiques de son matériel (carte graphique et surtout moniteur) ;
- il faut disposer d'un driver adapté à sa carte graphique pour le serveur X.

Le premier point n'est pas réellement trop difficile à résoudre, puisqu'il suffit souvent de regarder les fiches techniques du matériel installé. Bien entendu, cela suppose de les avoir conservées. Si ce n'est pas le cas, il faut espérer que les programmes d'installation connaissent la marque et le modèle du matériel. Il reste toujours la possibilité de demander des renseignements à des personnes qui ont également ce type de matériel (c'est là qu'Internet peut être utile). Les informations les plus importantes sont les plages de fréquences horizontales et verticales du moniteur, ainsi que les durées des signaux de synchronisation horizontale et verticale. Sans ces informations, vous ne parviendrez pas à installer XWindow. Rassurez-vous cependant, les programmes de configuration de XWindow connaissent la plupart des moniteurs à présent, ce qui fait qu'ils sont capables d'écrire les fichiers de configuration correctement sans que vous ayez à spécifier les paramètres du moniteur.

Le deuxième point en revanche est plus délicat. Bon nombre de fabricants de matériel tiennent secret les informations techniques permettant de programmer un driver, parce qu'ils considèrent que ce sont des informations stratégiques. Il est évident que dans ce cas, aucun driver libre ne peut être écrit. Depuis quelques temps, ce problème ne se pose plus réellement en ce qui concerne l'affichage 2D, car le champs de bataille des constructeurs de cartes graphiques s'est déplacé vers le monde de la 3D. Dans le pire des cas, la carte graphique ne sera reconnue que par le driver générique VESA, et l'affichage se fera correctement mais avec des performances bien en deçà de ce qu'elles auraient été si un driver approprié avait existé. Il est donc recommandé de se renseigner dans les groupes de discussion sur Internet avant d'acheter une carte graphique, ou d'acheter une bonne carte graphique mais un peu obsolète. Il faut savoir que de toutes façons, les dernières fonctionnalités sont toujours intégrées avec un train de retard sous Linux, car il faut au moins le temps d'écrire des drivers (et contrairement aux autres systèmes, ces drivers sont testés, ce qui prend plus de temps mais assure la fiabilité). Dans peu de temps, Linux sera sans aucun doute reconnu comme un système à part entière par les fabricants, et il est probable qu'ils fourniront des drivers comme pour les autres systèmes. En attendant cet âge d'or, assurez-vous bien que ce que vous achetez fonctionne sous Linux.

Il n'y a que quatre solutions si aucun serveur X adapté à votre matériel n'est fourni avec votre distribution :

- soit le fabricant de la carte graphique fourni un serveur X pour Linux (ce qui est très rare, mais commence à arriver) ;

- soit on utilise le driver VESA fourni avec XFree86, qui fonctionne avec toutes les cartes graphiques compatibles avec le standard VESA
- soit une société tierce vend un serveur X pour ce type de matériel (ce qui est moins rare, mais a l'immense inconvénient qu'il faut acheter ses drivers) ;

Ce chapitre décrit la manière de procéder pour installer et configurer XFree86, une implémentation libre de XWindow. Il indique également comment installer le serveur X pour le driver de frame buffer du noyau. L'installation des police Truetype, qui sont si chères aux utilisateurs de Windows et des Macintosh, est également traitée. En revanche, il ne décrira pas comment configurer les gestionnaires de fenêtres ni les gestionnaires de bureau, car ces opérations sont spécifiques à celui que vous choisirez d'une part, et spécifiques à vos desiderata d'autre part. Pour cela, vous devrez commencer à vous débrouiller tout seul, à lire les pages de manuel et à poser les questions qu'il faut aux personnes qu'il faut. Mais ne paniquez pas, si vous êtes arrivés jusqu'ici, c'est que vous commencez à savoir vous débrouiller. Vous ne devriez plus trop avoir de problèmes pour tirer de Linux tout ce dont vous avez besoin…

10.1. Généralités sur XWindow

Il est nécessaire de bien comprendre ce qu'est XWindow pour pouvoir le configurer et l'utiliser. Son architecture se distingue en effet fortement de celle des environnements graphiques des systèmes classiques comme Windows, OS/2 ou Macintosh, et ces différences se traduisent dans la manière de l'utiliser.

La principale différence entre XWindow et les autres environnement graphiques est qu'il s'agit d'un environnement graphique distribué sur un réseau. La notion de base de XWindow est que l'application qui effectue le traitement ne réalise pas elle-même la gestion de l'environnement graphique. Comme on l'a déjà vu, cette tâche est prise en charge par le serveur X, qui est un processus indépendant. L'application est donc cliente du serveur X, qui lui fourni les services graphiques dont elle a besoin. Cette séparation a plusieurs conséquences :

- premièrement, l'environnement graphique est isolé des fautes des applications qui l'utilisent. Ainsi, ce n'est pas parce qu'une application a planté en plein écran qu'on ne peut pas réduire sa fenêtre et accéder à nouveau au bureau sous-jacent. Inversement, une application est isolée des erreurs potentielles du serveur X, et peut poursuivre son traitement même si celui-ci s'est terminé. Par exemple, un processus de gravage de CD peut terminer le CD en cours même s'il a perdu son interface graphique ;
- deuxièmement, les clients doivent établir une connexion avec le serveur. Cette connexion est réalisée par l'intermédiaire du réseau. Ceci implique naturellement que les mécanismes de sécurité liés au réseau sont applicables pour toutes les applications désirant se connecter au serveur X. XWindow fournit par ailleurs des mécanismes de sécurité complémentaires, et les ressources graphiques ne peuvent être utilisées que par les processus qui y sont autorisées ;
- enfin, comme le client et le serveur X sont deux processus distincts et qu'ils communiquent par l'intermédiaire d'une connexion réseau, rien n'interdit de lancer le client et le serveur X sur deux

machines distinctes. Ainsi, tout processus déporté peut afficher ses données en local (et inversement).

Chaque client doit donc se connecter au serveur X avec lequel il désire travailler. En pratique, il n'existe souvent qu'un seul serveur X sur une machine, mais ceci n'est pas une obligation. Par exemple, une même machine peut disposer de deux cartes graphiques et de deux écrans, et faire tourner deux serveurs X distincts. Une autre possibilité est d'utiliser les deux écrans avec un seul serveur X pour faire un écran virtuel beaucoup plus grand. Enfin, il est tout à fait concevable de lancer plusieurs fois un même serveur X, même si l'on ne dispose que d'une seule carte graphique et d'un seul écran, afin de pouvoir utiliser plusieurs terminaux X virtuels. Comme on le voit, l'architecture client/serveur de XWindow lui apporte une très grande flexibilité.

Les serveurs X utilisent la notion de *display* pour gérer l'affichage. En fait, un display est constitué d'un clavier, d'une souris et d'un ou plusieurs écrans. Le display est donc l'extension de la notion de terminal pour XWindow. Notez bien qu'il est possible d'utiliser plusieurs écrans sur le même terminal X. Cependant, un serveur X ne peut prendre en charge qu'un seul terminal X sur une machine, et chaque display est géré par un serveur X qui lui est propre. Si l'on veut utiliser plusieurs terminaux X sur une même machine, il est nécessaire de lancer plusieurs serveurs X, à raison d'un par terminal.

Les clients qui désirent se connecter à un serveur X doivent donc indiquer le display avec lequel ils désirent travailler. Le système XWindow se chargera d'établir la connexion avec le serveur X en charge de ce display. Les displays sont spécifiés avec la syntaxe suivante :

```
machine :display.écran
```

Comme on le voit, la présence du champ `machine` confirme que XWindow est bien un système graphique réseau. Il peut contenir directement le nom de la machine ou son adresse IP. Si ce champ est absent, le serveur contacté sera l'un des serveurs X de la machine locale, avec un protocole de communication optimisé. Le champ `display` quant à lui est un numéro permettant d'identifier le display de la machine en question qui doit être utilisé. C'est ce champ qui déterminera le serveur X qui sera utilisé, dans le cas où plusieurs serveurs X fonctionnent sur la même machine. Enfin, le champ `écran` spécifie le numéro de l'écran de ce display sur lequel les affichages doivent être faits. Ce champ sera rarement utilisé en pratique, car il est assez rare de disposer de plusieurs écrans. Il peut donc être omis, la valeur par défaut utilisée est dans ce cas 0 (pour le premier et unique écran du display).

Une fois la connexion établie, les programmes clients continuent d'indiquer à XWindow le display qui doit être utilisé pour chaque opération graphique à effectuer. Il est donc possible pour un programme de répartir son affichage sur plusieurs écrans, voire de communiquer avec plusieurs serveurs X et donc de gérer plusieurs displays simultanément, éventuellement sur des machines différentes. Ce genre de programme est cependant assez rare, et ne se trouve en pratique que dans le monde de la conception assistée par ordinateur, la visualisation d'images médicales et l'architecture. Les programmes classiques se contentent d'un seul display, et effectuent toutes leurs opérations sur un même écran. En revanche, il est possible de configurer les serveurs X pour utiliser automatiquement plusieurs écrans pour un même display, afin de réaliser un écran virtuel gigantesque.

Le display utilisé pour un programme doit donc souvent être fixé par un paramètre de sa ligne de

commande. L'option utilisée est `-display`, avec la syntaxe suivante :

```
programme -display nom
```

où `programme` est le programme à exécuter, et `nom` est le nom du display tel qu'il a été décrit ci-dessus. Par exemple, la commande suivante :

```
xterm -display :0
```

permet de lancer le programme `xterm` et de réaliser l'affichage sur le display `:0` (sur l'écran par défaut) de la machine locale.

Vous pouvez cependant vous passer de l'option `-display`, à condition de définir la variable d'environnement DISPLAY pour fixer le display par défaut. Vous pouvez fixer la valeur de cette variable à l'aide d'une commande telle que celle-ci :

```
export DISPLAY= :0.0
```

(si vous utilisez bash). Dans cet exemple, le serveur X à utiliser se trouve sur la machine locale, le display porte le numéro 0, et l'écran à utiliser est le numéro 0. En général, cette variable d'environnement est fixée à la valeur du display courant lorsque l'on est connecté sous XWindow. Par conséquent, vous pouvez lancer vos programmes sans avoir à vous préoccuper du display qu'ils doivent utiliser.

En revanche, vous serez obligé de préciser le display à utiliser lorsque vous lancerez une application à distance en voulant avoir l'affichage en local. Bien entendu, vous devrez au préalable donner les droits à l'utilisateur distant sur votre display local, faute de quoi les mécanismes de sécurité de XWindow lui interdiront de se connecter (message d'erreur « `Can't open display` »). Nous verrons plus loin la manière dont XWindow gère la sécurité.

10.2. Configuration de XFree86

La configuration de XFree86 commence avant tout par l'installation du serveur X. Un même serveur X peut prendre en charge plusieurs cartes graphiques sur une même machine, pourvu qu'il dispose des drivers adéquats. Inversement, il est possible de lancer plusieurs serveurs X, chacun utilisant sa propre carte graphique, ou partageant la même carte si la machine n'a qu'un seul terminal X.

XFree86 fournit un unique serveur X, qui prend en charge tous les types de cartes graphiques à l'aide de drivers spécifiques. Dans une installation normale, ces drivers sont fournis sous forme de modules de ce serveur. Les drivers peuvent ainsi être chargés dynamiquement par le serveur X, selon la configuration du système. Cette architecture permet à un même serveur X de charger plusieurs drivers pour plusieurs cartes graphiques, afin de gérer les configuration disposant de plusieurs écrans connectés à des cartes graphiques de différents types. Il est donc nécessaire que les modules prenant en charge vos cartes graphiques soient installés, ce qui est normalement toujours le cas.

Par convention, le nom du serveur X est toujours `X`. Comme le nom du fichier programme du serveur X de XFree86 est `XFree86` (on s'en serait douté...), il doit donc exister un lien symbolique

`/usr/X11/bin/X` qui pointe vers le fichier `/usr/X11R6/bin/XFree86`. Ce lien est, encore une fois, normalement toujours présent sur les systèmes correctement configurés.

Une fois XWindow installé, la suite de la configuration de XFree86 se fait uniquement dans le fichier de configuration `XF86Config`. Ce fichier est classiquement stocké dans le répertoire `/etc/X11/`. Normalement, vous ne devez pas créer ce fichier vous-même. Votre distribution doit au moins vous en fournir un par défaut, et souvent, elle dispose d'un outil de configuration de XFree86, convivial et qui fera quasiment tout le travail pour vous. Ce genre d'outil est de plus capable de configurer XFree86 pour les drivers spécifiques fournis avec les distributions (il n'est pas rare que les sociétés éditrices de distributions développent des serveurs X pour les nouvelles cartes graphiques). Lisez donc votre documentation pour plus de détails à ce sujet.

Vous pouvez également utiliser les programmes de configuration fournis avec XFree86, qui vous permettront de générer ce fichier. Il existe trois possibilités pour générer un fichier de configuration `XF86Config`. La première méthode est de demander directement au serveur X de détecter le matériel présent sur votre machine et de générer le fichier `XF86Config` correspondant. La deuxième méthode, qui est aussi la plus sûre, est d'utiliser le programme xf86config. Il s'agit d'un outil fonctionnant en mode texte, qui est effroyablement peu pratique à utiliser. Enfin, un autre outil, beaucoup plus convivial, est en cours de développement. Il s'agit de xf86cfg. Cet outil permet de générer et de modifier les fichiers xf86cfg de manière conviviale, soit en mode texte avec des menus, soit en mode graphique. Lorsque cet outil sera terminé, la configuration de XFree86 sera beaucoup plus aisée. Encore une fois, je ne saurais que vous recommander d'utiliser l'outil de configuration fourni avec votre distribution.

10.2.1. Génération automatique du fichier XF86Config

Le serveur X de XFree86 est capable de détecter le matériel installé sur une machine et de générer un fichier de configuration `XF86Config` adapté à ce matériel. Pour cela, il suffit simplement de le lancer avec l'option `-configure` sous le compte root, comme dans l'exemple suivant :

```
XFree86 -configure
```

À l'issue de cette commande, le serveur X écrira un fichier `XF86Config.new` dans le répertoire personnel de l'utilisateur root. Ce fichier pourra être copié dans le répertoire `/etc/X11/` sous le nom `XF86Config` une fois qu'il aura été corrigé.

Cependant, le fichier de configuration ainsi généré n'utilisera que les options de configuration les plus sûres et devra souvent être révisé complètement. En pratique, seuls les paramètres de la carte graphique et de la souris seront correctement détectés. Il ne faut donc utiliser cette fonctionnalité que dans le but d'obtenir un squelette de fichier `XF86Config`, que l'on personnalisera ensuite.

10.2.2. Utilisation de xf86config

Comme il l'a été indiqué plus haut, xf86config est un programme fonctionnant en mode texte exclusivement. Son principe de fonctionnement est très simple : il pose une série de questions sur

votre matériel, puis il génère un fichier `XF86Config` générique pour votre matériel. Vous pourrez ensuite partir de ce modèle pour personnaliser votre configuration et pour préciser les paramètres que xf86config ne prend pas en charge.

Le principal problème de xf86config est qu'il ne laisse pas droit à l'erreur. La moindre faute de frappe ou la moindre hésitation est prise comme une réponse valide, et il est impossible de revenir en arrière. Ceci est d'autant plus énervant que dans ce cas, il ne reste plus qu'une solution, qui est de tout recommencer à partir du début. Après trois ou quatre erreurs, on finit par faire extrêmement attention à ce que l'on tape.

Lorsqu'on lance xf86config, il commence par afficher un message indiquant qu'il va créer un nouveau fichier de configuration `XF86Config` que l'on pourra utiliser par la suite comme point de départ pour paramétrer son système X Window. Il faut valider pour passer ce message et commencer la configuration proprement dite.

La première question que xf86config pose est le type de la souris que vous voulez utiliser. Il existe un grand nombre de types de souris sur le marché, cependant, seuls deux types sont réellement courants.

Initialement, les souris se connectaient sur le port série des ordinateurs. Ces souris, dites souris sérielles, étaient relativement répandues et sont généralement référencées sous le terme de « compatible Microsoft ». Il faut donc utiliser l'option « `Microsoft compatible (2-button protocol)` » pour ces souris. Notez que certaines souris sérielles utilisent un protocole différent pour gérer un troisième bouton. Si vous disposez d'une telle souris, il faut choisir l'option « `Mouse Systems (3-button protocol)` ».

Plus récemment, le port PS/2 est apparu pour les claviers et les souris. Ce port permet de gérer directement la plupart des souris actuelles, et il est probable que votre sourie soient une souris PS/2. L'option à utiliser est cette fois l'option « `PS/2 Mouse` ». Il faut surtout ne pas confondre les souris PS/2 avec les souris bus (que l'on peut utiliser avec l'option « `Bus Mouse` », qui sont des souris relativement peu répandues et qui utilisaient des bus spéciaux. Les autres options sont réservées à des souris peu répandues. Si vous en utiliser une, vous devez choisir l'option correspondante.

Notez que les souris à molette connectées sur le port PS/2 utilisent un protocole de communication légèrement différent de celui que les autres souris PS/2 utilisent. Malheureusement, xf86config ne propose pas d'option pour ces souris, et une intervention manuelle dans le fichier de configuration `XF86Config` est nécessaire pour les configurer. La manière de procéder sera détaillée dans la Section 10.2.4.

La question suivante demande si vous désirez activer la fonctionnalité d'émulation d'un troisième bouton pour les souris à deux boutons. Cette émulation permet d'utiliser les nombreux programmes pour X Window qui nécessitent d'utiliser une souris à trois boutons. Le clic sur le troisième bouton est alors simulé en appuyant sur les deux boutons de la souris simultanément. Il est très vivement recommandé de répondre par '`y`' à cette question si votre souris ne dispose que de deux boutons. En revanche, si elle dispose de plus de trois boutons, ou si elle dispose d'une roulette, il faut répondre par la négative.

xf86config demande ensuite le port sur lequel votre souris est connectée. Ce port est généralement le port `/dev/ttyS0` pour les souris sérielles et `/dev/psaux` pour les souris PS/2. Les distributions créent souvent un lien symbolique `/dev/mouse` sur le port effectivement utilisé par la souris, si bien

que la réponse par défaut utilise ce port. C'est la réponse recommandée, validez-donc pour passer à la question suivante.

Vient ensuite la sélection du type de clavier. Encore une fois, un certain nombre de modèles de claviers ont été vendus sur le marché. Cependant, seuls quelques-uns sont réellement répandus. En France, on trouve essentiellement les claviers internationaux 102 touches et 105 touches, auxquels correspondent les réponses « `Generic 102-key (Intl) PC` » et « `Generic 105-key (Intl) PC` ». Si vous utilisez un clavier Microsoft Natural Keyboard, choisissez l'option « `Microsoft Natural` ».

Vous devrez ensuite indiquer la disposition de ce clavier. Il faut évidemment choisir l'option « `French` ». xf86config demande alors de saisir un nom de variante pour le clavier choisi. Comme le clavier français n'est décliné que sous une seule variante, vous pouvez simplement valider pour passer à la suite de la configuration du clavier.

xf86config vous propose alors de modifier des paramètres additionnels du clavier (emplacement des touches modificatrices telles que les touches de majuscule, de contrôle et de jeu de caractères alternatif, état des diodes, etc...). En général, ceci n'est pas nécessaire et vous pouvez répondre '`n`' à cette question.

La partie difficile vient ensuite. xf86config vous le signale avec un message indiquant que les informations de synchronisation horizontale et verticale sont extrêmement importante pour configurer correctement les modes vidéo. La signification de ces valeurs vous sera décrite plus loin en détail, pour l'instant, contentez-vous de récupérer le manuel de votre moniteur et recherchez ses caractéristiques précises. Validez ensuite pour passer à la question suivante.

xf86config vous demande alors la plage de fréquences horizontales que votre moniteur est en mesure d'accepter. Il propose un certain nombre de choix standard, qui correspondent aux différents types de moniteurs existant sur le marché. Cependant, ces valeurs sont les plus mauvaises, car il est fort probable que votre moniteur sache faire mieux que ce que les standards imposent. Vous devez donc soit accepter une des plages de valeurs proposées, soit saisir la plage correspondant exactement à votre moniteur à l'aide de l'option « `Enter your own horizontal sync range` ». Si vous choisissez cette dernière option, vous devrez ensuite saisir la plage de valeur des fréquences horizontales de votre moniteur. Ne les inventez pas, ça ne marchera pas. Saisissez les vraies valeurs.

La même question est alors posée pour la plage de fréquences verticales. Encore une fois, vous pouvez choisir l'une des plages proposées selon le type de votre moniteur, ou saisir vous même la plage de fréquence citée dans ses caractéristiques techniques à l'aide de l'option « `Enter your own vertical sync range` ». À l'issue de cette question, xf86config vous demande de saisir le nom du moniteur que vous venez ainsi de configurer. Ce nom est arbitraire, il ne sert que pour l'identifier de manière lisible par la suite. Entrez, par exemple, le nom du modèle de l'écran dont vous disposez.

La question suivante vous demande simplement si vous désirez choisir votre carte graphique dans la liste des cartes graphiques supportées par XFree86. Il est recommandé de répondre par l'affirmative en tapant '`y`'. xf86config affiche alors la liste des cartes gérées, qui est assez longue. En fait, elle se présente sur plusieurs pages, et il faut appuyer sur la touche `Entrée` pour passer d'une page à la suivante. Si vous avez passé une page de trop, vous êtes bon pour passer toutes les pages pour revenir à la première, avant d'aller sur la page contenant votre carte graphique. Lorsque vous aurez trouvé votre carte, choisissez l'option correspondante et validez.

Si votre carte n'est pas présente dans la liste, cela ne signifie pas qu'elle n'est pas supportée par XFree86. En effet, chaque driver est capable de prendre en charge un type de puce électronique, et les cartes graphiques sont souvent gérées par les puces électroniques fournies par les mêmes fabricants. C'est la raison pour laquelle les drivers de XFree86 portent généralement le nom des puces utilisées par les cartes. Par conséquent, si vous ne trouvez pas votre carte dans la liste, vous pouvez essayer de spécifier le driver correspondant à l'électronique de votre carte. Dans le pire des cas, vous pourrez quasiment toujours utiliser le driver générique VESA, qui permet de piloter toutes les cartes graphiques compatibles avec les standard VESA 2.0. Cependant, vous ne bénéficierez d'aucune accélération matérielle avec ce driver.

Les questions qui suivent dépendent fortement de la carte sélectionnée. En effet, chaque carte peut avoir besoin d'un certain nombre de paramètres complémentaires pour fonctionner. Encore une fois, ces paramètres sont, normalement, indiqués dans le mode d'emploi de votre carte graphique, ou, au pire, sur les composants de la carte eux-mêmes. Une des questions courantes concerne la quantité de mémoire vidéo dont cette carte dispose. Pour cette question, les choix les plus courants sont proposés, mais vous pouvez également spécifier vous-même cette quantité à l'aide de l'option « `Other` ». L'unité est ici le kilo-octet, pensez donc bien à multiplier par 1024 le nombre de méga-octet de mémoire vidéo de votre carte graphique avant de saisir la valeur. Lorque la configuration de la carte graphique sera terminée, xf86config vous demandera de saisir le nom de cette carte. Encore une fois, ce nom est arbitraire, mais vous devriez saisir un nom cohérent avec votre modèle.

xf86config vous propose alors de modifier l'ordre d'utilisation des modes graphiques pour chaque profondeur de couleur. Cet ordre doit être spécifié en donnant les numéros des différents modes, les uns après les autres, et en ne les séparant pas pas des espaces. Par exemple, le nombre `432` sélectionnera les modes `4`, `3` et `2`, soit les modes 1024x768, 800x600 et 640x480. Notez que l'ordre des numéros est important, c'est l'ordre dans lequel ces modes seront choisi lorsque l'on basculera de l'un à l'autre. Lorsque vous aurez saisi les modes graphiques à utiliser et leur ordre d'apparition, xf86config vous proposera d'utiliser un écran virtuel de taille supérieure à la résolution physique du plus grand des modes graphiques choisi. Ceci signifie que la résolution de l'affichage sera supérieure à celle de votre écran, et que X Window fera défiler automatiquement l'image affichée lorsque la souris sortira de la portion d'image visible. Il est en général conseillé de répondre par la négative à cette question, car cela peut être facilement déroutant. Cependant, c'est une affaire de goût, et vous êtes libre d'accepter ce comportement. Notez que quelle que soit la réponse que vous donniez, X Window utilisera comme résolution logique la résolution du plus grand mode que vous avez sélectionné. Par conséquent, ce mécanisme de défilement sera encore utilisé pour les modes graphiques de résolution inférieure, avec comme taille d'écran virtuel la taille du mode ayant la résolution la plus grande. Lorsque vous aurez configuré tous vos modes graphiques, vous pourrez passer à la suite en choisissant l'option « `The modes are OK, continue.` ».

xf86config vous demande alors le nombre de couleurs à utiliser par défaut lorsque vous démarrez en mode graphique. Choisissez bien, car il ne sera pas possible de changer cette valeur une fois que l'environnement graphique sera lancé. Vous pourrez bien entendu la modifier manuellement, mais il vous faudra relancer le serveur X après cela. La plupart des cartes graphiques modernes supportent les modes graphiques à 16 millions de couleurs, la réponse recommandée est donc « `24 bits (16 million colors)` ».

Enfin, la dernière question est tout simplement si vous désirez enregistrer le fichier `XF86Config`

correspondant aux options que vous avez choisies. Il faut bien entendu répondre par 'y', ou sinon vous devriez vous demander pourquoi vous êtes en train de lire ceci…

Le fichier `XF86Config` généré est généralement fonctionnel, mais parfaitement améliorable. Le format de ce fichier sera décrit en détail dans la Section 10.2.4. Si vous l'ouvrez, vous constaterez que xf86config a ajouté un nombre impressionnant de commentaires pour vous aider dans vos expérimentations. Bien entendu, vous trouverez également des informations complémentaires dans la page de manuel `XF86Config`.

10.2.3. Utilisation de xf86cfg

La manière la plus agréable d'effectuer la configuration de XFree86 est sans nul doute d'utiliser l'utilitaire xf86cfg. Ce programme permet aussi bien de créer un fichier de configuration `XF86Config` initial que de modifier une configuration existante de manière graphique. Il dispose également d'un mode texte, qui reprend les mêmes questions que xf86config, mais d'une manière plus conviviale, à l'aide d'un système de menus.

10.2.3.1. Configuration en mode graphique

Lorsque l'on lance xf86cfg, celui-ci commence par regarder si la variable d'environnement DISPLAY est définie ou non. Si elle est définie, il tente de se connecter au serveur X gérant ce display afin de permettre l'édition du fichier de configuration `XF86Config` du système. Notez qu'il ne permet pas de modifier le fichier `XF86Config` utilisé par le serveur X auquel il se connecte si celui-ci utilise un autre fichier que le fichier du système : le serveur X n'est utilisé par xf86cfg que pour son affichage.

Si, en revanche, la variable d'environnement DISPLAY n'est pas définie, xf86cfg considère que X Window n'est pas installé sur la machine locale et appelle le serveur X avec l'option `-configure` afin de générer un nouveau fichier `XF86Config`. Il lance ensuite le serveur X détecté et utilise ce serveur pour permettre la modification du fichier `XF86Config` ainsi créé.

À son démarrage, il vérifie que la souris est correctement configurée. Si tel n'est pas le cas, il propose d'utiliser les touches du curseur du pavé numérique pour déplacer le pointeur de la souris, ainsi que les touches `/`, `*` et `-` respectivement pour le premier, le deuxième et le troisième bouton de la souris. Je déconseille fortement d'essayer d'effectuer la configuration dans ces conditions, car ce n'est réellement pas utilisable. Dans ce cas de figure, on cherchera plutôt à utiliser l'interface en mode texte de xf86cfg, que nous présenterons dans la section suivante.

L'interface de xf86cfg en mode graphique est très simple. La partie supérieure de la fenêtre comprend un menu avec deux entrées. La première entrée permet de choisir les différentes parties intervenant dans la configuration. La deuxième entrée (intitulée « `Expert Mode` ») donne accès quant à elle à un mode de paramétrage permettant d'éditer directement les propriétés du fichier `XF86Config`. Ce mode de fonctionnement est réservé aux utilisateurs avertis et ne sera pas décrite ici.

La première entrée de menu comprend plusieurs options. L'option « `Configure Layout` » permet d'effectuer la configuration générale du display. L'option « `Configure Screen` » permet de configurer la disposition des différents écrans d'un même display les uns par rapport aux autres. Elle n'est réellement utile que dans le cas des configurations à plusieurs écrans. L'option « `Configure`

`Modeline` » donne la possibilité de configurer les différents modes graphiques de chaque configuration. Enfin, l'option « `Configure AccessX` » permet de spécifier les options des différents périphériques d'entrée. Ces deux options de menus sont réservées aux utilisateurs expérimentés et ne seront pas décrites dans ce document.

L'essentiel de la configuration se fait dans l'écran affiché lorsque l'option « `Configure Layout` » est choisie. Cet écran est constitué d'une barre d'icônes permettant d'ajouter les différents périphériques à la configuration courante : souris, claviers, cartes graphiques et moniteurs. Il n'est possible d'éditer qu'une seule configuration à la fois, la configuration courante peut être sélectionnée à l'aide du bouton situé dans la partie inférieure gauche de la fenêtre. Le corps de la fenêtre elle-même contient une représentation de cette configuration, avec les différents périphériques utilisés et les relations qui existent entre eux.

En cliquant avec le bouton droit de la souris sur ces éléments, vous pouvez faire apparaître un menu contextuel concernant cet élément. Ce menu peut contenir l'option « `configure` », qui donne accès à une boîte de dialogue permettant d'éditer les propriétés de l'élément, l'option « `option` », qui permet d'ajouter des options générales à cet élément, les options « `enable` » et « `disable` », qui permettent d'activer ou de désactiver l'élément en question dans la configuration, et l'option « `remove` », dont le but est de supprimer l'élément concerné.

Une configuration typique comprend au moins une souris, un clavier, une carte graphique et un moniteur. Il est recommandé d'ajouter les éléments de la configuration dans cet ordre, afin d'éviter d'avoir à revenir plusieurs fois sur certains de ces éléments.

La boîte de dialogue ouverte par l'option « `configure` » sur une souris comprend un champs contenant l'identificateur de cette souris, la liste des fichiers spéciaux de périphériques pour la sélection du fichier spécial de périphérique auquel la souris est connectée, la liste des protocoles gérés par XFree86 et un bouton permettant d'activer l'émulation du troisième bouton pour les souris à deux boutons. Généralement, les souris sont connectées soit sur le port série (fichier spécial de périphérique `/dev/ttyS0` ou `/dev/ttyS1`), soit sur le port PS/2 (fichier spécial de périphérique `/dev/psaux`). Les principaux protocoles gérés par XFree86 sont les suivants :

Microsoft

> C'est le protocole utilisé par la plupart des souris connectées sur le port série. Ce type de souris est de plus en plus rare actuellement, ne choisissez cette option que si vous disposez de ce type de souris ;

IntelliMouse

> C'est le protocole utilisé par les souris à molette connectées sur le port série. N'utilisez pas ce protocole pour les souris à molette connectées sur le port PS/2, la souris ne fonctionnerait pas.

PS/2

> C'est le protocole qui convient pour la majorité des souris connectées sur un port PS/2. Notez toutefois que ce n'est pas le cas des souris à molette Logitech et Microsoft ;

IMPS/2

C'est le protocole des souris à molette connectées sur port PS/2. Malheureusement, ce protocole n'est pas proposé par xf86cfg. Il est donc impossible de configurer une souris de ce type avec cet utilitaire, sauf à passer dans le mode expert. La configuration à utiliser pour ce type de souris sera décrite dans la Section 10.2.4.

Lorsque vous aurez configuré votre souris, vous pourrez utiliser le bouton « `Apply changes` » pour prendre en compte les changements si vous avez modifié les paramètres de la souris courante. Dès lors, vous devriez avoir une souris fonctionnelle, et vous pouvez l'activer avec l'option « `enable` » du menu contextuel. Lorsque la souris est activée, un lien apparaît entre l'unité central de l'ordinateur et cette souris dans la fenêtre de xf86cfg.

La boîte de dialogue ouverte par l'option « `configure` » du menu contextuel des claviers comprend, comme la boîte de dialogue des propriétés des souris, un champ permettant de saisir le nom de ce clavier dans la configuration. Vous pouvez également choisir le modèle de clavier et sa disposition en fonction de sa langue. Pour les claviers français, le modèle à utiliser est généralement celui des claviers internationaux 102 ou 105 touches et la disposition à utiliser est la disposition « `French` ». Lorsque vous aurez configuré le clavier correctement, vous pourrez appliquer les changements avec le bouton « `Apply changes` ». Vous pourrez également activer le clavier dans la configuration courante à l'aide de l'option « `enable` » du menu contextuel du clavier. Un lien entre ce clavier et l'unité centrale doit alors apparaître dans la fenêtre principale de xf86cfg.

La boîte de configuration des cartes graphiques contient le champ habituel pour donner un nom à la carte et la liste des cartes graphiques reconnues automatiquement par XFree86. Si votre carte ne se trouve pas dans cette liste, vous devrez choisir le driver à utiliser. En général, les drivers portent le nom de la puce électronique sur laquelle la carte graphique est basée. Si aucun driver n'est adapté pour votre carte, vous pourrez malgré tout utiliser un des drivers génériques `vga` ou `vesa`. Enfin, dans le cas où plusieurs cartes graphiques sont installées dans l'ordinateur, vous devrez spécifier l'adresse de la carte sur le bus PCI/AGP. N'oubliez pas d'activer la carte graphique à l'aide de l'option « `enable` » du menu contextuel lorsque vous aurez fini la configuration.

La boîte de dialogue des moniteurs permet de spécifier un nom pour le moniteur ainsi que ses plages de fréquences horizontales et verticales. Vous pouvez aussi sélectionner la carte graphique à laquelle ce moniteur est connectée. La liste ne propose que les cartes graphiques que vous avez configuré précédemment. Les moniteurs n'ont pas besoin d'être activés, car il sont liés aux cartes graphiques.

Une fois la configuration générale définie, vous pouvez, si elle contient plusieurs écrans, spécifier la disposition relatives de ces écrans à l'aide de l'option « `Configure Screen` » du menu principal de xf86cfg. Cette option affiche les différents écrans dans la fenêtre de xf86cfg, et vous pouvez les faire glisser pour les disposer comme vous le désirez.

Enfin, l'option de menu principal « `Configure Modeline` » vous permettra de définir de nouveaux modes graphiques et de les ajuster. Cet écran fournit les mêmes fonctionnalités que l'utilitaire xvidtune, que l'on décrira en détail dans la Section 10.2.6. Cet écran ne sera donc pas traité plus en détail ici.

Lorsque vous quittez xf86cfg, il vous propose d'enregistrer le fichier de configuration `/etc/X11/XF86Config` correspondant à la configuration que vous venez de définir. Vous pouvez spécifier un autre nom si vous le désirer, afin d'enregistrer ces paramètres dans un autre fichier. xf86cfg demande également si vous

désirez sauvegarder la configuration du clavier, pour le cas où vous auriez modifié les paramètres des périphériques d'entrée.

10.2.3.2. Configuration en mode texte

xf86cfg permet également d'effectuer la configuration de XFree86 en mode texte, à l'aide d'une interface basée sur un système de menu. Bien que ce mode de fonctionnement soit en mode texte, il permet d'effectuer les mêmes tâches que xf86config d'une manière beaucoup plus conviviale.

Pour lancer xf86cfg en mode texte, il suffit simplement de lui passer l'option `-textmode` en ligne de commande :

```
xf86cfg -textmode
```

Lors de son démarrage, xf86cfg affiche un écran d'acceuil, que vous pouvez passer en validant l'option « `Ok` ».

xf86cfg affiche ensuite son menu principal, qui comprend des entrées pour ajouter des souris, claviers, moniteurs et cartes graphiques. Ce menu contient également une entrée « `Configure screen` », qui permet d'associer les cartes graphiques aux moniteurs, et une entrée « `Configure layout` », qui permet de configurer le display en spécifiant le clavier, la souris et les différents écrans à utiliser.

Le menu de configuration de la souris permet d'ajouter, de supprimer et de modifier des souris. L'ajout d'une souris nécessite de saisir un identificateur pour la souris, le protocole que cette souris utilise et si le troisième bouton doit être simulé ou non. xf86cfg demande également le fichier spécial de périphérique à utiliser pour accéder à cette souris. Notez que xf86cfg ne permet pas de choisir le protocole IMPS/2 utilisé par les souris à molette connectées sur le port PS/2, ce type de souris ne peut donc pas être configuré avec cet outil.

De la même manière, le menu de configuration du clavier demande de saisir un identificateur lorsque l'on ajoute un nouveau clavier. Le type de clavier est également demandé, ainsi que sa disposition. Pour les claviers français, il faut utiliser les claviers internationaux 102 ou 105 touches.

La configuration des moniteurs permet de spécifier un identificateur pour le moniteur, ainsi que les plages de fréquences horizontales et verticales. La configuration des cartes graphiques quant à elle permet également de spécifier un identificateur pour la carte, et de choisir le type de carte dans la liste des cartes prises en charge par XFree86. Si votre carte n'est pas listée, vous devrez choisir l'option « `** Unlisted cartd **` » et indiquer le driver à utiliser pour cette carte. Ce driver porte généralement le nom de la puce électronique sur laquelle cette carte est conçue, mais vous pouvez également utiliser les drivers génériques `vga` et `vesa` pour les cartes graphiques compatibles avec ces standards. Enfin, xf86cfg vous propose de spécifier l'adresse de la carte sur le bus PCI/AGP, pour le cas où vous auriez plusieurs cartes installées sur la même machine.

Les identifiants donnés aux moniteurs et aux cartes sont utilisés dans le menu de configuration des écrans, qui demande quelle carte et quel moniteur sont utilisés pour chaque écran. La profondeur de couleur par défaut est également demandée, ainsi que la liste des modes graphiques gérés par cet écran. Notez que l'interface en mode texte de xf86cfg ne permet pas de définir de nouveaux modes

graphiques, pour cela, il faut recourir à l'interface en mode graphique ou éditer manuellement le fichier de configuration `XF86Config`.

Enfin, le menu de configuration général permet de définir, de modifier et de supprimer les displays. Chaque display doit contenir une référence à une souris, un clavier et un ou plusieurs écrans. Remarquez encore une fois que l'interface en mode texte de xf86cfg ne permet pas de préciser la disposition relative des écrans les uns par rapport aux autres. Pour cela, vous devrez utiliser l'interface en mode graphique ou éditer manuellement le fichier de configuration de XFree86.

Lorsque vous aurez fini la configuration de XFree86, vous pourrez l'enregistrer à l'aide du menu « `Write XF86Config and quit` ». Ce menu vous demande le chemin complet sur le fichier de configuration à écrire. Vous pouvez spécifier un autre fichier que le fichier de configuration du système si vous désirez l'éditer et le modifier manuellement de manière indépendante.

10.2.4. Description du fichier XF86Config

Les fichiers générés par les outils de configuration de XFree86 sont, comme nous l'avons déjà précisé, des fichiers devant servir de point de départ pour réaliser votre configuration. Normalement, ils permettent de démarrer le serveur X et de travailler dans de bonnes conditions, toutefois, il est possible d'améliorer sensiblement la qualité de l'affichage et l'ergonomie du système en mettant un peu la main à la pâte. Cette section entreprend donc de décrire la structure du fichier de configuration `XF86Config` et les principales options que l'on peut utiliser. D'autre part, il peut être instructif de connaître la nature des informations que ce fichier contient, et quel sont les conséquences des choix faits dans les utilitaires de configurations.

10.2.4.1. Structure générale du fichier XF86Config

Le fichier `XF86Config` est constitué d'un certain nombre de sections contenant chacune les options pour une partie de l'architecture de XFree86. La configuration de XFree86 est un sujet ardu, aussi seules les principales sections du fichier `XF86Config` seront décrites dans ce document. Vous pouvez vous référer à la page de manuel `XF86Config` et à la documentation de XFree86 pour plus de détails à ce sujet.

Certaines sections peuvent être définies plusieurs fois, avec différents jeux d'options à chaque fois, afin de permettre la définition de plusieurs configurations. Bien entendu, lorsqu'un serveur X tourne, seule une seule configuration est utilisée. Cette configuration est spécifiée soit directement en ligne de commande lors du lancement du serveur X, soit en indiquant une configuration par défaut dans le fichier de configuration.

D'autres sections sont globales, et contiennent les options de configuration générales du serveur X. Ces options concernent typiquement l'environnement dans lequel il est supposé fonctionner, et les options qui ne dépendent pas de la configuration utilisée.

La première section globale est la section « `Files` », qui porte relativement mal son nom, car au lieu de fichiers, elle indique les chemins vers les différentes ressources que le serveur X peut utiliser.

Ces ressources comprennent les polices de caractères, l'emplacement des modules complémentaires et l'emplacement des fichiers de définition de couleurs.

La deuxième section globale est la section « `ServerFlags` ». Cette section contient les options générales à utiliser par défaut pour tous les serveurs X. Elles permettent d'activer ou de désactiver certaines fonctionnalités ou extensions non standard. Ces options fournissent des valeurs par défaut, elles peuvent être modifiées pour chaque serveur X de manière indépendante si nécessaire.

La troisième section globale est la section « `Module` ». Cette section contient les commandes de chargement des différents modules du serveur X, lorsque ce serveur est compilé sous forme modulaire. Chaque module gère une certaine partie des fonctionnalités, et celles-ci peuvent donc être activées ou désactivées au niveau de cette section.

Viennent ensuite les sections de configuration des displays. Ces sections décrivent chacune un des aspects du display, et peuvent apparaîtres en de multiples exemplaires.

Le premier type de section de configuration des displays regroupe toutes les sections qui définissent les périphériques d'entrée de données (clavier, souris, table de digitalisation, etc…). Ces sections sont toutes nommées « `InputDevice` ».

Le deuxième type de section de configuration contient la définition des adaptateurs graphiques. Ces sections permettent de donner les renseignements nécessaires pour la configuration des différentes cartes graphiques installées sur le système. Elles sont toutes nommées « `Device` ».

Viennent ensuite les sections « `Monitor` », qui, comme leur nom l'indique, permettent de définir les caractéristiques physiques des différents moniteurs utilisables. Encore une fois, il peut exister plusieurs sections de ce type, pour définir plusieurs moniteurs dans les configurations multi-têtes.

Normalement, chaque moniteur est, en fonction de ses capacités, capable de gérer différents modes graphiques. Les sections « `Monitor` » peuvent donc contenir des définition de modes graphiques, mais ce n'est pas la technique recommandée. En effet, il est possible d'utiliser plusieurs moniteurs différents avec des modes graphiques identiques, aussi XFree86 donne-t-il la possibilité de définir les modes graphiques en dehors des sections « `Monitor` » (ce n'est toutefois pas une obligation). Dans ce cas, les modes graphiques seront définis dans des sections nommées « `Modes` », et celles-ci seront référencées par les sections des moniteurs qui les utilisent.

Les moniteurs sont faits pour être branchés sur des cartes graphiques. Comme on l'a vu ci-dessus, il est possible de définir plusieurs sections « `Device` » et plusieurs sections « `Monitor` », afin de décrire plusieurs cartes graphiques et plusieurs moniteurs. Il faut donc spécifier quels moniteurs sont connectés à quelles cartes graphiques. C'est exactement le rôle des sections « `Screen` ». Pour XFree86, un écran n'est donc rien d'autre qu'un couple fonctionnel moniteur / carte graphique. Notez que certaines cartes graphiques haut de gamme permettent de connecter plusieurs écrans sur une même carte graphique. Dans ce cas, il faudra définir plusieurs sections « `Device` » pour la même carte graphique, et spécifier les ports de sortie dans les options de ces sections.

Enfin, il ne reste plus qu'à définir le display lui-même. Nous avons déjà défini la notion de display ci-dessus comme étant le regroupement d'un clavier, d'une souris et d'un ou plusieurs écrans. Ces associations sont donc définies dans des sections nommées « `ServerLayout` ». Un section de ce type contient donc nécessairement une référence à une section « `InputDevice` » pour le clavier et une section « `InputDevice` » pour le périphérique de pointage, et une ou plusieurs sections

« `Screen` » pour les écrans utilisés par le serveur X. Dautre part, comme leur nom l'indique, les sections « `ServerLayout` » permettent également de spécifier la disposition relative des écrans les un par rapport aux autres, dans le cas des configurations multi-têtes.

Nous allons à présent décrire un peu plus en détail les différentes sections qui ont été présentées ici. Toutes ces sections sont introduites par le mot-clé « `Section` » suivi du type de la section, indiqué entre guillemets, et se terminent toutes par le mot-clé « `EndSection` ». Entre ces deux balises, un certain nombre d'informations peuvent être spécifiées, et peuvent même être regroupées en sous-sections. Étant donné le grand nombre d'options qui peuvent être indiquées dans les sections du fichier `XF86Config`, seules les plus classiques seront décrites ci-dessous.

10.2.4.2. Section « Files »

La section « `Files` », contient les chemins vers les ressources utilisés par XFree86. Ce peut être les répertoires de polices de caractères, les répertoires d'installation des modules du serveur X, ou encore des chemins indiquant l'adresse et le port de serveurs de polices sur un réseau. Cette section est donc relativement riche de renseignements pour le serveur X. Un exemple typique de section « `Files` » est donné ci-dessous :

```
Section "Files"
    RgbPath     "/usr/X11R6/lib/X11/rgb"
    FontPath    "/usr/X11R6/lib/X11/fonts/local"
    FontPath    "/usr/X11R6/lib/X11/fonts/misc"
    FontPath    "/usr/X11R6/lib/X11/fonts/75dpi"
    FontPath    "/usr/X11R6/lib/X11/fonts/100dpi"
    ModulePath "/usr/X11R6/lib/modules"
EndSection
```

Les chemins indiqués sont des chemins Unix classiques, et sont introduits par les mots-clés « `FontPath` », « `RgbPath` » et « `ModulePath` » respectivement pour les chemins sur les répertoires de polices, les répertoires de fichiers de définition de couleurs et les répertoires d'installation des modules. Cependant, comme on le verra plus loin, l'accès aux serveurs de polices se fait à l'aide d'une syntaxe de chemin particulière.

10.2.4.3. Section « ServerFlags »

Cette section regroupe les principales options globales de tous les serveurs X. Ces options sont généralement introduites à l'aide du mot-clé « `Option` », suivi du nom de l'option entre guillemets, lui-même suivi d'une éventuelle valeur pour cette option, elle aussi entre guillemets. Par exemple, l'option :

```
Option "DontZap"
```

permet d'empêcher que l'utilisateur puisse utiliser la combinaison de touches `CTRL+ALT+ BACKSPACE` (en temps normal, cette combinaison de touches a pour conséquence de tuer le serveur X, et de forcer la déconnexion de toutes les applications en cours de fonctionnement).

De même, l'option :

```
Option "Dont Zoom"
```

désactive les combinaisons de touches `CTRL+ALT+PLUS` et `CTRL+ALT+MINUS`, où `PLUS` et `MINUS` sont respectivement les touches plus et moins du pavé numérique. Ces deux combinaisons de touches sont normalement utilisées respectivement pour passer d'un mode vidéo au suivant ou au précédent.

Il existe beaucoup d'autres options générales du serveur X. Les plus intéressantes sont sans doute les options « `BlankTime` », « `StandbyTime` », « `SuspendTime` » et « `OffTime` », qui permettent de contrôler les durées des économiseurs d'écrans matériels. Vous trouverez la description des autres options dans les fichiers `XF86Config` générés automatiquement, et bien entendu dans la page de manuel `XF86Config`.

10.2.4.4. Section « Module »

Cette section contient la liste des modules que les serveurs X doivent charger lorsqu'ils démarrent. Ces modules peuvent être spécifiés de deux manières différentes, une abrégée et une plus complète. La forme abrégée utilise le mot-clé « `Load` », suivi du nom du module entre guillemets. Par exemple, la ligne suivante permet de demander le chargement du module de gestion des polices TrueType :

```
Load "freetype"
```

La forme complète utilise une sous-section, introduite par le mot-clé « `SubSection` » suivi du nom du modules entre guillemets, et se terminant par le mot-clé « `EndSubSection` ». Ces sous-sections ont la même structure que les sections normales, et permettent de spécifier des options pour le module à l'aide du mot-clé « `Option` ». Par exemple, les extensions du serveur X peuvent être chargées à l'exception de l'extension DGA à l'aide de la sous-section suivante :

```
SubSection "extmod"
    Option "omit XFree86-DGA"
EndSubSection
```

> **Note:** L'extension DGA permet aux applications d'accéder directement à la mémoire vidéo de la carte graphique. La désactiver accroît donc la sécurité du système, mais peut empêcher certaines applications de fonctionner correctement. C'est le cas des jeux, et surtout des programmes de télévision sous X Window. Par conséquent, il est probable que vous vouliez laisser ces extensions activées. Dans ce cas, il suffit simplement de ne pas spécifier l'option `omit` de l'exemple précédent.

10.2.4.5. Section « InputDevice »

Les sections « `InputDevice` » permettent de décrire tous les types de périphériques d'entrée. En pratique, il s'agira souvent de claviers et de souris, mais il est également possible de connecter des périphériques plus exotiques tels que les joysticks et les tablettes de dessin.

Les sections « `InputDevice` » doivent être nommées avec un identificateur unique, introduit par le mot-clé « `Identifier` ». Ce mot-clé doit être suivi du nom de la section, donné entre guillemets. C'est ce nom qui sera utilisé pour référencer la section dans les sections « `ServerLayout` » pour définir les différents displays.

La nature du périphérique d'entrée est ensuite spécifiée par la valeur de l'option « `Driver` ». En pratique, on utilisera quasiment toujours `Keyboard` ou `Mouse`, qui correspondent bien évidemment aux drivers pour les claviers et les souris.

La suite des options de la section « `InputDevice` » est fonction de la nature du driver utilisé. Les options les plus courantes pour un clavier sont données dans l'exemple suivant :

```
Section "InputDevice"
    Identifier  "Clavier 1"
    Driver      "Keyboard"

# L'option suivante permet de spécifier les délai
# avant répétition (en millisecondes) et la vitesse
# de répétition une fois ce délai passé (en caractères
# par secondes) :
    Option      "AutoRepeat" "500 30"

# L'option suivante indique que la définition de clavier
# à utiliser est celle de XFree86 :
    Option      "XkbRules" "xfree86"

# L'option suivante indique que le modèle de clavier
# utilisé est un clavier 105 touches :
    Option      "XkbModel" "pc105"

# L'option suivante indique que la disposition
# des touches est celle d'un clavier français :
    Option      "XkbLayout" "fr"

EndSection
```

De même, vous trouverez ci-dessous un exemple typique de section « `InputDevice` » pour une souris :

```
Section "InputDevice"
    Identifier  "Souris 1"
    Driver      "Mouse"
```

```
# L'option suivante indique que la souris est
# une souris de type PS/2 :
    Option      "Protocol" "PS/2"

# L'option suivante indique le fichier spécial
# de périphérique à utiliser pour cette souris :
    Option      "Device" "/dev/psaux"

# L'option suivante indique le nombre de boutons
# de la souris :
    Option      "Buttons" "2"

# L'option suivante demande à XFree86 d'émuler
# le troisième bouton de la souris par clic
# simultané des deux boutons :
    Option      "Emulate3Buttons"

EndSection
```

> **Note:** Pour les souris à molette connectées sur port PS/2, vous devrez choisir le protocole « `IMPS/2` » au lieu du protocole « `PS/2` » et indiquer que la souris a 5 boutons. De plus, vous devrez ajouter l'option « `ZAxisMapping` » dans la section « `InputDevice` » de votre souris, et lui donner la valeur « `4 5` ». Cette option permet de rediriger la rotation de la molette sur les événements des boutons 4 et 5 de la souris. De cette manière, vous pourrez utiliser votre molette dans toutes les applications qui gèrent ces deux boutons.

10.2.4.6. Sections « Device »

Les sections « `Device` » contiennent les descriptions des cartes graphiques utilisées. Cette description n'indique absolument pas quel est le serveur X utilisé : elle ne contient qu'une description du matériel. Notez, encore une fois, qu'il est possible de définir plusieurs sections « `Device` », qui pourront être utilisées dans les sections « `Screen` » associant les cartes graphiques aux moniteurs.

Tout comme les sections « `InputDevice` », les sections « `Device` » doivent disposer d'un identificateur unique, afin de pouvoir les référencer dans les sections « `Screen` » qui les utilisent. Cet identificateur est introduit par le mot-clé « `Identifier` », suivi d'un nom indiqué entre guillemets.

Les sections « `Device` » peuvent contenir un certain nombre d'options permettant de caractériser la carte graphique et de préciser ses capacités (mémoire, rapidité, adresse sur le bus PCI, etc…). Toutes ces options sont bien entendu spécifiques aux différentes cartes graphiques, et ne seront pas décrites en détail ici.

L'option la plus importante est sans doute l'option « `Driver` », qui permet d'indiquer le driver que le serveur X devra charger pour piloter cette carte graphique. C'est à ce niveau qu'il faut surtout ne pas se tromper, puisque la moindre erreur ferait échouer le démarrage du serveur X. En général, les drivers portent le nom du chipset utilisé par la carte graphique, si bien qu'il n'est pas difficile de déterminer la

valeur exacte à utiliser. XFree86 fournit également deux drivers génériques `vga` et `vesa`, permettant respectivement de prendre en charge les cartes graphiques compatibles avec le standard VGA ou compatibles avec le standard VESA 2.0. Sachez cependant que ces drivers ne disposent d'aucune accélération matérielle, et que leurs performances sont donc bien inférieures aux drivers spécialisés pour chaque carte graphique.

Vous trouverez ci-dessous un exemple de section « `Device` » typique :

```
Section "Device"
    Identifier  "Carte 1"
    VendorName  "CHAINTECH"
    BoardName   "TORNADO I7000"

# Cette option indique au serveur X qu'il doit
# charger le module i740_drv pour piloter cette
# carte graphique :
    Driver      "i740"

# Cette option indique la quantité de mémoire
# vive dont cette carte dispose (en ko) :
    VideoRam    8192
EndSection
```

Note: Les options des sections « `Device` » ne sont pas très compliquer à utiliser. Cependant, quelques-unes peuvent poser des problèmes lorsque l'on réalise des configurations multi-têtes (c'est à dire des configurations disposant de plusieurs écrans). Dans la majorité des cas, il faut installer plusieurs cartes graphiques dans le PC pour pouvoir utiliser plusieurs écrans. En général, on utilise une carte AGP et une ou plusieurs cartes PCI. Il va de soi qu'il faut que chaque driver utilise la bonne carte, aussi faut-il indiquer, dans chaque section « `Device` », l'adresse de la carte définie par la section. Cette adresse est définie à l'aide de l'option « `BusID` ». Cette option permet de retrouver sur les différents bus systèmes la carte graphique. Les cartes AGP utilisant le même mécanisme d'adressage que les cartes PCI, on utilisera la syntaxe suivante :

```
"PCI:bus:périphérique:fonction"
```

où `bus` est le numéro du bus PCI sur lequel se trouve la carte graphique, `périphérique` est le numéro du périphérique sur ce bus, et `fonction` est le numéro de la fonction à utiliser pour accéder à ce périphérique. Vous pouvez utiliser la commande **lspci** pour afficher les caractéristiques des différents bus PCI de votre machine et déterminer les valeurs utilisées par vos cartes graphiques.

Certaines cartes graphiques haut de gamme disposent de plusieurs contrôleurs vidéo et sont donc capables de gérer plusieurs écrans. Pour ces cartes graphiques, le principe de fonctionnement est le même : il faut écrire une section « `Device` » pour chaque contrôleur vidéo existant. Cependant, étant donné que toutes ces sections s'adressent à la même carte graphique, il est impossible de les distinguer par l'intermédiaire de l'option « `BusID` ». Dans ce cas, on utilise alors l'option « `Screen` », suivie du numéro du contrôleur vidéo que la section décrit. Ce numéro doit être donné juste après l'option « `Screen` », sans guillemets. La numérotation des contrôleurs

vidéo commence à 0. C'est cette valeur qui est utilisée pour toutes les cartes graphiques ne disposant que d'un seul contrôleur vidéo.

10.2.4.7. Sections « Monitor »

Les sections « `Monitor` » contiennent les informations descriptives des écrans. Notez qu'il est possible de définir plusieurs sections « `Monitor` », pour plusieurs écrans potentiels. Cependant, chaque section « `Screen` » devra utiliser un moniteur et un seul.

Les sections « `Monitor` » sont certainement les sections les plus importantes du fichier `XF86Config`. Elles contiennent en effet toutes les informations concernant les moniteurs et tous les paramètres des modes graphiques utilisés par ces moniteurs. Ces informations sont utilisées par le serveur X pour générer les signaux vidéo que la carte graphique envoie au moniteur. Il est donc évident que chaque section « `Monitor` » est spécifique à un moniteur donné, et que la détermination des valeurs qui doivent y être écrites requiert une bonne connaissance du fonctionnement de ce moniteur et de ses caractéristiques physiques. Vous trouverez une description générale du fonctionnement des moniteurs dans les paragraphes qui suivent. Il est impératif de bien les assimiler si vous désirez créer ou modifier une section « `Monitor` », car si les informations qui y sont stockées sont erronées, le moniteur correspondant risque d'être gravement endommagé (et vous risquez d'être irradié de rayons X).

Le principe de fonctionnement des tubes cathodiques utilisés dans les moniteurs est très simple : à chaque instant, un faisceau d'électrons généré par un canon à électrons vient frapper un point d'une plaque phosphorescente située juste derrière la vitre du moniteur. L'énergie cinétique des électrons du faisceau est transformée en lumière par le phosphore en ce point, qui reste ainsi lumineux tant que le faisceau frappe ce point. Comme un seul faisceau est utilisé pour tous les points de la plaque phosphorescente, il faut déplacer le faisceau de point en point pour pouvoir tous les éclairer. Ceci a bien entendu pour conséquence que les points ne restent lumineux que pendant un temps très court. Cependant, nous percevons la luminosité de ce point pendant un temps beaucoup plus long, en raison de la rémanence des cellules de l'½il. Les points de l'écran semblent donc être tous allumés en même temps, bien qu'à chaque instant un seul d'entre eux est atteint par le faisceau d'électrons. Nous voyons ainsi l'image complète du moniteur.

Le déplacement du faisceau d'électrons est assuré par un champ magnétique généré par des bobinages magnétiques à l'arrière du tube cathodique. Le faisceau balaye l'écran ligne par ligne, de gauche à droite. À chaque fois qu'il atteint le bord gauche, il retourne rapidement vers le début de la ligne suivante, sur le bord droit du moniteur. Ce retour se fait simplement en modifiant brusquement le champ magnétique qui dirige le faisceau. Il va de soi que le faisceau d'électrons est coupé pendant ce retour de balayage, car s'il ne l'était pas, on le verrait traverser l'écran de droite à gauche et perturber ainsi l'image. De plus, le champ magnétique n'est pas facilement contrôlable lorsque le retour du faisceau commence. Par conséquent, il est nécessaire que ce retour se fasse un certain temps après que le bord droit ait été éteint. Si ce n'était pas le cas, l'image serait déformée près des bords du moniteur. Le faisceau électronique continue donc de balayer l'écran sur une petite zone après le bord de l'image, sans émettre de lumière. Cette zone s'appelle le « blanking » horizontal.

De même, le faisceau électronique retourne rapidement en haut de l'écran dès qu'il en atteint le bas. Pour les mêmes raisons que pour les retours de balayage horizontal, le faisceau doit être éteint un peu

avant et pendant la modification du champ magnétique lors des retours de balayage vertical. La zone ainsi parcourue par le faisceau après le bord de l'image s'appelle le « blanking » vertical.

> **Note:** Il est possible d'utiliser d'autres méthodes de balayage que celle décrite ci-dessus. En particulier, les modes graphiques entrelacés ne suivent pas ce principe. Dans ces modes, seulement une ligne sur deux est balayée à chaque génération d'image. Les lignes paires sont d'abord affichées pour une première image, puis les lignes impaires sont affichées pour l'image suivante. Ces modes graphiques étaient utilisés lorsque les moniteurs n'étaient pas suffisamment rapides pour afficher une image complète sans que l'on puisse voir le balayage du faisceau d'électrons. De nos jours, il est déconseillé d'utiliser ces modes, car l'image a tendance à trembler du fait de l'écart de luminosité alternatif entre les lignes paires et impaires de l'écran.

Les limitations techniques de la technologie des tubes cathodiques imposent une limite supérieure pour la vitesse de balayage, et une limite inférieure pour la durée des retours de balayage horizontal et vertical, ainsi que pour la durée minimale du blanking. Les caractéristiques techniques des tubes sont donc souvent spécifiés en terme de fréquences de balayage et de durée des signaux de synchronisations. Les vieux moniteurs ne pouvaient supporter qu'un nombre limité de fréquences de balayage, que l'on qualifiaient alors de « fixes ». De nos jours, les écrans multisynchrones acceptent toutes les fréquences comprises dans une plage de valeur, et s'adaptent donc aux fréquences des signaux émis par les cartes graphiques.

La fréquence de balayage vertical utilisée dans un mode graphique donné constitue le taux de rafraîchissement de l'écran, c'est à dire le nombre de fois que l'image de l'écran est affichée par secondes. Il est évident qu'un taux de rafraîchissement trop faible ne permet pas d'avoir une image agréable à regarder, car elle semble trembler, ou clignoter. Un tube cathodique de bonne qualité est donc un tube qui permet d'utiliser de hautes fréquences verticales. Comme il est nécessaire d'afficher toutes les lignes pour afficher une image complète, il va de soi qu'un bon tube doit également être capable d'afficher les lignes rapidement, et donc accepter des fréquences de balayage horizontal élevées. Typiquement, les fréquences de balayage horizontal sont environ mille fois plus élevées que les fréquences de balayage vertical, puisque chaque ligne peut contenir environ mille pixels.

La sensibilité au taux de rafraîchissement dépend des personnes et de l'emploi que l'on fait du tube. Les tubes destinés à être regardés de loin peuvent utiliser des fréquences beaucoup plus faibles que les tubes de proximité. Ainsi, les télévisions affichent 25 images par secondes (en fait, elles affichent 50 demi-images par seconde, car elles utilisent l'entrelacement). En revanche, ce taux de rafraîchissement est très insuffisant pour les moniteurs d'ordinateurs, où la limite inférieure est de 60 Hz. Le standard VESA exige des taux de 72 Hz ou plus pour que l'image soit agréable à regarder, mais de nombreux écrans supportent facilement 85 Hz ou plus de nos jours. Certaines personnes ne sont cependant pas dérangées par des taux aussi faibles que 60 Hz. Il est bon de savoir quel taux de rafraîchissement vous convient, car un balayage insuffisant peut fatiguer vos yeux, même sans que vous en soyez conscient. Et cette fatigue se traduit toujours un jour ou un autre par des maux de têtes inexpliqués… Ne vous vantez donc pas de voir le balayage, c'est un grave handicap par rapport à ceux qui ne le voient pas et qui donc ont moins de maux de tête que vous !

Vous pouvez utiliser la méthode suivante pour déterminer si le taux de rafraîchissement est suffisant pour vous. Commencez par afficher une image blanche sur la totalité de la surface visible de votre moniteur. Puis réglez la luminosité du moniteur à son maximum. Placez-vous ensuite de biais par

rapport à votre moniteur, et regardez en face de vous. Concentrez ensuite votre attention sur l'image de l'écran, sans le regarder directement (regardez-le « en coin »). Normalement, vous devez percevoir l'image avec les cellules latérales de la rétine, qui donnent une image moins précise que les cellules centrales, mais qui sont plus sensibles aux mouvements (ces cellules nous servent en quelque sorte d'antennes). Ces cellules sont donc plus aptes à percevoir les variations rapides de luminosité de l'image dues au balayage du faisceau électronique. Si l'image semble clignoter, c'est que le taux de rafraîchissement de l'écran est insuffisant. Sinon, vous tenez le bon taux, ne le lâchez pas.

Certaines caractéristiques techniques des tubes cathodiques influent sur les autres et permettent donc de modifier le taux de rafraîchissement. Il est évident que si la durée du retour du faisceau est importante, l'écran ne peut pas générer beaucoup d'images par seconde. Par conséquent, plus la durée du retour de balayage est longue, moins bon peut être le taux de rafraîchissement. Il en va de même pour la durée du blanking. Les bons écrans disposent donc souvent de retours de balayage rapides et de courtes durées de blanking. Mais avoir diminuer ces durées nécessite une électronique plus précise au niveau du contrôle des champs magnétiques dirigeant le faisceau d'électrons. Il est également évident que la résolution du mode graphique influe sur le taux de rafraîchissement. Si la résolution est élevée, il y a plus de pixels à afficher au total sur chaque ligne de l'écran. À vitesse de balayage des pixels fixée, le temps nécessaire pour balayer une ligne est donc plus grand. Et comme il y a plus de lignes à afficher, le temps nécessaire pour générer une image complète est plus grand, et le taux de rafraîchissement est plus faible. Si le test décrit dans le paragraphe précédent vous indique que votre taux de rafraîchissement est trop faible, vous pouvez donc tenter de baisser la résolution.

> **Note:** Notez que la profondeur de couleur utilisée n'intervient absolument pas dans le taux de rafraîchissement. Cette profondeur n'est un facteur limitant qu'au niveau de la carte graphique, qui peut ne pas avoir assez de mémoire pour afficher un mode de grande résolution avec toutes les couleurs désirées ou ne pas avoir une électronique assez rapide pour traiter toutes les données à une vitesse raisonable.

Le signal vidéo émis par la carte graphique contient toutes les informations dont le moniteur a besoin pour générer l'image. Ces informations spécifient bien entendu l'intensité du faisceau d'électrons, mais aussi les signaux de synchronisation qui contrôlent les retours de balayages horizontaux et verticaux. Ce sont les informations de synchronisation que les sections « `Monitor` » du fichier `XF86Config` contiennent. On comprend donc l'importance de ces informations pour la qualité de l'image, aussi bien en terme de stabilité qu'en termes de taux de rafraîchissement.

Les sections « `Monitor` » sont constituées grosso modo de deux parties. La première partie contient les caractéristiques générales de l'écran, telles que son nom, son modèle et le nom de son fabricant. Elle contient également les plages de fréquences de balayage horizontal et vertical. La deuxième partie contient les informations des modes graphiques, à raison d'une ligne par mode graphique utilisable. La documentation de XFree86 appelle ces lignes des « lignes de modes » (« `modelines` » en anglais). Il est évident que les informations fournies dans les lignes de mode sont très sensibles, et déterminent tous les paramètres du mode graphique : résolution, position et taille de l'image sur l'écran, taux de rafraîchissement.

La première partie d'une section « `Monitor` » ressemble à ceci :

```
Section "Monitor"
    Identifier "Moniteur 1"
# Marque et modèle du moniteur (informations facultatives) :
    VendorName "SONY"
    ModelName     "MULTISCAN 100ES"
# Plage de fréquence horizontale (information capitale) :
    HorizSync     30-70
# Plage de fréquence verticale (information capitale) :
    VertRefresh   50-120
```

Le mot-clé « `Identifier` » permet de donner un nom à la section « `Monitor` ». Ce sera ce nom que l'on utilisera plus loin dans la section « `Screen` » pour indiquer que l'on désire utiliser ce moniteur. Les mots-clés « `VendorName` » et « `ModelName` » permettent de donner la description du moniteur. Les informations stockées ici n'ont aucun effet sur la configuration du moniteur et sont simplement indicatives. En revanche, les plages de fréquences horizontales et verticales indiquées respectivement après les mots-clés « `HorizSync` » et « `VertRefresh` » sont d'une importance capitale. Elles donnent les plages de fréquences auxquelles le moniteur peut travailler, et servent à contrôler la validité des lignes de modes données dans la deuxième partie de la section « `Monitor` ». Il est donc important de donner ici les valeurs exactes, que vous trouverez normalement dans la fiche technique de votre moniteur (cette fiche est souvent imprimée à la fin du mode d'emploi). Pour les moniteurs multisynchrones, les plages de fréquences utilisables peuvent être données par leurs fréquences minimum et maximum, séparées par un tiret. Les fréquences fixes peuvent être spécifiées simplement avec une seule valeur. Il est possible de spécifier plusieurs fréquences fixes et plusieurs plages de fréquences en les séparant par des virgules. L'unité utilisée pour les fréquences horizontales est le kilohertz, et celle utilisée pour les fréquences verticales est le Hertz.

La deuxième partie de la section « `Monitor` » contient les lignes de mode, à raison d'une ligne de mode par mode graphique que le moniteur est capable d'afficher. En général, une ligne de mode ressemble à ceci :

```
Modeline "1024x768" 92.96 1024 1064 1240 1352 768 768 778 802
```

mais il existe cette une autre syntaxe, beaucoup plus claire car elle différencie les différentes valeurs données dans la ligne précédente :

```
Mode "1024x768"
    DotClock 92.96
    HTimings 1024 1064 1240 1352
    VTimings 768 768 778 802
EndMode
```

La première valeur indiquée entre guillemets dans les lignes de mode est le nom du mode graphique. Comme on le verra plus tard, c'est ce nom qui est utilisé pour spécifier les modes graphiques utilisables dans la section « `Screen` ». La deuxième valeur est la fréquence de base des signaux émis par la carte graphique. Cette vitesse est exprimée en MHz et représente le nombre de pixels par seconde que

le faisceau électronique du moniteur balaye. Les quatre valeurs suivantes fixent les paramètres horizontaux du mode graphique : résolution horizontale, pixel à partir duquel le signal de synchronisation pour le retour de balayage horizontal commence, pixel auquel ce signal s'arrête et longueur totale de la ligne en pixel, blanking compris. De même, les quatre dernières valeurs fixent les paramètres verticaux du mode graphique, à savoir résolution verticale, ligne de début et ligne de fin du signal de synchronisation pour le retour de balayage vertical, et nombre de lignes total du mode graphique. On notera que les informations concernant le signal de synchronisation du balayage horizontal sont toutes multiples de 8. Cette contrainte est imposée par le matériel de la plupart des cartes graphiques. D'autre part, l'unité utilisée pour ces valeurs est le pixel. Il faut donc se baser sur la fréquence de base indiquée au début de la ligne de mode pour les convertir en temps. Les informations concernant le signal de synchronisation vertical, quant à elles, sont exprimées en nombre de lignes. Elles peuvent ne pas être multiples de 8. La conversion en temps se calcule cette fois à partir de la fréquence de base et de la longueur totale d'une ligne horizontale pour ce mode graphique. Enfin, il est possible de rajouter des options complémentaires à la fin des lignes de mode pour préciser le fonctionnement du mode graphique, à l'aide du mot clef « `Flags` ». L'option la plus courante étant sans aucun doute « `Interlace` », qui permet de créer des modes entrelacés. L'option « `Doublescan` » permet de faire en sorte que chaque ligne est affichée deux fois de suite. Elle permet de créer des modes graphiques de faible résolution, qui utilisent des fréquences de balayage trop faibles pour les moniteurs actuels. Vous pourrez trouver les autres options dans la page de manuel `XF86Config`.

La section « `Monitor` » se termine, comme toute section du fichier `XF86Config`, par une ligne de fin de section :

```
EndSection
```

Comme vous vous en êtes aperçu, les données stockées dans les lignes de mode sont assez techniques. Quelques explications complémentaires ne seront donc pas de trop pour comprendre l'influence de ces paramètres et pour créer de nouvelles lignes de mode.

Pour commencer, la fréquence de base de la ligne de mode doit évidemment être gérée par la carte graphique. Vous pouvez déterminer la fréquence de base maximale en regardant les informations affichées par le serveur X adapté à cette carte lorsqu'il démarre. Cette information est toujours juste, même si les lignes de mode écrites dans votre fichier `XF86Config` ne conviennent pas et que le serveur X ne démarre pas correctement. La ligne contenant cette fréquence maximale est semblable à celle-ci :

```
Maximum allowed dot-clock : 163.000 MHz
```

Cette fréquence de base doit peut être générée par la carte graphique, mais il n'est pas certain qu'elle soit supportée par le moniteur. En effet, les moniteurs ne sont pas capables de gérer les fréquences de base au delà d'une certaine limite. La gamme de fréquences qu'ils acceptent est ce que l'on appelle leur bande passante. La fréquence de base doit donc être dans cette bande passante, c'est à dire qu'elle doit être inférieure à la fréquence maximale que le moniteur peut gérer. La bande passante de votre moniteur est normalement indiquée dans sa fiche de spécifications techniques. Si toutefois ce n'est

pas le cas, vous pouvez vous faire une idée de la fréquence la plus élevée de cette bande passante en regardant la résolution maximale qu'il peut afficher et en consultant ce tableau :

Tableau 10-1. Fréquence maximale des moniteurs

Résolution	Fréquence maximale
640x480	25
800x600	36
10254x768	65
1280x1024	110
1600x1200	185

Les valeurs des fréquences indiquées dans ce tableau sont les plus petites valeurs constatées pour l'ensemble des moniteurs gérés par XFree86. On peut donc supposer que votre moniteur gère ces fréquences, et certainement même de plus hautes.

La résolution du mode graphique est donnée par la première des quatre valeurs des paramètres horizontaux et verticaux dans la ligne de mode. Ainsi, dans la ligne de mode exemple donnée ci-dessus, la résolution horizontale est de 1024 et la résolution verticale est de 768. Sachez que vous êtes parfaitement libre d'utiliser des résolutions non standard, en écrivant les lignes de mode correspondantes. Dans l'exemple donné ci-dessus, il s'agit simplement du mode graphique 1024x768. Notez, encore une fois, que la profondeur de couleur n'intervient absolument pas dans les paramètres du moniteur. Ils n'interviennent que dans les composants de la carte graphique qui sont chargés de générés les signaux vidéos pour le moniteur.

La quatrième valeur des paramètres horizontaux donne la longueur totale des lignes physiques dans ce mode. Cette longueur comprend bien entendu la partie visible de l'image, mais également la partie due au blanking horizontal. Rappelons que le blanking est la partie balayée par le faisceau alors qu'il est éteint, ce qui comprend la partie balayée pendant la durée du signal de synchronisation horizontal. Dans l'exemple donné ci-dessus, le blanking a une longueur égale à 1352-1024 pixels, soit 328 pixels. La quatrième valeur des paramètres verticaux fixe de même le nombre de lignes total du mode, en tenant compte des lignes du blanking vertical.

Comme vous pouvez le constater dans la ligne de mode exemple, le pixel auquel le signal de synchronisation commence n'est pas le dernier pixel visible. En effet, il y a 40 pixels d'écart entre le dernier pixel visible (en l'occurrence, le pixel 1024) et le pixel auquel le signal de synchronisation horizontal commence (le pixel 1064). Cet espace fait partie du blanking, il sert de marge pour éviter les perturbations dues au retour de balayage et pour positionner l'image sur l'écran. Cette zone noire se trouve au delà du bord droit de l'image affichée. Elle est suivie par la zone balayée pendant le retour de balayage, qui se termine au pixel indiqué par la troisième valeur (1240 dans notre exemple). Ce pixel est donc toujours positionné sur le point le plus à droite que le moniteur peut afficher. Par conséquent, la taille de la partie du blanking situé à la droite de l'image est égale au numéro du pixel où le signal de synchronisation horizontal se termine moins le nombre de pixels affichés. Dans notre exemple, cette partie de blanking a une taille de 1240-1024 pixels, soit 216 pixels. La taille de la partie du blanking horizontal situé à la gauche de l'écran est donc logiquement égale à la longueur totale

de la ligne (1352) moins le pixel où le signal de synchronisation se termine (1240), ce qui donne 112 pixels. Vous voyez ainsi que l'on peut décaler l'image vers la gauche en augmentant les valeurs de départ et de fin du signal de synchronisation. Inversement, on peut décaler l'image vers la droite en diminuant ces valeurs.

Le même raisonnement peut être appliqué aux paramètres verticaux du mode graphique. Le blanking vertical comprend les lignes balayées avant et après le signal de synchronisation, ainsi que les lignes balayées pendant le temps où ce signal est généré. La dernière ligne de ce signal est, encore une fois, la ligne la plus basse que le moniteur peut afficher. On peut donc déplacer l'image vers le haut ou vers le bas en jouant sur les valeurs de départ et de fin du signal de synchronisation vertical dans les paramètres verticaux du mode graphique.

En augmentant le nombre de pixels physiques pour chaque ligne et en conservant la même résolution, la largeur de l'image visible est réduite. En effet, la proportion de la partie visible de l'image sur la largeur totale de l'écran est d'autant plus faible que le nombre de pixels physiques des lignes est grand. Le même raisonnement peut être appliqué sur les paramètres verticaux. L'image peut donc être agrandie ou réduite horizontalement et verticalement en jouant sur la longueur totale des lignes et sur la hauteur totale de l'image. Si l'on veut que l'image reste centrée, il faut également décaler les points de début et de fin des signaux de synchronisation, pour que la proportion du blanking placée de part et d'autre de l'image reste constante.

Il faut bien comprendre que la modification des paramètres de synchronisation et de la longueur totale de la ligne sont soumis à de fortes contraintes. Il faut s'assurer que les durées des signaux de synchronisation et du blanking restent dans les limites de ce que peut accepter le moniteur. De plus, le nombre de points balayés au total est évidemment le produit de la longueur totale des lignes et du nombre de lignes total du mode. À fréquence de base fixe, ceci revient à définir le taux de rafraîchissement, puisque plus il y a de pixels à balayer dans chaque image, moins il est possible d'afficher d'images par seconde. Ceci n'est pas gênant si vous pouvez augmenter la fréquence de base. Par contre, si vous avez atteint la limite de votre carte graphique, ou la limite de bande passante de votre moniteur, vous serez limité dans le taux de rafraîchissement si vous devez réduire les dimensions de l'image affichée.

Vous devez vous en doutez à présent : l'écriture d'une ligne de mode est un sport très difficile. La technique à utiliser est itérative. Vous pouvez tourner en rond pendant un certain temps, si vous ne savez pas vous y prendre. La première des choses à faire est de choisir la dimension de l'image que vous désirez obtenir. Cette dimension va fixer la largeur totale et le nombre de lignes total qui seront utilisés pour ce mode graphique. Vous pouvez toujours partir des lignes de mode utilisées pour les modes graphiques VESA pour un moniteur semblable au vôtre. Une fois que vous aurez ces valeurs, vous pourrez essayer de faire monter le taux de rafraîchissement en augmentant la fréquence de base. Vous aurez peut-être à redimensionner et à recentrer l'image sur le moniteur. Les quatre contraintes à respecter lorsque vous augmentez la fréquence de base sont bien entendu la durée du blanking horizontal et vertical, la durée des signaux de synchronisation, et les plages de fréquences horizontales et verticales acceptées par votre moniteur. Il est probable que vous aurez à accroître l'écart entre les valeurs de début et de fin des signaux de synchronisation, car leur durée peut baisser dangereusement lorsque la fréquence de base augmente. Malheureusement, ceci peut vous limiter dans la possibilité de centrer l'image sur votre moniteur. Une image dont les bords ne sont pas rectilignes ou pas nets est un signe indiquant que le signal de synchronisation n'est pas suffisamment long. Si vous ne vous

en sortez pas, c'est que vous essayez d'atteindre un taux de rafraîchissement trop élevé.

Si vous désirez fabriquer un mode graphique de résolution non standard, vous ne trouverez pas forcément une ligne de mode pouvant servir de point de départ. Dans ce cas, il va vous falloir créer cette ligne de mode ex-nihilo. Vous trouverez dans le fichier de documentation `VideoModes.doc` de XFree86 les explications permettant de créer une ligne de mode correcte à partir des caractéristiques de votre moniteur. Pour les moniteurs multisynchrones, la technique suivante peut être utilisée.

La première étape est de trouver la bande passante de votre moniteur et les plages de fréquences horizontales et verticales utilisables par votre moniteur. Ces données sont très souvent fournies dans leur documentation. Par exemple, pour un moniteur Sony Multiscan 100ES, ces plages de fréquences sont les suivantes :

	Horizontales (kHz)	Verticales (Hz)
Fréquence minimum	30	50
Fréquence maximum	70	120

La deuxième étape est de choisir la résolution que l'on désire utiliser. Supposons que l'on veuille utiliser une résolution intermédiaire entre 1024x768 et 800x600 sur ce type d'écran 15 pouces. Prenons par exemple 904x678. Notez que la résolution horizontale est divisible par 8, et que la résolution verticale est égale aux trois quarts de cette dernière. Ce rapport permet simplement de conserver la proportion de 4/3 des écrans d'ordinateurs. Pour pouvez cependant choisir un autre rapport si vous le désirez.

Vous devez ensuite déterminer les longueurs totales des lignes en pixels et le nombre total de lignes, blanking compris. C'est cette partie qui est la plus difficile, elle nécessite de connaître les durées minimales des signaux de synchronisation horizontaux et verticaux. Ces données sont hélas rarement fournies par les fabricants. Heureusement, on peut s'en passer en pratique, car tous les écrans cathodiques utilisent la même technologie. En général, la longueur totale d'une ligne est égale à la résolution horizontale divisée par 0,8, et le nombre total de ligne est égal à la résolution verticale augmentée de 5%. Pour notre mode graphique, le calcul est donc le suivant :

```
longueur totale des lignes = 904 pixels / 0,8 = 1130 pixels
hauteur totale de l'image = 678 lignes * (1+5/100) =
    678 lignes * 1,05 = 711,9 lignes.
```

La longueur totale des lignes n'est pas multiple de 8, nous devons donc l'arrondir à 1136. De même, la hauteur totale de l'image doit être arrondie à 712 lignes.

Nous disposons à présent des premières et dernières valeurs des paramètres horizontaux et verticaux de la ligne de mode. Les paramètres intermédiaires, qui fixent le début et la fin des signaux de retour de balayage, doivent être choisis de telle sorte que ces signaux commencent un peu après le début du blanking et se terminent un peu avant le début de la génération de l'image suivante. En pratique, on peut couramment considérer qu'une marge de 32 à 40 pixels est suffisante pour les signaux de synchronisation horizontaux, et qu'il faut 0 à 10 lignes pour les signaux de synchronisation verticaux. L'écart entre le début et la fin de ces signaux doit être suffisamment grand pour garantir leurs durées

minimales exigées (mais a priori inconnues) par votre moniteur. Dans notre exemple, nous pouvons donc choisir les paramètres suivants :

```
    HTimings 904 944 1096 1136
    VTimings 678 683 707 712
```

Il reste à déterminer la fréquence de base à utiliser. Le but est bien entendu d'obtenir le taux de rafraîchissement le plus élevé possible, donc la fréquence de base la plus élevée, tout en restant dans les limites de la carte graphique et du moniteur. La contrainte la plus forte en générale est la fréquence maximale de balayage horizontal.

Dans notre cas, l'écran est capable de travailler à 70 kHz au plus, c'est à dire d'afficher 70000 lignes par secondes. Comme notre mode graphique utilise 712 lignes physiques au total, ceci nous donne un taux de rafraîchissement de 70000/712, soit environ 98 images par secondes. Bien entendu, il faut contrôler que cette fréquence est bien dans la plage de fréquences verticales du moniteur. C'est bien le cas ici, puisqu'il peut supporter des fréquences allant jusqu'à 120 Hz. En général, il faut prendre la plus petite des valeurs comme taux de rafraîchissement de base.

Une fois le taux de rafraîchissement déterminé, nous devons calculer la fréquence de base à utiliser. Pour afficher 98 images de 710 lignes de 1125 pixels par secondes, nous calculons cette fréquence de la manière suivante :

```
98 * 712 * 1136 = 79,26 MHz
```

Encore une fois, il faut contrôler que la carte graphique peut générer un signal de cette fréquence et que celle-ci est bien inférieure à la bande passante du moniteur. Dans le cas contraire, il faudrait encore une fois se limiter à la plus petite de ces valeurs. La plupart des cartes graphiques récentes peuvent monter au moins jusqu'à 150MHz, et l'écran Sony Multiscan 100ES a une bande passante bien supérieure à cette valeur. Nous pouvons donc conserver cette fréquence de base, et obtenons donc la ligne de mode suivante :

```
Mode "904x678"
    DotClock 79.26
    HTimings 904 944 1096 1136
    VTimings 678 683 707 712
EndMode
```

Cette ligne de mode a peu de chances d'être satisfaisante sans retouches, mais elle peut servir de point de départ pour la recherche de la ligne de mode idéale. En général, les paramètres étant choisis proches de l'optimum, l'image sera bien trop grande ou déformée sur les bords. Vous devrez donc faire des ajustements manuels pour vous rapprocher, petit à petit, d'une ligne de mode utilisable. Vous pourrez utiliser l'utilitaire xvidtune pour cela. Cet utilitaire est un utilitaire fourni avec XFree86, qui permet de modifier les dimensions et la position de l'image interactivement. Cet utilitaire fonctionne directement sous X Window, et permet donc d'avoir un aperçu de l'image avec les paramètres modifiés. Vous trouverez une description plus détaillée de la manière d'utiliser xvidtune dans la Section 10.2.6.

Seul le redimensionnement de l'image modifie le nombre total de lignes et leur longueur totale. En particulier, la réduction de la taille de l'image accroît la taille totale des lignes et leur nombre, ce qui entraîne la baisse du taux de rafraîchissement. À chaque fois que vous réduisez l'image, vous pourrez recalculer le taux de rafraîchissement et la fréquence de base nécessaire pour atteindre ce taux, tout en veillant à ne pas dépasser ni les limites de votre carte graphique, ni la bande passante de votre moniteur.

Ainsi, après quelques modifications de la ligne de mode dans xvidtune et réajustements de la fréquence de base, la ligne de mode suivante peut être obtenue :

```
Mode "904x678"
    DotClock 79.82
    HTimings 904 920 1104 1144
    VTimings 678 679 710 712
EndMode
```

Cette ligne de mode permet d'obtenir un affichage correct en 904 par 678, avec un taux de rafraîchissement de 97.99 Hz, ce qui est plus qu'honorable. On constate que la première ligne de mode utilisait une durée de balayage vertical trop petite, ce qui provoquait des déformations sur les bords de l'image.

Si vous voulez approfondir le sujet, vous pouvez lire les formules mathématiques donnant les relations entre les différents paramètres techniques influant sur la génération des images dans l'Annexe B. Ces formules sont données à titre indicatif, elles ne vous donneront pas les caractéristiques techniques de votre écran. Encore une fois, je ne saurais trop vous recommander d'utiliser les outils fournis avec votre distribution…

10.2.4.8. Sections « Modes »

Généralement, les lignes de modes sont définies pour un écran donné, car l'apparence de l'image dépend des caractéristiques de chaque écran. Par conséquent, ces lignes sont définies la plupart du temps dans les sections « `Monitor` ». Cependant, il peut être utile de regrouper toutes les lignes de modes d'un écran donné dans une section à part, afin de structurer et d'organiser le fichier `XF86Config`.

Pour cela, on écrira des sections « `Modes` », qui n'auront donc pas d'autre rôles que de contenir une ou plusieurs lignes de modes, et qui seront référencées dans les sections « `Monitor` » des moniteurs qui désirent utiliser ces lignes de modes. L'intérêt de regrouper toutes les lignes de modes dans une section « `Modes` » apparaît clairement dès que plusieurs sections « `Monitor` » utilisent les mêmes lignes de modes. Ce peut être le cas pour des moniteurs d'un même fabricant ou possédant des caractéristiques techniques similaires.

Le format des sections « `Monitor` » est très simple, puisqu'elles ne contiennent qu'un identificateur, introduit comme à l'accoutumée par le mot clef « `Identifier` » suivi du nom de la section entre guillemets, et les lignes de modes elles-mêmes. La syntaxe de ces lignes est exactement la même que celle qui est utilisée dans la section « `Monitor` » et ne sera donc pas décrite plus en détail ici. Vous trouverez ci-dessous un exemple de sections « `Modes` » :

```
# Exemple de section Modes :
Section "Modes"
    Identifier "Modes Standards"
    Modeline "640x480"    50.00 640 656 720 832 480 480 489 501
    Modeline "800x600"    72.80  800 816 928 1040 600 600 610 626
    Modeline "1024x768"  92.00 1024 1076 1252 1344 768 772 782 806
EnsSection
```

10.2.4.9. Sections « Screen »

Les sections « `Screen` » permettent de regrouper les moniteurs et les cartes graphiques pour effectuer l'affichage des écrans d'un display, et d'indiquer les modes graphiques que l'on désire utiliser parmi tous les modes que le moniteur indiqué sait gérer, pour chaque profondeur de couleur.

Les sections « `Screen` » sont, comme les sections « `Monitor` », consitutées de deux parties. La première partie spécifie les informations générales de l'écran, à savoir essentiellement la carte graphique et le moniteur à utiliser. La deuxième partie quant à elle permet de préciser, pour chaque profondeur de couleur utilisable avec cet écran, les modes résolutions des modes graphiques utilisables ainsi que les paramètres des résolutions virtuelles. Voici trouverez ci-dessous un exemple de section « `Screen` » :

```
Section "Screen"
    Identifier "Ecran 1"
    Device      "Carte 1"
    Monitor     "Moniteur 1"
    DefaultDepth 24
    SubSection "Display"
        Depth  24
        Modes  "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth  16
        Modes  "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth  8
        Modes  "1024x768" "800x600" "640x480"
    EndSubSection
EndSection
```

L'option « `Identifier` » permet, comme d'habitude, de nommer la section « `Screen` » courante. Le nom doit être indiqué entre guillemets, à la suite du mot-clé « `Identifier` ». C'est ce nom qui sera utilisé dans les sections « `ServerLayout` » pour référencer les écrans du display. Le mot clef « `Device` » indique la section « `Device` » à utiliser pour obtenir les informations sur la carte graphique qui gère l'affichage de cet écran. Il doit être suivi du nom qui a été spécifié après le mot clef

« `Identifier` » dans la section « `Device` » correspondante. De même, le mot clef « `Monitor` » indique la section « `Monitor` » à utiliser pour obtenir les informations sur le moniteur courant. Enfin, le mot clef « `DefaultColorDepth` » indique la profondeur de couleur utilisée par défaut au démarrage de XWindow. Comme nous le verrons plus loin, une autre profondeur de couleur que celle-ci pourra être utilisée en passant un argument sur la ligne de commande du serveur X.

Comme vous pouvez le voir, la section « `Screen` » contient une ou plusieurs sous-sections « `Display` ». Ce sont ces sous-sections qui indiquent les paramètres de l'écran spécifiques à chaque profondeur de couleur utilisable. Les sous-sections « `Display` » contiennent les informations suivantes :

- la profondeur de couleur considérée, exprimée en bits par pixels après le mot clef « `Depth` » ;
- la liste des modes graphiques utilisables avec cette profondeur de couleur, référencés par le nom de leur ligne de mode, après le mot clef « `Modes` ». Plusieurs modes peuvent être spécifiés, en donnant leurs noms respectifs, séparés par des espaces.
- des informations complémentaires concernant les résolutions virtuelles pour les modes dans lesquels la résolution logique de l'écran est supérieure à la résolution réellement utilisée. Ces paramètres indiquent la résolution logique de l'écran virtuel, ainsi que le positionnement de l'image affichée par l'écran physique dans cet écran virtuel. Vous trouverez de plus amples informations sur ces options dans la page de manuel `XF86Config`.

10.2.4.10. Sections « ServerLayout »

Le dernier type de section, qui regroupe toutes les informations des sections vues précédemment, est le type de sections « `ServerLayout` ». Comme son nom l'indique, les sections de ce type fournissent les informations concernant la disposition des écrans d'un display. Cependant, dans la plupart des cas, les displays n'ont qu'un seul écran, et il est évident que la disposition de cet écran est triviale dans ce cas.

Il peut exister plusieurs sections « `ServerLayout` » dans le fichier XF86Config. En effet, ce fichier peut être utilisé par plusieurs serveurs X qui peuvent tourner simultanément sur la même machine (que l'on ait plusieurs cartes graphiques ou non). Chaque serveur X peut donc utiliser une section « `ServerLayout` ». La section utilisée par défaut est la première « `ServerLayout` » que le serveur X trouvera dans le fichier `XF86Config`, mais il est possible de lui demander d'en utiliser une autre à l'aide d'une option en ligne de commande. Toutefois, encore une fois, la plupart des machines ne disposent que d'un seul écran, et même si l'on lance plusieurs serveurs X, il n'y a ici besoin que d'une seuls section « `ServerLayout` ».

Les sections « `ServerLayout` » indiquent surtout au serveurs X quels sont les périphériques d'entrée à utiliser (au moins un clavier et une souris) en plus de l'écran ou les écrans du display. Ce sont donc ces sections qui définissent complètement un display donné. L'exemple donné ci-dessous présente une section « `ServerLayout` » simple :

```
Section "ServerLayout"
    Identifier "Serveur 1"
```

```
    Screen "Ecran 1"
    InputDevice "Clavier 1" "CoreKeyboard"
    InputDevice "Souris 1" "CorePointer"
    Option "BlankTime" "5"
EndSection
```

Les sections « `ServerLayout` » disposent d'un nom qui permet de les identifier. Ce nom est le nom qui sera utilisé dans la ligne de commande du serveur X si plusieurs sections « `ServerLayout` » sont définies. Il doit être indiqué, comme d'habitude, à la suite du mot clef « `Identifier` », entre guillemets.

Les sections « `ServerLayout` » doivent ensuite indiquer au moins un écran pour le display qu'elles décrivent. Les écrans sont introduits à l'aide du mot clef « `Screen` », suivi de l'identificateur de la section « `Screen` » à utiliser. Si plusieurs écrans doivent être utilisés, il est possible de spécifier la position des écrans les uns par rapport aux autres à l'aide d'options complémentaires, qui sont spécifiées à la suite de l'identificateur de l'écran. Ces options permettent de positionner les écrans relativement les uns aux autres, ou de manière absolue en précisant leurs coordonnées dans le système de coordonnées de l'écran virtuel. Les options les plus simples à utiliser sont les options `RightOf`, `LeftOf`, `Above` et `Below`, qui permettent d'indiquer respectivement que l'écran courant est situé à droite, à gauche, au dessus ou en dessous de l'écran dont l'identificateur est spécifié juste après. Par exemple, si l'on veut rajouter un deuxième écran à la section précédente, et que cet écran doit être situé à droite du premier écran, on ajoutera simplement la ligne suivante :

```
    Screen "Ecran 2" RightOf "Ecran 1"
```

La section « `ServerLayout` » doit ensuite contenir au moins une référence à un clavier et une référence à un périphérique de pointage. Ces références sont introduites à l'aide du mot clef « `InputDevice` », suivie de l'identificateur de la section « `InputDevice` » à utiliser. Le serveur X doit interpréter les événements provenant des différents périphériques d'entrée correctement. Il faut donc lui indiquer la nature du périphérique à l'aide de l'option `CorePointer` pour la souris principale et de l'option `CoreKeyboard` pour le clavier principal. Notez qu'il est possible plus d'un clavier et d'une souris, mais dans ce cas, les événements des différents périphériques se mélangeront et l'installation sera franchement inutilisable…

Enfin, comme vous pouvez le voir dans l'exmple donné ci-dessus, il est possible d'utiliser des options spécifiques pour chaque section « `ServerLayout` ». Ces options sont les mêmes que celles utilisées dans la section « `ServerFlags` ». Vous pouvez donc ici spécifier une autre valeur que celles indiquées dans cette section, afin de préciser le comportement d'un serveur spécifique. Dans l'exemple donné ci-dessus, l'économiseur d'écran a été réglé pour s'activer au bout de 5 minutes.

10.2.5. Informations utilisées lors du démarrage de XFree86

Les informations stockées dans le fichier de configuration `XF86Config` sont utilisées par XFree86 pour déterminer la manière dont l'affichage doit se faire. Lorsque le serveur X démarre, il commence

par rechercher la section « `ServerLayout` » qu'il doit utiliser pour déterminer les principales options du display. Par défaut, il utilisera systématiquement la première section « `ServerLayout` » du fichier `XF86Config`. Cependant, il est possible de lui demander d'en utiliser une autre, à l'aide de l'option de ligne de commande `-layout`, suivie de l'identificateur de la section à utiliser.

Le serveur X initialise alors tous les écrans du display et prend en charge les périphériques d'entrée. Pour chaque écran, il choisit la profondeur de couleur indiquée dans la variable « `DefaultDepth` » de la section « `Screen` » et tente de l'utiliser. Cependant, il est possible de choisir une autre profondeur de couleur à l'aide de l'option `-depth`, suivie de la profondeur de couleur à utiliser, spécifiée en bits par pixels. Remarquez que dans le cas des configurations à plusieurs écran, il n'est possible de spécifier qu'une seule profondeur de couleur. Il faut donc que toutes les sections « `Screen` » des écrans utilisés contiennent une sous-section « `Display` » pour la profondeur de couleur choisie.

Une fois la profondeur de couleur déterminée, le serveur X choisit pour chaque écran la première résolution indiquée dans la liste des modes de la sous-section « `Display` » dont la profondeur de couleur correspond à celle utilisée. Il tente alors de passer l'adaptateur graphique dans ce mode graphique, en utilisant pour cela la ligne de mode correspondante à cette résolution pour piloter le moniteur.

Il est possible, lorsque XWindow fonctionne, de changer cette résolution avec les combinaisons de touches `CTRL+ALT+'+'` et `CTRL+ALT+'-'` (`'+'` et `'-'` doivent être les touches + et - du pavé numérique). Ces deux combinaisons de touches permettent respectivement de passer d'une résolution à la suivante ou à la précédente dans la liste des modes spécifiés dans la sous-section « `Display` ». Pour chaque mode choisi, la ligne de mode correspondante à la résolution de ce mode est utilisée pour générer les signaux à destination du moniteur.

> **Note:** La résolution virtuelle de l'écran n'est pas modifiée lorsque l'on change de résolution physique. Elle est toujours égale à la plus grande des résolutions disponibles dans la section « `Display` » utilisée. Seule la résolution physique de l'affichage est modifiée. Vous pourrez déplacer la zone d'affichage en déplaçant la souris, et faire défiler ainsi tout l'écran.
>
> Il est impossible de changer la profondeur de couleur utilisée sans redémarrer le serveur X. Les seuls changements acceptés sont ceux concernant les résolutions graphiques listées dans la sous-section « `Display` » pour la profondeur de couleur courante.

10.2.6. Utilisation de xvidtune

xvidtune est un petit utilitaire permettant de régler les paramètres d'affichage de chaque mode graphique. Grâce à lui, vous pourrez ajuster la taille de l'image horizontalement et verticalement, ainsi que sa position par rapport aux bords du moniteur. xvidtune se lance simplement, en tapant son nom en ligne de commande :

```
xvidtune
```

Dès qu'il démarre, il commence par afficher une fenêtre d'avertissement, pour vous prévenir qu'un usage par une personne non avertie peut provoquer de sérieux dommages aussi bien au moniteur qu'à l'utilisateur. En effet, il permet de modifier les paramètres de synchronisation horizontale et

verticale du moniteur pour ajuster l'image. Si vous choisissez de mauvais paramètres, vous pouvez sortir des spécifications techniques de votre moniteur. Ceci pourrait fort bien l'endommager, et, ce qui est plus grave, le conduire à émettre un rayonnement X dangereux pour vous. Fort heureusement, la plupart des moniteurs modernes disposent de mécanismes de sécurité qui les empêche de générer l'image lorsque ces paramètres sont incorrects. Par conséquent, il est assez rare d'avoir des problèmes avec cet utilitaire, d'autant plus qu'il effectue des contrôles avant toute tentative douteuse. Vous pouvez donc valider en toute confiance, mais soyez malgré tout averti des risques potentiels. Rappelons que le serveur X peut être arrêté à n'importe quel moment à l'aide de la séquence de touches `CTRL+ALT+BACKSPACE`.

L'écran de xvidtune se compose de quatre parties. La partie supérieure gauche permet de régler les paramètres horizontaux du mode graphique, à savoir la largeur de l'image et sa position par rapport aux bords verticaux du moniteur. La partie supérieure droite permet de régler les paramètres verticaux, à savoir la hauteur et la position par rapport aux bords horizontaux du moniteur. La partie inférieure droite contient les informations essentielles sur le mode vidéo courant. Enfin, la partie inférieure gauche contient les boutons permettant de choisir les actions à effectuer.

Les paramétrages horizontaux et verticaux du mode graphique peuvent être modifiés très simplement. Les boutons « `Left` » et « `Right` » permettent de centrer l'image horizontalement, et les boutons « `Wider` » et « `Narrower` » permettent d'ajuster sa largeur. De même, les boutons « `Up` » et « `Down` » influent sur le centrage vertical, et les boutons « `Shorter` » et « `Taller` » ajustent sa hauteur.

Vous pouvez tester les modifications apportées à tout instant à l'aide du bouton « `Test` ». Il est même recommandé d'essayer régulièrement les paramètres choisis, et de procéder pas à pas. Lorsque les paramètres conviennent, ils peuvent être appliqués immédiatement à l'aide du bouton « `Apply` ». Si vous désirez générer une ligne mode valide pour la configuration courante, par exemple afin de modifier votre fichier de configuration `XF86Config`, vous pouvez cliquer sur le bouton « `Show` ». La ligne de mode sera affichée sur le terminal à partir duquel vous avez lancé xvidtune, vous pourrez la recopier dans le fichier `XF86Config` à la place de la ligne de mode du mode graphique en cours d'utilisation.

Il est possible de changer le mode graphique courant en cliquant sur les boutons « `Next` » et « `Prev` ». Ils permettent de passer en revue tous les modes graphiques qui ont été enregistrés dans le fichier `XF86Config` par XF86Setup. Une fois que vous aurez ajusté tous les modes graphiques et que vous les aurez enregistrés à l'aide du bouton « `Apply` », vous pourrez quitter xvidtune en cliquant sur le bouton « `Quit` ». Notez que le bouton « `Apply` » validez les choix en cours, mais ne les enregistre pas dans le fichier `XF86Config`. Vous devez reporter manuellement les paramètres du mode courant, tels qu'ils sont présentés par la commande « `Show` », dans le fichier `XF86Config`.

10.3. Utilisation du driver frame buffer du noyau

Si par malheur votre carte graphique n'est gérée par aucun des drivers de XFree86 (cas relativement exceptionnel), vous serez sans doute obligé d'utiliser le driver `vesa`. Ce driver permet d'utiliser toutes les cartes compatibles avec le standard VESA 2.0 (c'est à dire la plupart des cartes graphiques, mais il existe des exceptions notables).

Il existe une alternative à ce driver, qui se base sur les fonctionnalités « `frame buffer` » du noyau de Linux. Cette fonctionnalité permet d'utiliser Linux complètement en mode graphique, en fournissant un accès linéaire direct à la mémoire vidéo de la carte graphique grâce au fichier spécial de périphérique `/dev/fb0`. Il existe un driver XFree86 pour le frame buffer du noyau, qui permet donc de démarrer le serveur X en s'appuyant complètement sur le noyau. L'avantage du frame buffer du noyau est que même les consoles en mode texte feront leur affichage en mode graphique (ceci est aussi un inconvénient du point de vue des performances). En revanche, vous ne pourrez pas changer de résolution une fois le système aura démarré.

Pour accéder à la mémoire vidéo, le noyau se base également sur l'interface de programmation du standard VESA 2.0, qui est géré par le BIOS de la plupart des cartes graphiques récentes. Ceci signifie également que vous ne disposerez pas d'accélération matérielle en général, sauf pour quelques cartes graphiques courantes reconnues par le noyau.

10.3.1. Configuration du noyau et installation du driver

La mise en ½uvre du driver frame buffer se fait évidemment dans la configuration du noyau. Les options à activer sont toutes dans le menu « `Console drivers` ». En plus de l'option « `VGA text console` », vous devez impérativement activer « `Video mode selection support` ». Cette option vous permettra de choisir le mode VESA à utiliser lors du démarrage de l'ordinateur. Vous devrez également cocher l'option « `Support for frame buffer devices (EXPERIMENTAL)` » (cette option ne vous sera proposée que si vous avez validé l'option « `Prompt for development and/or incomplete code/drivers` » du menu « `Code maturity level options` »). Les options suivantes du driver frame buffer devront également être activées :

- « `VESA VGA graphics console` » (cette option permet d'utiliser les modes graphiques VESA indiqués au démarrage pour l'affichage de la console) ;
- « `Advanced low level driver options (NEW)` » (cette option vous permet de préciser les structures de la mémoire vidéo que les différents modes graphiques sont susceptible d'utiliser) ;
- « `8 bpp packed pixels support` », « `16 bpp packed pixels support` », « `24 bpp packed pixels support` » et « `32 bpp packed pixels support` » (ces options correspondent aux différentes structures de la mémoire vidéo qui sont utilisées par les modes graphiques VESA) ;
- « `Select compiled-in fonts (NEW)` » (cette option vous permet de choisir les polices de caractères qui seront utilisées par la console) ;
- « `VGA 8x8 font` » et « `VGA 8x16 font` » (ces deux polices sont les polices standard utilisées par la console).

Il faut ensuite vérifier que le fichier spécial de périphérique `/dev/fb0` a été créé par le programme d'installation de votre distribution. Si ce n'est pas le cas, vous devez le créer à l'aide de la commande **mknod**. Le numéro de périphérique majeur de ce fichier est 29. Le numéro de fichier mineur à utiliser est le numéro du périphérique. Par exemple, le fichier spécial de périphérique `/dev/fb0` porte les

numéros 29 et 0, le fichier `/dev/fb1` porte les numéros 29 et 1, etc... Ces fichiers sont tous de type caractère, la ligne de commande pour créer un de ces fichiers est donc la suivante :

```
mknod fbn c 29 n
```

où `n` est le numéro du fichier spécial de périphérique à créer.

Il est également recommandé de créer un lien symbolique `/dev/fb` vers `/dev/fb0` afin d'assurer la compatibilité avec de vieux programmes utilisant ce nom pour accéder au fichier spécial de périphérique du driver frame buffer.

Une fois ces opérations réalisées, vous devez compiler le noyau et l'installer, en suivant la méthode décrite dans la partie décrivant la compilation du noyau. Lors du redémarrage du système, vous pourrez passer l'option suivante au noyau pour lui préciser le mode graphique VESA à utiliser :

```
vga=mode
```

où `mode` est le numéro du mode graphique désiré. Les numéros valides sont indiqués dans le tableau donné ci-dessous :

Tableau 10-2. Numéros des modes graphiques VESA

Couleurs	**Résolution**				
	640x480	**800x600**	**1024x768**	**1280x1024**	**1600x1200**
256	**769**	**771**	**773**	**775**	**796**
32768	**784**	**787**	**790**	**793**	**797**
65536	**785**	**788**	**791**	**794**	**798**
16,8M	**786**	**789**	**792**	**795**	**799**

Si tout se passe correctement, votre système devrait démarrer dans le mode graphique indiqué et afficher le logo de Linux (un pingouin nommé « Tux », pour ceux qui ne le sauraient pas encore). Lorsque vous aurez déterminé le mode graphique qui vous convient, vous pourrez modifier le fichier de configuration de Lilo et spécifier le numéro de ce mode dans la ligne « `vga=...` ». De cette manière, votre système redémarrera automatiquement dans ce mode graphique.

10.3.2. Configuration du serveur X

Les manipulations précédentes n'ont pas grand intérêt si vous ne désirez travailler qu'avec la console. En effet, l'affichage en mode graphique est beaucoup plus lent que l'affichage en mode texte, et l'affichage du pingouin Tux au démarrage ne vous apportera pas grand chose. C'est pour cela que l'étape suivante est normalement de configurer le serveur X de XFree86 pour le driver frame buffer, afin d'utiliser l'environnement graphique XWindow et son système de fenêtrage.

La configuration du serveur X est élémentaire. Il faut avant tout s'assurer que l'on dispose bien du

driver permettant au serveur X d'utiliser l'interface `/dev/fb0`. Ce driver se nomme `fbdev`, et utilise un autre module spécifique au système d'exploitation nommé `fbdevhw`. Il faut ensuite de modifier ou créer le fichier `XF86Config` pour utiliser ce driver. Les seules sections à modifier pour utiliser le driver frame buffer sont la section « `Device` » et la section « `Screen` ».

La section « `Device` » est réduite à sa plus simple expression, puisque tous les paramètres sont fixés par le mode VESA choisi au démarrage d'une part, et parce que le serveur X ne saurait pas les exploiter d'autre part. Il suffit donc simplement d'indiquer que le driver à utiliser est le driver `fbdev`, et de donner l'adresse de la carte vidéo sur le bus à l'aide du mot clef « `BusID` » :

```
Section "Device"
    Identifier "Carte 1"
    Driver "fbdev"
    BusID  "PCI :1 :5 :0"
EndSection
```

Vous pourrez déterminer l'adresse de votre carte graphique à l'aide de la commande **lspci**, ou en demandant au serveur X de scanner les bus PCI en lui passant l'option `-scanpci` en paramètre :

```
XFree86 -scanpci
```

La section « `Screen` » est elle aussi très simplifiée, puisque le seul mode graphique utilisable est le mode choisi au démarrage de la machine. La liste des modes utilisables peut donc être franchement omise, ou se réduire à la valeur spéciale « `default` » :

```
Section "Screen"
    Device    "Carte 1"
    Monitor   "Moniteur 1"
    DefaultDepth 16
    SubSection "Display"
        Depth 16
        Modes "default"
    EndSubSection
EndSection
```

Notez qu'il est impératif que la profondeur de couleur de la sous-section « `Display` » soit la même que celle du mode VESA indiqué au démarrage. Prenez garde à ne pas utiliser une profondeur de couleur trop élevée, car cela dégraderait encore un peu plus les performances. Par ailleurs, comme aucun mode n'est spécifié dans la section « `Screen` », les lignes de modes des sections « `Monitor` » sont à présent facultatives. Ces sections peuvent donc être simplifiées également.

Une fois ces modifications réalisées, vous devrez pouvoir démarrer XWindow simplement avec la commande **startx**. Vous disposerez alors de toutes les fonctionnalités de XWindow, avec des performances quelques peu inférieures à celles que vous auriez avec un serveur X adapté à votre carte graphique. Il est conseillé de suivre l'actualité de XFree86 afin de savoir si un tel serveur est en cours de développement, et si oui, d'en récupérer une version finale dès que possible.

10.4. Configuration des terminaux X

Nous avons vu dans le chapitre de configuration du système de base que la connexion des utilisateurs se faisait par l'intermédiaire d'un terminal. La plupart des terminaux sont souvent en mode texte, mais il est également possible d'utiliser des terminaux graphiques sous XWindow. Ce type de terminaux est logiquement appelé « terminaux X ».

Si le niveau d'exécution par défaut de votre système est le niveau associé au démarrage sous XWindow (ce niveau est en général 3, 4 ou 5 selon les distributions), XWindow est lancé dès le démarrage de la machine et la demande de connexion se fait via un écran graphique. Dans ce cas, vous vous connecterez directement par l'intermédiaire de ce terminal X. Le principe de fonctionnement des terminaux X n'est pas exactement le même que celui que nous avons vu pour les terminaux virtuels (voir la Section 6.9). En effet, les terminaux X ne sont pas gérés par les processus getty et login classiques, mais par un programme spécifique nommé « xdm » (abréviation de l'anglais « X Display Manager »).

10.4.1. Principe de fonctionnement de xdm

Contrairement à ce qu'il fait avec les processus getty, init ne lance qu'un script d'initialisation de XWindow lorsqu'il passe dans le niveau d'exécution correspondant. Ce script a pour tâche essentiellement le démarrage du démon xdm. C'est ce démon qui est en charge de gérer les connexions sur tous les serveurs X qu'il peut trouver, en proposant la fenêtre de login et en permettant ainsi à l'utilisateur de s'identifier et s'authentifier.

En général, xdm lance lui-même le serveur X de la machine locale, et surveille son exécution. Lorsque l'utilisateur se déconnecte, le serveur X se réinitialise, et xdm affiche une nouvelle fenêtre de connexion. Cependant, xdm est également capable d'utiliser des serveurs X distants (qui sont donc déjà lancés), afin de permettre la connexion d'utilisateurs sur des terminaux X déportés. Enfin, il est possible de le configurer pour qu'il se signale sur un réseau, afin que les serveurs X fonctionnant de ce réseau puisse lui demander une connexion. Les serveurs X capable de se connecter à xdm de cette manière doivent comprendre le protocole XDMCP (abréviation de l'anglais « XDM Control Protocol »). Comme on le voit, la souplesse de ce mécanisme est tout simplement exceptionnelle.

> **Note:** Pratiquement, le processus xdm se duplique pour chaque terminal X dont il gère la connexion. Ainsi, chaque processus xdm propose la connexion et surveille la déconnexion de l'utilisateur sur un terminal X donné. Tous les processus xdm sont lancés par le processus xdm initial qui a été lancé par les scripts de changement de niveau d'exécution. Ce mécanisme est donc complètement transparent pour l'utilisateur.
>
> La plupart des environnement graphiques modernes (comme KDE et GNOME) fournissent leur propre gestionnaire de display. Ces gestionnaires sont toutefois compatibles avec xdm, et leur configuration se fait de la même manière.

10.4.2. Configuration de xdm

Le comportement de xdm est complètement défini dans ses fichiers de configurations, qui sont en général placés dans le répertoire `/etc/X11/xdm/`. Ces fichiers décrivent chacun un des aspects du comportement de xdm. Le fichier de configuration principal est le fichier `xdm-config`, qui contient les références sur les autres fichiers de configuration.

10.4.2.1. Serveurs X locaux

La définition des serveurs X qui n'utilisent pas le protocole XDMCP et que xdm doit prendre en charge directement est réalisée dans le fichier de configuration référencé par la ligne « `DisplayManager.servers` » du fichier `xdm-config`. Par défaut, ce fichier est le fichier `/etc/X11/xdm/Xservers`. Il contient une ligne par serveur, chaque ligne ayant la syntaxe suivante :

```
display type [commande]
```

où `display` est le nom du display à utiliser, `type` est le type de serveur X (serveur local ou distant), et `commande` est la commande à exécuter pour lancer le serveur s'il est local. Le display indiqué sera celui qui sera utilisé pour définir la variable d'environnement DISPLAY, et doit donc utiliser la syntaxe normale des displays. Les deux types de serveurs disponibles sont « `local` », pour les serveurs de la machine locale que xdm doit lancer lui-même, et « `foreign` », pour les serveurs distants sur lesquels xdm doit afficher la fenêtre de demande de connexion (ils doivent donc être déjà lancés). La ligne de commande à utiliser pour le lancement du serveur ne doit être spécifiée que pour les serveurs de type « `local` ».

Les serveurs X peuvent prendre un certain nombres de paramètres en ligne de commande pour leur démarrage. Vous en trouverez la liste complète dans la page de man `Xserver` et dans la page de man `XFree86`. Les options les plus intéressantes dans le cadre du lancement du serveur X par xdm sont celles permettant de définir le display que le serveur aura en charge et le terminal virtuel qu'il devra utiliser pour effectuer son affichage. Notez bien que le display doit être le même que celui donné à xdm. Ces deux options sont passées directement à la suite du nom de l'exécutable du serveur X à lancer :

```
/usr/X11R6/bin/X display vtNN
```

où `vtNN` est le nom du terminal virtuel à utiliser (`vt01` pour `/dev/tty01`, `vt02` pour `/dev/tty02`, etc…).

Ainsi, si l'on veut faire en sorte que xdm lance automatiquement deux sessions X sur les terminaux virtuels 11 et 12, en leur affectant respectivement les displays :0.0 et :1.0, on devra placer ces deux lignes dans son fichier de configuration Xservers :

```
:0 local /usr/X11R6/bin/X :0 vt11
:1 local /usr/X11R6/bin/X :1 vt12
```

> **Note:** La spécification du terminal virtuel à utiliser est facultative. Si aucun terminal n'est indiqué, le serveur X utilisera le premier terminal qu'il trouvera. Cependant, il est plus sûr de toujours indiquer le terminal à utiliser, car le noyau ne crée les terminaux qu'à la demande, lorsqu'un

processus désire y accéder. Or les serveurs X ne cherchent pas à en créer de nouveau. Si tous les terminaux virtuels existants sont déjà utilisés, le serveur X tentera donc de s'en approprier un, sans se préoccuper du programme qui l'utilise à ce moment. Le fait d'indiquer manuellement le terminal à utiliser vous évitera donc des problèmes assez curieux, tels que ceux qui peuvent survenir si les terminaux utilisés par les serveurs X le sont pas déjà par un getty par exemple. De plus, cela permet également d'éviter, si on lance plusieurs serveurs X sur la même machine, qu'ils n'accèdent ensemble au même terminal virtuel lors de leur initialisation.

Il est recommandé par ailleurs de basculer sur chaque terminal virtuel sur lequel un serveur X est lancé avant de se connecter afin de les forcer à créer leur terminal. En effet, si vous ne procédez pas ainsi, il est possible que vous ayez le temps de vous connecter dans une session X avant que les autres terminaux X n'aient fini de s'initialiser. Lorsque ceux-ci démarreront, ils changeront le terminal virtuel courant et l'affichage de votre bureau risque d'être corrompu. Si cela se produit, ne vous inquiétez pas : votre session X n'a pas planté. Assurez-vous seulement que vous vous trouvez bien sur le bon terminal virtuel, et faites en sorte que l'écran soit rafraîchi (par exemple, changez de bureau virtuel ou déplacez une fenêtre sur l'écran).

Remarquez également que les fichiers de configuration fournis avec XFree86 ne prévoient pas l'utilisation de plus de deux sessions X par défaut. Si d'aventure vous désirez en utiliser plus, vous devrez compléter le fichier de configuration xdm-config avec les lignes suivantes :

```
DisplayManager._2.authorize:    true
DisplayManager._3.authorize:    true
etc...
```

Chaque ligne indique que le display correspondant (`:2` pour « `DisplayManager._2.authorize` », etc...) doit utiliser les mécanismes de sécurité de XWindow. Nous verrons ces mécanismes dans la section suivante.

Méfiez-vous enfin de la consommation mémoire requise par chaque session X. Les serveurs X sont des programmes gourmands en mémoire, car ils doivent manipuler des images. Les performances du système risquent donc de se dégrader sensiblement si vous lancez plusieurs serveurs X. N'oubliez pas qu'après tout, XWindow est un système de fenêtrage et que la plupart des gestionnaires de fenêtres X donnent la possibilité d'utiliser des bureaux virtuels. En pratique, deux sessions X devraient suffire, afin de se connecter sous deux utilisateurs différents (un utilisateur normal et l'utilisateur root par exemple).

Enfin, notez que le changement d'un terminal X à un autre terminal virtuel peut provoquer quelques problèmes en ce qui concerne l'état du clavier. En effet, l'affichage des diodes (Verrou numérique et Majuscule en particulier) n'est pas rétabli correctement, même si le clavier continue à fonctionner correctement. Ceci peut surprendre quelque peu la première fois que l'on rencontre ce problème. Il suffit d'appuyer deux fois de suite sur les touches `Verr Num` et `Caps Lock` (c'est à dire la touche de verrouillage des majuscules) afin de rétablir l'état correct de ces diodes. Remarquez également que les serveurs X mémorisent l'état des terminaux virtuels lors de leur démarrage, et neutralisent à chaque basculement vers un terminal en mode texte les éventuels changements de police de caractères.

10.4.2.2. Serveurs X utilisant XDMCP

Comme nous l'avons dit plus haut, xdm est également capable de se signaler sur un réseau à l'aide du protocole XDMCP, afin de permettre aux serveurs X distants de réaliser une connexion sur la machine où il est lancé. De cette manière, le lancement d'un serveur X sur une machine du réseau permettra d'obtenir la fenêtre de connexion à votre machine Linux.

xdm est capable de répondre aux demandes de connexion directes des serveurs X distants (remarquez que dans ce cas, xdm est serveur de connexion pour les serveurs X des postes clients. Faites bien attention à la terminologie client/serveur ici !). Ceci suppose que chaque poste client connaît l'adresse des machines qui utilisent xdm. Cette information peut faire partie de la configuration du poste client, mais elle peut également être déterminée dynamiquement. Pour cela, le serveur X du poste client émet une requête XDMCP en mode broadcast sur le réseau (c'est à dire à destination de tout le monde). Chaque machine sur laquelle xdm fonctionne répondra à ce client, et le serveur X proposera ainsi la liste des machines sur lesquels une connexion est réalisable via xdm. Ainsi, l'utilisateur du poste client pourra choisir le serveur sur laquelle il désire se connecter, et le processus xdm de ce serveur lui enverra la fenêtre de login.

En réalité, tous les serveurs X ne sont pas capables de gérer les réponses reçues de plusieurs processus xdm provenant de plusieurs machines à la suite de l'envoi d'une requête en mode broadcast. Par ailleurs, il n'est pas toujours utilisable d'utiliser le mode broadcast, car les paquets de ce type peuvent très bien ne pas être routés lors de la traversée d'un passerelle. Par conséquent, xdm offre également la possibilité d'interroger les machines du réseau local en réponse à une requête de connexion indirecte. Le serveur X reçoit en réponse le résultat de la requête effectuée par broadcast, et l'utilisateur peut choisir la machine à laquelle il désire accéder.

La configuration de xdm pour le protocole XDMCP se fait dans le fichier identifié par la ligne « `DisplayManager.accessFile` » du fichier `xdm-config`. Par défaut, le fichier ainsi référencé est le fichier `Xaccess` du répertoire `/etc/X11/xdm/`. La syntaxe de ce fichier est très simple. Il contient des règles indiquant le comportement de xdm lorsqu'il reçoit une requête de connexion de la part d'un serveur X. Ces règles sont définies pour des groupes de machines. Les machines sont identifiées par leur nom ou leur adresses IP, plusieurs machines pouvant être spécifiées par une même règle grâce aux caractères génériques classiques '`*`' et '`?`'. Il est également possible d'exclure une machine ou un groupe de machines en préfixant le nom du caractère de négation '`!`'.

Il existe quatre types de lignes dans le fichier `Xaccess`. Le premier type permet simplement d'indiquer que les connexions directes ou en mode broadcast provenant d'une machine sont toutes acceptées. Ce sont les lignes les plus simples, puisqu'il suffit simplement d'indiquer la machine ou le groupe de machines. Par exemple, la ligne suivante :

```
*.localnet.org
```

indique que les demandes de connexions provenant de toutes les machines du domaine « `localnet.org` » sont acceptées. Il est possible de faire en sorte que xdm ne réponde qu'aux demandes de connexions directes simplement en ajoutant le mot-clé « `NOBROADCAST` » à la suite de la ligne. Ainsi, si la machine « `termX3.localnet.org` » doit pouvoir se connecter directement, mais ne doit pas recevoir de réponse de notre machine, on ajoutera la ligne suivante dans le fichier `Xaccess` :

```
termX3.localnet.org NOBROADCAST
```

Le deuxième type de ligne permet de spécifier le comportement de xdm pour les demandes de connexions indirectes. Ces lignes contiennent toujours le nom de la machine ou du groupe de machines, mais ce nom est suivi de la liste des machines auxquelles xdm doit faire suivre la demande de connexion du serveur X. Par exemple, supposons que la machine « `termX2.localnet.org` » ne soit pas capable d'effectuer des requêtes en mode broadcast, et que l'on veuille qu'elle puisse se connecter sur les serveurs « `s5.localnet.org` », « `s11.localnet.org` » et « `s13.localnet.org` ». On devra alors ajouter la ligne suivante dans le fichier `Xaccess` :

```
termX2.localnet.org   s5.localnet.org s11.localnet.org \
    s13.localnet.org
```

Le troisième type de ligne permet de lancer un programme nommé « chooser » et qui va proposer la liste des serveurs à l'utilisateur du serveur X qui demande à se connecter. Ce programme est très utile pour les serveurs X qui ne sont pas capables de proposer ce choix à l'utilisateur. La syntaxe de ces lignes est très simple, puisqu'il suffit de faire précéder les noms des serveurs par le mot-clé `CHOOSER`. Si l'on reprend l'exemple précédent, la ligne de configuration pour le terminal X « `termX2` » deviendrait :

```
termX2.localnet.org   CHOOSER s5.localnet.org s11.localnet.org s13.localnet.org
```

La liste des serveurs qui suit peut être assez longue, et il peut être utile de faire en sorte que le programme chooser effectue une requête en mode broadcast pour le compte du serveur X qui n'en n'est pas capable. Il suffit pour cela de remplacer la liste des serveurs par le mot-clé « `BROADCAST` ». Ainsi, la ligne suivante :

```
termX4.localnet.org   CHOOSER BROADCAST
```

signifie simplement que lorsque le serveur X de la « `machine termX4.localnet.org` » se connecte au processus xdm local, celui-ci lance le programme chooser. Ce dernier lance une requête en broadcast sur le réseau afin de déterminer la liste des machines exécutant le processus xdm. Une fois qu'il a obtenu cette liste, il l'affiche sur le display géré par le serveur X qui s'est connecté. De cette manière, l'utilisateur peut choisir la machine sur laquelle il va se connecter. Le serveur X est mis en relation avec le processus xdm de cette machine, et la fenêtre de login est enfin proposée à l'utilisateur.

Le quatrième type de ligne permet de définir des « macros » de remplacement pour gérer les listes de noms de machines plus simplement. La définition d'une macro se fait avec la syntaxe suivante :

```
%nom    liste
```

où `nom` est le nom de la macro à définir, et `liste` est la liste des machines qu'elle représente. Il est possible d'utiliser des macros dans les définitions de macros.

Note: Le protocole XDMCP utilise le port 117 du protocole UDP. Vous pourrez donc ajouter la ligne suivante dans votre fichier `/etc/services`, si elle ne s'y trouve pas déjà :

```
xdmcp    117/udp
```

Notez également que vous ne pourrez pas tester votre configuration en utilisant l'interface réseau loopback. En effet, xdm vérifie la validité des adresses sources des paquets qu'il reçoit, et il refuse par défaut tout paquet provenant de l'adresse 127.0.0.1. Si vous désirez malgré tout utiliser XDMCP sans réseau réel, vous pourrez utiliser l'interface réseau dummy. Cette interface peut être créée en activant l'option « `Dummy net driver support` » du menu « `Network device support` ».

10.4.2.3. Paramétrage du serveur X pour utiliser le protocole XDMCP

L'écriture du fichier `Xaccess` représente la partie serveur du travail de configuration du protocole XDMCP. En effet, il faut encore indiquer au serveur X distant à quel processus xdm il doit tenter de se connecter lors de son démarrage. Bien entendu, ceci dépend fortement du serveur X utilisé, qui peut être quelconque. Par exemple, ce serveur X peut très bien fonctionner sur une machine Windows. Par conséquent, seule la syntaxe des serveurs de XFree86 sera décrite ici.

Les serveurs X de XFree86 utilisent des options en ligne de commande pour déterminer comment ils doivent se comporter à leur démarrage. Lorsqu'ils sont lancés directement par xdm (c'est à dire lorsqu'ils sont lancés en local, avec les options indiquées dans le fichier Xservers), ils n'ont pas besoin d'options spécifiques car ils n'utilisent pas XDMCP dans ce cas. En revanche, si l'on désire les lancer manuellement (ou dans un script d'initialisation), on devra utiliser l'une des options suivantes :

- `-query` permet de demander une connexion directe sur une machine donnée. Le nom de la machine doit être spécifié à la suite de l'option ;
- `-indirect` permet de demander une connexion indirecte sur une machine donnée. Le nom de cette machine doit être spécifié à la suite de l'option. La liste des machines qui a été indiqué dans le fichier `Xaccess` sera retournée en retour, ou le programme chooser sera lancé ;
- `-broadcast` permet de demander au serveur X d'effectuer une requête de connexion en broadcast sur le réseau.

Les serveurs de XFree86 ne sont pas capables de proposer une liste de machines à l'utilisateur. Ils prennent systématiquement la première machine qu'ils trouvent. Vous ne contrôlerez donc pas la machine sur laquelle la connexion sera faite si vous utilisez l'option `-broadcast`. C'est pour cela qu'il est recommandé d'utiliser le programme chooser dans le fichier de configuration `Xaccess` et de configurer les serveurs pour faire des demandes de connexions indirectes.

10.4.2.4. Fichiers d'initialisation de sessions

xdm fournit la possibilité d'effectuer des actions lorsque divers événements concernant la session XWindow en cours ont lieu. Par exemple, il est possible de réaliser un traitement particulier pour l'initialisation de l'environnement d'un serveur X, pour fixer l'environnement de l'utilisateur qui se connecte, et pour effectuer le ménage lorsqu'il ferme sa session. Toutes ces actions sont effectuées dans des scripts qui sont référencés par des lignes du fichier `xdm-config`.

La ligne « `DisplayManager.DISPLAY.setup` », où `DISPLAY` est le nom du display (`_0` pour `:0`, `_1` pour `:1`, etc...), référence un script qui est exécuté au nom de l'utilisateur root avant que la fenêtre de login ne soit présentée. Le fichier par défaut utilisé par XFree86 est le fichier `/etc/X11/xdm/Xsetup`. C'est donc dans ce script que vous pourrez mettre des commandes d'initialisation spécifiques pour préparer le login.

De la même manière, la ligne « `DisplayManager.DISPLAY.startup` » référence le script `Xstartup` par défaut. Ce fichier est exécuté sous le compte root juste après l'authentification de l'utilisateur. C'est ici que l'on pourra par exemple enregistrer la connexion de l'utilisateur dans le système.

La ligne « `DisplayManager.DISPLAY.session` » référence quant à elle le script `Xsession`, qui est exécuté après `Xstartup`, mais au nom de l'utilisateur cette fois. Ce script prend généralement en charge le lancement des programmes nécessaires à la gestion de la session de l'utilisateur. Par exemple, il est possible d'y placer les commandes de lancement du gestionnaire de fenêtres ou du gestionnaire de bureau.

Enfin, la ligne « `DisplayManager.DISPLAY.reset` » référence le script `Xreset` du répertoire `/lib/X11/xdm/`, qui est exécuté au nom de l'utilisateur root lorsque la session X se termine. Vous pouvez donc exécuter les tâches nécessaires pour faire le ménage à cet endroit, comme par exemple le désenregistrement de la connexion de l'utilisateur dans le système.

Tous ces scripts sont lancés sans paramètres, sauf le script `Xsession`. Ce script peut en effet recevoir des paramètres fournis par le gestionnaire de connexion (xdm, kdm ou autre), qui permettent de représenter les choix que l'utilisateur a fait pour sa connexion dans la fenêtre de login. Par exemple, le premier paramètre de ce script est souvent le nom du gestionnaire de fenêtres à utiliser. Notez que le nom « `failsafe` » référence un mode dégradé, et n'est normalement utilisé que quand les autres gestionnaires de fenêtres disponibles ne se lancent pas correctement.

Les scripts lancés au nom de l'utilisateur root (`Xsetup` et `Xreset`) sont exécutés dans un environnement minimal. La valeur de la variable d'environnement PATH pour ces scripts est déterminée par la valeur de la ligne « `DisplayManager.DISPLAY.systemPath` » du fichier de configuration `xdm-config`. De même, le script `Xsession` est exécuté avec la variable d'environnement PATH fixée à la valeur spécifiée par la ligne « `DisplayManager.DISPLAY.userPath` » du fichier `xdm-config`. Si vos scripts de gestion de connexions ne fonctionnent pas correctement, c'est sans doute un problème lié à l'environnement.

En pratique, ces fichiers de scripts de gestion de connexions sont fournis par votre distribution et vous ne devriez donc pas à avoir à y toucher. Vous pouvez toutefois y faire quelques modifications si vous estimez que les actions effectuées ne vous conviennent pas. En particulier, certaines distributions n'enregistrent pas l'utilisateur qui commence une nouvelle session. Cette opération est nécessaire, parce que xdm n'est pas un processus de login classique et ne le fait pas automatiquement (il n'utilise pas de terminal pour faire la connexion). Vous aurez pour cela à utiliser l'utilitaire **sessreg** fourni avec

XWindow. Cet utilitaire permet d'enregistrer un utilisateur avec l'option `-a` et de le supprimer avec l'option `-d`. Vous devrez donc typiquement placer une ligne telle que celle-ci :

```
sessreg -a -l $DISPLAY -x /usr/X11R6/lib/xdm/Xservers $LOGNAME
```

dans le fichier `Xstartup`, afin d'enregistrer les utilisateurs qui commencent une session XWindow, et une autre ligne telle que celle-ci :

```
sessreg -d -l $DISPLAY -x /usr/X11R6/lib/xdm/Xservers $LOGNAME
```

dans le fichier `Xreset`, afin de les supprimer lorsqu'ils terminent cette session.

10.4.3. Paramétrage des terminaux X

La configuration des terminaux X comprend un certain nombre de points, qui vont de l'ajustement du nombre de couleurs et de la résolution, à la disposition du clavier et les paramètres de la souris, en passant par l'économiseur d'écran. Heureusement, nous avons déjà vu comment nous pouvons contrôler un grand nombre de ces paramètres. Par exemple, la résolution graphique, ainsi que le nombre de couleurs et la disposition du clavier, sont définis dans le fichier de configuration général de XFree86 `XF86Config`. Nous ne verrons donc ici comment modifier des paramètres qui tiennent plus de l'ergonomie que de la configuration de base.

10.4.3.1. La commande xset

Ces paramètres peuvent souvent être fixés lors du démarrage du serveur X, ou dynamiquement avec la commande **xset**. Les paramètres que l'on peut fixer avec **xset** sont très nombreux, et seuls les plus utiles seront décrits ici. Veuillez consulter la page de manuel `xset` pour plus de détails.

L'option `dpms` permet de fixer les temps d'attente avant la mise en veille du moniteur. Sachez que la mise en veille constitue de loin le meilleur économiseur d'écran, et que les économiseurs logiciels du type ballet de lignes sont très jolis mais ne servent à rien. Pire, ils peuvent ralentir la machine, ce qui est inadmissible si vous l'utilisez en tant que serveur (heureusement, les économiseurs d'écran logiciels se lancent avec des priorités très faibles, afin de ne pas perturber les autres processus).

Cette option prend un à trois paramètres, qui représentent respectivement le temps avant le passage en mode économie d'énergie du moniteur, le temps avant le passage en mode veille, et le temps avant l'extinction complète du moniteur. Ces temps sont exprimés en secondes, la valeur nulle permettant de désactiver cette fonctionnalité. Ainsi, la commande suivante :

```
xset dpms 0 0 600
```

permet d'éteindre l'écran au bout de dix minutes d'inactivité.

Ces temps peuvent également être fixés dans le fichier de configuration `XF86Config`, où ils sont exprimés en minutes, à l'aide des mots-clés « `StandbyTime` », « `SuspendTime` » et « `OffTime` ». Ces

mots-clés doivent être placés dans la section « `ServerFlags` » du fichier de configuration. Cependant, les serveurs X n'activent ces fonctionnalités que pour les moniteurs capables de gérer le standard DPMS. Ces moniteurs doivent donc être signalés à l'aide de la ligne suivante :

```
Option "DPMS" "On"
```

dans la section « `Monitor` » qui les définit.

> **Note:** Les valeurs des temps d'attente que vous pouvez fixer dans le fichier de configuration `XF86Config` peuvent être écrasées par les valeurs définies dans la configuration de votre gestionnaire de bureau. Vous devrez donc vérifier ces paramètres si les modifications que vous faites dans le fichier `XF86Config` ne sont pas prises en compte ou si l'économie d'énergie reste désactivée malgré la présence de l'option `DPMS` dans la section « `Monitor` » de votre moniteur.

Une autre option importante de la commande **xset** est l'option `fp`, qui permet d'ajouter et de supprimer des chemins de répertoires de polices de caractères. Pour ajouter un répertoire, il suffit d'utiliser l'option `+fp` et de faire suivre le chemin de ce répertoire. La suppression d'un répertoire se fait de la même manière, avec l'option `-fp`. Par exemple, pour ajouter le répertoire de polices `/usr/X11R6/lib/X11/fonts/100dpi/` à la liste des répertoires de polices du serveur, il suffit de taper la commande suivante :

```
xset +fp /usr/X11R6/lib/X11/fonts/100dpi
```

De plus, comme nous l'avons déjà vu plus haut, les chemins des répertoires de polices de caractères peuvent être fixés statiquement à l'aide du mot-clé « `FontPath` » de la section « `Files` » du fichier de configuration `XF86Config`.

Enfin, l'option `q` de xset vous permettra de visualiser l'ensemble des paramètres en cours de validité.

10.4.3.2. Configuration de la disposition du clavier

Nous avons vu dans le chapitre de configuration du système de base le fonctionnement du clavier sous Linux. Les applications qui lisent les données provenant du driver clavier de la console peuvent travailler au niveau ASCII, au niveau keycode ou au niveau scancodes. Les serveurs X font partie des logiciels qui préfèrent la troisième solution, ce qui signifie qu'ils interprètent eux-mêmes le flux de scancodes provenant du clavier.

La raison de ce choix est que XFree86 est une implémentation portable du système XWindow pour PC. Ceci signifie qu'il est prévu pour fonctionner sur plusieurs systèmes d'exploitation de type Unix et fonctionnant sur l'architecture compatible PC. Par conséquent, il ne pouvait pas se baser sur des keycodes, qui sont évidemment dépendant du système utilisé.

L'inconvénient en revanche est que cela impose que les serveurs X définissent un mécanisme parallèle de gestion des touches du clavier, qui permette de modifier la disposition des touches et de définir le type de clavier utilisé. Plusieurs mécanismes sont utilisables, cependant, il semble que ce qui est le plus utilisé actuellement est le mécanisme « `Xkb` » (abréviation de l'anglais « X KeyBoard »).

Ce mécanisme permet de définir un grand nombre de données concernant le clavier, puisqu'outre l'emplacement des touches, il fournit la géométrie du clavier. Ainsi, toute application le désirant peut déterminer la forme du clavier et en dessiner une représentation fidèle.

Malheureusement, le protocole Xkb ne prévoit absolument pas la possibilité de définir de nouvelles touches avec leurs scancodes. La gestion des codes reçus du clavier n'est en effet pas paramétrable. Par conséquent, il est impossible d'utiliser des claviers exotiques si ceux-ci ne sont pas reconnus par XWindow.

Le protocole Xkb utilise les fichiers de configuration stockés dans le répertoire `/usr/X11R6/lib/X11/xkb/`. Ce répertoire contient un certain nombre de sous répertoires, dont chacun est relatif à un des aspects de la configuration du clavier. Le rôle des principaux sous-répertoires est détaillé dans le tableau suivant :

Nom du répertoire	Fonction
`geometry/`	Contient les fichiers de description de la géométrie des claviers. Ces fichiers peuvent être utilisés par les applications pour obtenir une représentation graphique du clavier.
`keycodes/`	Contient les fichiers de définition des « keycodes » X du clavier. Les keycodes X ont la même fonction que les keycodes de Linux : donner un nom unique à chaque touche. Cependant, ils diffèrent des keycodes de Linux en ce sens qu'ils ne sont pas définis à partir des scancodes du clavier, mais à partir de codes numériques générés par le serveur X. Ce mécanisme suppose que le serveur X connaisse tous les scancodes envoyés par les claviers. Comme il est impossible de déclarer de nouvelles séquences de scancodes au niveau du serveur X, on ne peut pas compléter les définitions de clavier existantes pour prendre en charge d'éventuelles nouvelles touches.
`symbols/`	Contient la définition du plan de clavier, ou autrement dit l'association entre les keycodes X et les symboles obtenus lors de l'appui sur les touches.
`rules/`	Contient la définition des modèles de claviers de XFree86. Ces modèles de claviers sont utilisés pour simplifier la définition des claviers dans le fichier de configuration `XF86Config.`
`keymaps/`	Contient les définitions des principaux claviers connus. Une définition de clavier comprend entre autres la définition des keycodes, des touches modificatrices et des symboles affectés aux keycodes.

Il est probable que vous n'ayez pas à modifier un seul des fichiers de ces répertoires, car ils sont correctement définis par défaut. Le seul fichier qui peut être intéressant est le fichier `fr` du sous-répertoire `symbols/`. En effet, c'est dans ce fichier que vous trouverez la définition des symboles affectés à chaque touche pour un clavier français, en fonction des modificateurs (`ALT`, `CTRL`, etc...) actifs lors de l'appui sur la touche. Le principe de fonctionnement de ce fichier est semblable aux plans de claviers de Linux : pour chaque keycode, un ensemble de symboles peuvent être définis. Ces symboles sont indiqués entre accolades (caractères '`{`' et '`}`'), en deux jeux de symboles entourés de crochets (caractères '`[`' et '`]`'). Le premier jeu de symboles indique les symboles accessibles

directement et en utilisant la touche majuscule, et le deuxième jeux définit les symboles accessibles avec la combinaison de la touche `AltGr`.

Par exemple, la touche du chiffre '`3`' au dessus des lettres '`Z`' et '`E`' est définie comme suit dans le fichier de définition du clavier français (fichier `fr`) :

```
key <AE03> { [ quotedbl,   3        ],
             [ numbersign, sterling ] };
```

Ceci signifie que la touche dont le keycode est « `<AE03>` » (c'est à dire celle la quatrième de la deuxième rangée de touches du clavier) génère les symboles « `"` » et « `3` » selon que la touche majuscule est utilisée ou non, et les symboles « `#` » et « `£` » respectivement en minuscule ou en majuscule et avec la touche `AltGr` enfoncée.

Si vous regardez le fichier du clavier français, vous constaterez que toutes les touches ne sont pas définies. La raison de ceci est que ces fichiers utilisent par défaut les affectations de touches du clavier américain standard. Par conséquent, seules les différences ont été redéfinies dans ce fichier.

Vous pourrez éventuellement modifier certaines affectations de touches si vous le désirez. Ce peut être nécessaire si vous désirez homogénéiser les combinaisons de touches entre la console Linux et XWindow. Par exemple, il peut être utile de redéfinie le comportement de la touche '`E`' pour qu'elle renvoie le symbole Euro (caractère '`¤`') en combinaison avec la touche `AltGr`. Pour cela, vous pourrez utiliser la configuration suivante dans le fichier de clavier utilisé :

```
key <AD03> { [ e,          E        ],
             [ currency,   E        ] };
```

> **Note:** Notez que la touche Euro est déjà accessible par défaut sur les claviers français avec la touche des monnaies (Livre Sterling, Dollar).
>
> Une autre touche qui peut être redéfinie pour un clavier français est la touche '`o`', afin de lui faire générer les symboles o e dans l'o français ('½' et '¼') au lieu des symboles o barrés ('ø' et 'Ø') norvégiens, qui ne sont pas très utiles en français. Notez que les noms de symboles utilisés dans les tables d'encodage correspondent aux symboles de l'encodage ISO 8859-1, et que par conséquent, vous devrez utiliser les noms de symboles « `onehalf` » et « `onequarter` » pour représenter les caractères '½' et '¼'. En effet, ces symboles sont les symboles de l'encodage ISO 8859-1 qui ont le même code que les symboles '½' et '¼' dans l'encodage ISO 8859-15.
>
> Sachez enfin que l'obtention de ces symboles est bien entendue assujettie à l'utilisation d'une police de caractères utilisant l'encodage ISO 8859-15.

Comme vous avez dû le constater, un certain nombre de plans de clavier sont définis en standard dans XFree86. De même, plusieurs géométries et définitions de keycodes sont fournies dans les sous-répertoires `geometry/` et `keycodes/` du répertoire `/usr/X11R6/lib/X11/xkb/`. Le serveur X utilise les fichiers qui sont référencés dans la section « `Keyboard` » du fichier de configuration `XF86Config`. Comme on l'a déjà vu plus haut dans ce chapitre, le mot-clé « `XkbLayout` » permet de définir le plan de clavier à utiliser. Ainsi, si vous spécifiez le plan de clavier « `fr` » à la suite de

ce mot-clé, le fichier de définition des associations keycodes-symboles utilisé sera le fichier `fr` du répertoire `/usr/X11R6/lib/X11/xkb/symbols/`. Les autres options sont fixées par les mots-clés « `XkbRules` » et « `XkbModel` ». Le premier mot-clé indique quel est le fichier à utiliser pour définir les modèles de claviers de XFree86. Ce fichier est localisé dans le sous-répertoire `rules/`, et fournit les valeurs par défaut à utiliser pour chaque modèle de clavier. Le deuxième mot-clé indique le modèle de clavier à utiliser. Ainsi, il n'est nécessaire de spécifier que trois paramètres pour définir le clavier sous XFree86 : le fichier contenant la définition des modèles de clavier, le modèle lui-même et le fichier définissant la disposition des touches.

> **Note:** Vous pouvez également définir tous les paramètres du clavier dans le fichier `XF86Config`. Cette technique est plus compliquée et n'est pas nécessaire pour définir un clavier français, elle ne sera donc pas décrite ici. Vous trouverez plus de renseignements à ce sujet dans la page de manuel `XF86Config`.

10.5. Paramétrage des applications et ressources X

Du fait que toutes les applications X font appel au serveur X pour effectuer leur affichage, et au gestionnaire de fenêtre pour définir l'apparence et la disposition de leurs fenêtres, elles sont soumises aux options générales de ces deux programmes. Cependant, elles peuvent être paramétrées elles-aussi, et leur comportement peut être personnalisé à souhait par chaque utilisateur.

Pour cela, XWindow fournit un mécanisme standard aux applications, afin qu'elles puissent gérer leurs options de configuration de manière uniforme. Ce mécanisme se base sur des paramètres gérés par le serveur X, que l'on appelle des *ressources X*. La plupart des applications utilisent ces ressources pour stocker leur paramètres de configuration, et le fait que ce soit le serveur X qui centralise leur gestion assure une certaine cohérence entre les applications. En général, les ressources X décrivent essentiellement les aspects visuels des applications, comme par exemple les polices de caractères utilisées, les couleurs et l'épaisseur des traits. Cependant, les applications peuvent parfaitement utiliser les ressources pour enregistrer des paramètres qui leurs sont propres. Le serveur X n'interprète en aucune manière les ressources utilisées par les applications, il ne fait que les mettre à disposition des applications.

Le serveur X gère les ressources dans une base de données qui est initialisée lors de son démarrage : la « X Ressource DataBase » (« xrdb » en abrégé). Cette base de données est initialisée lors du démarrage du serveur à partir de fichiers de configuration au format texte, que l'on peut donc éditer et modifier aisément. Le serveur X utilise deux jeux de fichiers lors de son initialisation : les fichiers de configuration par défaut des applications d'une part, et le fichier de préférences personnelles de chaque utilisateur d'autre part. Les fichiers de configuration des applications sont normalement placés dans le répertoire `/usr/X11R6/lib/X11/app-defaults/`. Les applications y copient leurs propres fichiers lors de leur installation. Le nom de chaque fichier correspond au nom de l'application à laquelle il appartient, ce qui permet de retrouver aisément le fichier de configuration d'une appli-

cation donnée. Le fichier de préférences personnelles de chaque utilisateur, quant à lui, se place dans son répertoire racine et porte le nom `.Xresources`.

Les fichiers de ressources des applications sont lus en premier, en général dans les scripts d'initialisation de XWindow. En revanche, le fichier de préférence d'un utilisateur n'est lu que lors de l'ouverture d'une session X par cet utilisateur. Lorsqu'une ressource présente dans le fichier de configuration par défaut d'une application est redéfinie dans le fichier de préférence d'un utilisateur, la valeur utilisée est bien entendue celle de l'utilisateur. Ainsi, chacun peut redéfinir les valeur par défaut des ressources de toutes les applications qu'il utilise. La plupart des applications peuvent également prendre des options en paramètres de ligne de commande, qui permettent de fixer les valeurs de certaines ressources. Ces options prévalent sur les valeurs définies dans les fichiers de configuration des applications et de préférence des utilisateurs.

Vous pourrez vous inspirer du contenu des fichiers de configuration des applications pour paramétrer vos applications. Pour cela, il vous suffira simplement de recopier les définitions des ressources qui vous intéressent dans votre fichier `.Xresources`, et de tester le résultat avec différentes valeurs. L'affectation d'une valeur à une ressource se fait avec la syntaxe suivante :

```
ressource    : valeur
```

où `ressource` est le nom de la ressource, et `valeur` est la valeur à lui affecter.

Les noms de ressources sont structurés de manière hiérarchique. Ils sont en effet constitués d'un certain nombre de composantes, séparées par des points (caractère '`.`'). La première composante qualifie l'application elle-même, et porte son nom. Il est d'usage de mettre la première lettre de ce nom en majuscule, et si cette lettre est un 'X', de mettre également la deuxième lettre du nom en majuscule (beaucoup d'applications X ont en effet un nom commençant par un 'X'). Les composantes suivantes définissent un sous-ensemble de paramètres apparentés dans l'ensemble des paramètres déterminé par les composantes précédentes. Enfin, la dernière composante du nom de ressource constitue le nom du paramètre lui-même.

Par exemple, le nom de ressource suivant :

```
XApplication.mainWindow.background
```

qualifie la couleur d'arrière plan (propriété « `background` ») de la fenêtre principale (propriété « `mainWindow` ») de l'application « `Xapplication` ». Notez que les deux premières lettres du nom de la ressource sont en majuscules ici, car la première lettre du nom de l'application est elle-même un 'X' majuscule.

> **Note:** Les fichiers de ressources utilisent des noms de couleurs prédéfinies pour définir les couleurs des différentes parties des applications. Vous pourrez trouver la liste de ces noms de couleurs ainsi que leur définition dans le fichier `/usr/X11R6/lib/X11/rgb.txt`. Vous pouvez également définir vos propres couleurs avec la syntaxe « `rgb:R/V/B` », où « `R` » représente la portion de rouge de la couleur, « `V` » représente la portion de vert, et « `B` » la portion de bleu.

Ainsi, l'ensemble des paramètres des applications X est organisé un peu comme sont organisés les fichiers dans une arborescence de répertoires. L'analogie ne s'arrête pas là : il est possible de ca-

ractériser un ensemble de ressources grâce à des caractères génériques. Par exemple, en utilisant une étoile (caractère '`*`') comme séparateur à la place du point dans un nom de ressource, toutes les ressources dont le nom comprend les deux composantes seront sélectionnées, que ces deux composantes soient adjacentes ou séparées d'autres composantes intermédiaires. Par exemple, le nom de ressource suivant :

```
XApplication*background
```

qualifie la couleur d'arrière plan de toutes les fenêtres de l'application « `Xapplication` ».

De même, le caractère générique point d'interrogation (caractère '`?`') permet de remplacer une composante par un nom quelconque. Le nom de ressource suivant :

```
XApplication.?.background
```

représente donc toutes les ressources comportant les composantes « `XApplication` » et « `background` », séparées par une composante de nom quelconque.

La structure des noms de ressources d'une application n'est pas due au hasard. Les paramètres sont regroupés soit par fonctionnalité, soit par thème, soit par appartenance à la même partie de l'application. En général, les noms de ressources utilisés par les applications sont simples à comprendre. Vous trouverez des exemples de noms de ressources dans les fichiers de ressources des applications.

Vous pouvez obtenir la liste exhaustive des ressources d'une application avec l'utilitaire **appres**. En fait, cet utilitaire permet d'obtenir la liste des noms des ressources appartenant à une branche de l'arborescence des ressources. Cet utilitaire s'utilise selon la syntaxe suivante :

```
appres branche
```

où `branche` est le nom de la branche que l'on veut explorer.

Comme la base de données des ressources est initialisée au démarrage du serveur X et à l'ouverture de la session X, les modifications que vous pourrez apporter à votre fichier de ressources personnelles ne seront pas prises en compte immédiatement. Pour cela, vous devez demander la relecture de la base de données explicitement et relancer les applications concernées. Cette opération peut être effectuée à l'aide de l'outil **xrdb** (abréviation de l'anglais « X ressource DataBase »). En fait, cet outil permet d'effectuer diverses opérations sur la base de données des ressources.

L'option `-merge` est certainement celle que vous utiliserez le plus. Elle permet de mettre à jour les valeurs des ressources avec celles décrites dans un fichier. La syntaxe utilisée est la suivante :

```
xrdb -merge fichier
```

où `fichier` est le nom du fichier contenant la définition des ressources (généralement, il s'agit de votre fichier `.Xresources`).

L'option `-load` permet d'effectuer le même travail, mais la base de données est vidée au préalable. Cette option permet donc de réinitialiser complètement le contenu de la base de données avec le contenu du fichier passé en paramètre. Sa syntaxe est la même que celle de l'option `-merge`.

L'option `-remove` permet de vider la base de données et de supprimer toutes les ressources que vous auriez pu définir au préalable. Elle s'utilise selon la syntaxe suivante :

```
xrdb -remove
```

Enfin, l'option `-query` permet de lister l'ensemble des ressources que vous avez redéfinies.

10.6. Gestion de la sécurité sous XWindow

XWindow n'effectue pas de contrôle d'accès sur les opérations demandées par un processus connecté à un serveur X. Ceci signifie que tous les processus qui ont accès à un display en ont un accès total. Par exemple, un processus peut parfaitement faire une prise d'écran et récupérer tous les événements provenant du clavier. Il va de soi que ce genre de processus pose un problème de sécurité majeur, aussi XWindow considère-t-il que seuls les processus lancés sur la machine locale au nom de l'utilisateur courant ont le droit de se connecter au serveur X. Ceci protège donc le display de l'utilisateur contre les malversations de personnes mal intentionnées.

Cependant, cette restriction est assez forte, parce qu'elle va vous empêcher de lancer un programme graphique sous un autre nom d'utilisateur. Par exemple, vous ne pourrez pas lancer un programme nécessitant les droits administrateur et dont le bit setuid n'est pas positionné. En effet, une fois que vous vous serez identifié en tant que root avec la commande **su**, les programmes que vous lancerez le seront au nom de root, et le serveur X refusera la connexion. Un autre cas d'école où cette restriction peut être gênante est le lancement d'un programme graphique sur une machine distante. La situation est encore plus grave, car cette fois la demande de connexion au serveur X local ne provient pas de la machine locale, et n'a aucune chance d'aboutir.

XWindow fournit donc les outils nécessaires pour contrôler le niveau de sécurité en cours. Classiquement, le serveur X utilise deux techniques pour contrôler la validité des demandes de connexion des clients. Le premier mécanisme de contrôle d'accès est relativement grossier, puisqu'il se base simplement sur les adresses des machines. Le deuxième mécanisme a été introduit ultérieurement afin de permettre des contrôles plus fins. Les contrôles se font alors à l'aide d'un mécanisme de clés privées, que seuls les clients autorisés connaissent.

10.6.1. La commande xhost

La commande **xhost** permet de fixer le niveau des contrôles d'accès basés sur les adresses de machines. Comme on l'a vu, le serveur X n'accepte les connexions que de la machine locale. Vous pouvez cependant lui indiquer d'accepter toutes les connexions provenant d'une autre machine avec une commande telle que celle-ci :

```
xhost +machine
```

où `machine` est le nom de la machine en qui l'on a confiance. Si aucun nom de machine n'est spécifié, tous les mécanismes de contrôle d'accès sont désactivés.

Inversement, la suppression d'une machine de la liste des machines autorisées est réalisée par la commande suivante :

```
xhost -machine
```

La commande **xhost -** (sans nom de machine) permet de réactiver les mécanismes de contrôle d'accès s'ils ont été désactivés par un **xhost +**.

Notez bien que donner les droits de connexions à une machine suppose que l'on fasse confiance à tous les utilisateurs de cette machine, car alors ils pourront tous se connecter sur votre serveur X local. De plus, rien ne vous garantie que la machine à qui vous donnez ces droits n'a pas été usurpée par celle d'un pirate (technique dite de l'« IP spoofing »). Ce n'est donc pas évidemment la solution recommandée, surtout si vous êtes connecté à Internet.

10.6.2. La commande xauth

Le deuxième mécanisme de sécurité utilise une clé privée. Tous les clients qui désirent se connecter au serveur local doivent connaître cette clé, faute de quoi leur requête sera refusée. Ainsi, vous pouvez très simplement permettre l'utilisation de votre display à une personne donnée sur une machine donnée, simplement en lui communiquant la clé utilisée par le serveur X gérant votre display. Bien entendu, cette communication doit se faire de manière sûre.

Par défaut, les clés privées utilisées par les clients sont enregistrées dans le fichier `.Xauthority` de votre répertoire personnel. Ce fichier contient une clé privée pour chaque display auquel vous avez le droit de vous connecter. Les clients que vous lancez consultent donc ce fichier afin de déterminer la clé à utiliser, en fonction du display auquel ils désirent accéder. Une fois cette clé connue, ils peuvent s'authentifier auprès du serveur X gérant ce display, et ainsi obtenir une connexion. Notez bien que le fichier `.Xauthority` ne doit être accessible que par vous.

Si vous utilisez xdm, une nouvelle clé est automatiquement générée à chaque fois que vous vous connectez à un terminal X. xdm enregistre cette clé dans un fichier temporaire du répertoire référencé par le lien symbolique `/etc/X11/xdm/authdir` (qui référence généralement le répertoire `/var/lib/xdm/authdir/`), que seul root peut accéder. Il communique ensuite ce fichier au serveur X local à l'aide de l'option `-auth` du serveur X pour que celui-ci puisse lire la clé à utiliser. Enfin, xdm enregistre cette clé dans votre fichier `.Xauthority`, pour que vous puissiez lancer des clients dans cette session X.

Comme vous pouvez le constater, ce mécanisme de sécurité est totalement transparent pour l'utilisateur. Cependant, il peut être nécessaire de manipuler le fichier `.Xauthority` pour lire les clés privées et les communiquer aux personnes de confiance qui doivent avoir accès à votre display. Ces manipulations peuvent être effectuées à l'aide de la commande **xauth**.

La clé public associée à un display peut être obtenue avec la commande suivante :

```
xauth list display
```

où `display` est le nom du display auquel la clé donne accès. **xauth** affiche alors le display, le type d'authentification utilisé (`MIT-MAGIC-COOKIE-1`) et la clé privée.

Vous pouvez communiquer cette clé par les moyens que vous voulez à la personne devant accéder à votre display. Celle-ci pourra alors utiliser la commande suivante pour ajouter la clé à son fichier `.Xauthority` :

```
xauth add display . clé
```

où `display` est le display utilisant la clé `clé`. Le caractère '`.`' est une abréviation pour le type d'authentification « `MIT-MAGIC-COOKIE-1` » (type d'authentification par défaut). Dès que la clé aura été ajoutée dans son fichier `.Xauthority`, cette personne aura accès à votre display.

Enfin, la commande suivante :

```
xauth remove display
```

permet de supprimer la clé utilisée pour le display `display`.

10.7. Gestion des polices de caractères

La gestion des polices de caractères est relativement compliquée sous XWindow. En effet, elle est gérée par un protocole complexe, qui permet de décrire avec précision les diverses polices de caractères, quel que soient leur type et leur aspect. De plus, la gestion des polices nécessite de traiter correctement les symboles spécifiques à chaque pays, ce qui complique encore un peu plus les choses. Enfin, pour couronner le tout, XWindow n'utilise pas la notion de graphisme indépendant du périphérique, comme la GDI de Windows. Ceci implique, hélas, qu'il ne se charge que de l'affichage et pas de l'impression. Chaque programme doit donc se charger de lui-même de l'impression, et en général les programmes ne sont capables que d'imprimer sur des imprimantes PostScript. Par conséquent, la configuration des polices de caractères doit non seulement se faire pour XWindow, mais également pour chacun des programmes (ou, au moins, pour l'interpréteur GhostScript, utilisé pour l'impression avec des imprimantes non PostScript).

10.7.1. Gestion des polices de caractères sous XWindow

Originellement, XWindow ne pouvait afficher que des polices de type bitmap, c'est à dire des polices de caractères définies par un dessin pour chaque caractère et pour un certain nombre de résolutions. Cette technique avait l'avantage d'être rapide à l'affichage, mais de relative mauvaise qualité lorsque les tailles demandées n'étaient pas exactement celles pour lesquelles la police avait été dessinée. En effet, la police devait alors être redimensionnée à partir de la taille la plus proche disponible. Ultérieurement, les polices Postscript sont apparues. Ces polices sont définies vectoriellement, c'est à dire par des formules mathématiques. La forme des caractères est ainsi calculée pour chaque dimension, ce qui permet d'avoir une qualité irréprochable. L'avantage des polices Postscript est qu'elles sont gérées de manière native par les imprimantes PostScript, et assurent une qualité d'impression optimale. En revanche, leur affichage sur les écrans n'est pas toujours correct, car les formules utilisées

ont été conçues pour les périphériques disposant de grandes résolutions et ne donnent pas forcément un résultat esthétique pour les résolutions d'écrans. Enfin, les polices Truetype ont été inventées par Apple. Ces polices sont également vectorielles, mais disposent en plus de petites astuces permettant d'améliorer leur lisibilité sur les périphériques à faible résolution tels que les écrans. Microsoft a licencié la technologie Truetype et l'a intégré à Windows par la suite.

Afin de décrire les polices le plus précisément possible, XWindow leur donne des noms relativement complexes. Ces noms constituent ce que l'on appelle la *description logique* des polices (« XLFD », qui est l'abréviation de l'anglais « X Logical Font Description »). Cette convention de dénomination spécifie que la description des polices doit être constituée de différents champs, séparés par des tirets (`'-'`). De plus, la description doit elle-même être précédée d'un tiret. Les différents champs utilisés dans la description logique des polices sont les suivants :

- le nom de l'éditeur de la police, ou le nom du type de la police ;
- le nom de la police (par exemple, « `arial` ») ;
- la graisse de la police (par exemple, « `bold` » pour gras, « `medium` » pour normal) ;
- l'inclinaison de la police (par exemple, `'r'` pour roman, `'i'` pour italique) ;
- la largeur de la police ;
- des options de styles avancées ;
- la taille de la police ;
- la taille des points ;
- la résolution horizontale de la police ;
- la résolution verticale ;
- le type d'espacement de la police (par exemple, `'c'` pour constant, `'p'` pour proportionnel) ;
- la largeur moyenne de la police ;
- le jeu de caractères de la police ;
- la page de code de la police.

Vous pouvez consulter la documentation de XWindow pour une description plus détaillée de ces champs. Ces informations ne sont pas toutes supportées par les polices de caractères. Inversement, certaines polices peuvent correspondre à plusieurs descriptions (ne serait-ce que parce qu'elles disposent de plusieurs tailles).

Parmi les informations décrivant les polices se trouvent le jeu de caractères de la police et sa page de codes (rappelons que ces informations constituent ce que l'on appelle l'*encodage* de la police). Il peut y avoir plusieurs pages de codes pour un jeu de caractères, chacune représentant une manière de numéroter les différents caractères du jeu. Une police peut disposer de plusieurs jeux de caractères, mais en pratique ce n'est que rarement le cas. En revanche, la manière de numéroter les caractères (c'est à dire la page de codes) peut avoir une influence certaine.

Comme on le verra plus tard, le jeu de caractères le plus pratique pour les pays d'Europe de l'Ouest est le jeu ISO 8859 (jeu de caractères dit « latin »). Ce jeu de caractères dispose de la plupart des caractères utilisés en Europe. Pour les alphabets occidentaux, la page de codes la plus utilisée est la page de code 1, ce qui fait que l'encodage des polices occidentales est *ISO 8859-1*. Cependant, quelques caractères ont été oubliés dans cette page de code (notamment le o e dans l'o français ('½'). Ces caractères sont pourtant disponibles dans certaines polices (en particulier, les polices Truetype provenant de Windows), mais ne sont malheureusement pas disponibles avec l'encodage ISO 8859-1. Pour y accéder, on est obligé d'utiliser un autre encodage, comme par exemple l'encodage ISO 8859-15. Cet encodage est quasiment identique à l'encodage ISO 8859-1, aux caractères additionnels près, qui ont été ajoutés pour les pays européens. Il est également possible d'utiliser la page de code 1252 des polices de caractères de Windows. Cette page de code correspond à l'encodage « windows-1252 », parfois également nommé « microsoft-ansi » ou encore « microsoft-cp1252 ». En résumé, le jeu de caractères et la page de code permettent d'indiquer pour quel pays (ou quel alphabet) une police est destinée.

> **Note:** Le problème des encodages est que seul l'encodage ISO 8859-1 est vraiment utilisé par la majorité des gens. Ceci implique que les autres encodages risques de ne pas être reconnus par tous les programmes. En particulier, il est assez difficile d'imprimer des textes encodés avec des encodages non standard.

La description logique des polices de caractères est très précise, puisqu'elle permet de spécifier l'origine de la police, son nom, les informations la concernant, les caractères qu'elle comprend et comment ils sont numérotés. Lorsque l'on choisit une police de caractères, on peut parfaitement ne préciser que certains critères (comme par exemple, l'encodage de la police). Dans ce cas, les champs constituant le nom de la police non spécifiés pourront être remplacés par des caractères génériques. Le caractère '`*`' permet ainsi de spécifier n'importe quelle valeur pour ce champs, et le caractère '`?`' permet de spécifier n'importe quel caractère dans un champ.

Le programme xfontsel permet de sélectionner les polices de caractères installées sur un système. Il peut être utile pour comprendre la signification des différents champs de la description logique des polices. Des exemples de descriptions logiques de polices sont donnés ci-dessous :

```
-adobe-helvetica-medium-r-*-*-12-*-*-*-*-*-iso8859-1
-*-courier-bold-*-normal-*-10-*-*-*-*-*-iso8859-15
-winfonts-arial-*-i-normal-*-16-*-*-*-*-*-microsoft-cp1252
```

Les polices de caractères sont souvent regroupées dans des répertoires. Il peut exister plusieurs répertoires de polices sur un système, si bien qu'il est nécessaire d'indiquer à XFree86 dans quels répertoires ils doivent rechercher les polices de caractères utilisables. Les répertoires des polices utilisées par le serveur X sont indiqués dans le fichier `XF86Config`, dans la section « `Files` ». Comme on le verra plus tard, cette section peut également contenir des références sur des serveurs de polices de caractères. Les serveurs de polices de caractères peuvent rechercher les polices dans des répertoires indiqués de différentes manières selon le serveur. Pour certains serveurs, les répertoires de polices sont indiqués dans un fichier de configuration, pour d'autres, ils sont indiqués en ligne de commande.

Dans le protocole X, les répertoires de polices doivent contenir un fichier `fonts.dir` donnant la liste des polices de ce répertoire. Ce fichier indique, sur sa première ligne, le nombre de polices installées dans ce répertoire, et, sur les lignes suivantes, l'association entre les fichiers de polices et leurs description logique. Ce fichier est créé normalement par le programme **mkfontdir**. Ce programme génère le fichier `fonts.dir` à partir des informations contenues dans les fichiers des polices ou à partir du nom des fichiers des polices eux-mêmes. En général, les polices à taille variable ne contiennent pas ces informations dans le format standard des polices X11 classiques, et **mkfontdir** ne peut donc pas créer le fichier `fonts.dir` automatiquement. Pour ces polices, il faut créer un fichier `fonts.scale` contenant les mêmes informations que le fichier `fonts.dir`, et que **mkfontdir** utilisera pour créer ce dernier. La méthode pour créer le fichier `fonts.scale` dépend du type de polices utilisé. Celle utilisée pour les polices Truetype sera décrite dans la Section 10.7.2.

En général, les encodages les plus standards sont gérés directement par le serveur X ou par le serveur de polices. En particulier, l'encodage ISO 8859-1 est géré nativement. Il est toutefois possible de définir de nouveaux encodages dans des fichiers d'encodages. Pour que le serveur X ou le serveur de polices puisse utiliser les polices définies avec ces encodages, il faut que le répertoire d'installation des polices contiennent un fichier `encodings.dir`. Ce fichier a la même structure que le fichier `fonts.dir`, c'est à dire qu'il contient le nombre des encodages sur la première ligne, et, sur chaque ligne suivante, le nom de l'encodage et le nom d'un fichier contenant la définition d'un encodage, séparés par un espace. De cette manière, le serveur X et le serveur de polices sont capables de réaliser l'association entre le nom de l'encodage utilisé dans la description logique de polices et le fichier de définition de cet encodage. La méthode permettant de créer le fichier `encodings.dir` sera décrite plus loin dans la section traitant de l'installation des polices Truetype.

L'écriture des fichiers de définition d'encodages de polices est une tâche assez compliquée, qui nécessite de bien connaître la numérotation des caractères de chaque type de fichiers de polices. De manière très simplifiée, on peut dire que les fichiers de définition d'encodage font l'association entre le numéro de chaque caractère dans la police et une numérotation standardisée des caractères (XWindow utilise souvent la numérotation Unicode) de cette police. Ceci permet de manipuler les textes avec la numérotation standard, et d'utiliser des polices qui ne définissent pas tous les caractères de cette numérotation ou qui ne les numérotent pas de la même manière. Heureusement, les encodages les plus standards ont déjà été écrits, il est fort peu probable que vous ayez à vous intéresser à ce problème. La structure des fichiers d'encodages ne sera donc pas décrits plus en détail dans ce document.

10.7.2. Installation des polices Truetype

L'installation des polices TrueType sous X Window ne pose désormais plus de problème particulier, puisqu'il suffit de définir le fichier `fonts.dir` dans le répertoire d'installation de ces polices. Cependant, un certain nombre d'opérations supplémentaires devront être réalisées pour permettre l'impression des documents utilisant les polices TrueType. Ce paragraphe détaille la manière de déclarer les polices TrueType au niveau du serveur X, et présente les opérations nécessaires à leur impression pour quelques logiciels courants.

Généralement, les fichiers de polices sont placées dans des sous-répertoires du répertoire `/usr/X11R6/lib/X11/font`
Ces sous-répertoires permettent de classer les fichiers de polices par type et par taille. Dans la suite

de ce document, il sera supposé que les polices TrueType sont toutes situées dans le sous-répertoire `truetype/`.

10.7.2.1. Configuration du serveur X

Pour permettre au serveur X d'utiliser les polices TrueType, il faut simplement écrire le fichier `fonts.dir` de chaque répertoire de polices. Rappelons que ce fichier contient la liste des polices du répertoire dans lequel il se trouve, et est utilisé à la fois par le serveur X et par les serveurs de polices de caractères. La notion de serveur de polices sera vue en détail dans la Section 10.7.3.

Normalement, le fichier `fonts.dir` est généré par le programme **mkfontdir**. Ce programme utilise les informations contenues dans les fichiers de polices ou dans le nom de ces fichiers de polices pour le générer. Malheureusement, les polices Truetype, comme la plupart des polices à taille variable, ne contiennent pas ces informations dans un format compréhensible par mkfontdir, ni dans leurs fichiers, ni dans leurs noms. Celui-ci ne peut donc pas créer le fichier `fonts.dir` directement. Dans ce cas, mkfontdir utilise le fichier de définition des polices à taille variable `fonts.scale`. Ce fichier doit être créé manuellement, ou à l'aide de l'utilitaire **ttmkfdir** de Joerg Pomnitz, que vous pouvez trouver sur Internet (http ://www.joerg-pommnitz.de/TrueType/ttmkfdir.tar.gz). Cette archive contient un fichier binaire déjà compilé, nommé `ttmkfdir.linuxbin.glibc2`. Vous pouvez utiliser directement ce fichier si vous le voulez, en le renommant en `ttmkfdir` et en le copiant dans le répertoire des binaires de X Window.

L'utilitaire ttmkfdir s'utilise de la manière suivante :

```
ttmkfdir > fonts.scale
```

pour générer le fichier `fonts.scale`.

Une fois le fichier `fonts.scale` créé, il ne reste plus qu'à appeler **mkfontdir** avec la ligne de commande suivante :

```
mkfontdir
```

Cette commande aura pour effet de créer le fichier `fonts.dir` à partir du fichier `fonts.scale` (en fait, il s'agit d'une simple recopie).

Les encodages utilisés par le serveur X pour les polices Truetype sont tous définis dans le fichier `fonts.dir`, qui donne la correspondance entre les descriptions logiques de polices et les fichiers de polices Truetype. Par conséquent, vous pourrez aisément modifier ou ajouter des encodages différents pour ces polices de la manière suivante :

- éditez le fichier `fonts.scale` contenant la liste des noms de polices et des descriptions logiques de polices ;
- modifiez la description logique de chaque police ;
- relancez mkfontdir.

La modification de la description logique de la police consiste simplement à changer l'encodage utilisé par la police en un autre encodage. Vous pouvez utiliser n'importe lequel des encodages suivants, qui sont gérés nativement par le serveur de polices et par le serveur X :

- `iso10646-1` (encodage Unicode) ;
- `iso8859-1` à `iso8859-10` ;
- `iso8859-15` ;
- `koi8-r`, `koi8-u`, `koi8-ru`, `koi8-uni`, ou `koi8-e` ;
- `microsoft-symbol` (uniquement pour les polices Truetype) ;
- `apple-roman` (uniquement pour les polices Truetype d'Apple).

Vous pouvez également utiliser des encodages additionnels, à condition d'utiliser l'option `-e` de **mkfontdir** lors de la génération du fichier `fonts.dir`. Cette option permet d'indiquer le répertoire contenant les fichiers de définition des encodages à utiliser :

```
mkfontdir -e répertoire
```

où `répertoire` est le répertoire contenant les fichiers de définition d'encodages. Cette commande a pour conséquence de créer un fichier `encodings.dir` en plus du fichier `fonts.dir`.

X Window est fourni avec un certain nombre de fichiers de définition d'encodages standard, tous situés dans le répertoire `/usr/X11R6/lib/X11/fonts/encodings/`. L'un des plus intéressants est sans doute `microsoft-cp1252`, qui permet de disposer des caractères additionnels ajoutés par Microsoft dans ses polices, tout en conservant un encodage compatible avec les documents Windows.

> **Note:** Si vous voulez rester compatible avec les normes ISO, vous pouvez utiliser l'encodage ISO 8859-15 au lieu de `microsoft-cp1252`. Vous accéderez ainsi à tous les caractères utilisés en Europe, mais vous ne pourrez plus utiliser les caractères additionnels définis par Microsoft.
>
> Il faut noter que la plupart des polices sont défectueuses et indiquent une page de code erronée dans leurs fichiers. Ceci explique pourquoi quelques descriptions logiques de polices dans le fichier `fonts.scale` généré automatiquement par **ttmkfdir** peuvent être fausses. Pour ces polices, il faudra corriger l'encodage manuellement en suivant la méthode décrite ci-dessus.
>
> Certains programmes ne donnent pas la possibilité de choisir l'encodage pour les polices de caractères. C'est en particulier le cas pour la suite bureautique StarOffice. Il y a qu'une seule solution à ce problème : ne définir qu'un seul encodage pour la police en question.

10.7.2.2. Configuration des polices Truetype pour l'impression

Comme il l'a été dit plus haut, la configuration des polices Truetype pour l'impression dépend fortement de chaque programme utilisé. D'une manière générale, plusieurs cas peuvent se présenter :

- soit on dispose d'une imprimante PostScript, auquel cas il faut convertir les polices Truetype en polices Adobe Type 42, qui constituent une encapsulation des polices Truetype en PostScript. Cette conversion a l'avantage de ne pas provoquer de perte de qualité, puisque la police Truetype est utilisée telle quelle ;
- soit on ne dispose pas d'imprimante PostScript, auquel cas on utilise nécessairement un interpréteur PostScript. Cet interpréteur est souvent GhostScript, car il s'agit encore une fois d'un logiciel libre. Quel que soit l'interpréteur PostScript utilisé, il faut le configurer pour utiliser les polices Truetype. Nous verrons comment le faire pour les versions de GhostScript ultérieures à la 6.01.

Dans les deux cas, il se peut qu'il faille configurer le logiciel utilisé pour qu'il puisse utiliser les polices Truetype. C'est en particulier le cas de la suite bureautique StarOffice, qui ne permet de choisir que les polices qu'il peut imprimer.

Les paragraphes suivants décrivent les procédures à suivre pour imprimer les polices Truetype en général. La configuration de StarOffice 5.2 sera également décrite, car c'est l'une des suites logicielles les plus abouties sous Linux (qui de plus est devenue un logiciel libre grâce à Sun). La configuration des autres logiciels ne sera pas abordée, consultez leur aide ou recherchez des renseignements sur Internet quant à la procédure à suivre.

10.7.2.2.1. Conversion des polices Truetype en polices Adobe de Type 42

La conversion des polices Truetype en polices Adobe de Type 42 est une opération nécessaire si l'on utilise directement une imprimante PostScript, ou que l'on utilise un interpréteur PostScript incapable de gérer directement les polices Truetype. De manière général, il est recommandé d'utiliser GhostScript même si l'on possède une imprimante PostScript, car la configuration de l'impression se fait de la même manière que les autres utilisateurs de Linux, et il est donc plus facile de trouver de l'aide sur Internet.

La conversion en polices Adobe de Type 42 peut être réalisée avec le programme **ttfps**, disponible sur Internet sous le nom `ttfps.tar.gz`. Cet outil est tout à fait correct, mais il souffre d'un grave défaut : il ne permet pas de choisir l'encodage de la police PostScript qu'il génère. Plus grave encode, il utilise systématiquement l'encodage standard Adobe, qui ne dispose pas de la plupart des lettres accentuées françaises (c'est aussi la raison pour laquelle il est recommandé d'utiliser GhostScript). Vous pouvez bien entendu modifier le code source si vous vous en sentez le courage.

Vous devrez compiler **ttfps** pour pouvoir l'utiliser. Lorsque vous aurez extraits les fichiers sources de l'archive, vous aurez à éditer le fichier `Makefile` pour :

- définir la variable d'environnement CC avec pour valeur le nom du compilateur que vous utilisez (en l'occurence, GCC) :

  ```
  CC=gcc
  ```

- choisir entre les deux jeux d'options de compilations CFLAGS. Vous ne devez en choisir qu'un seul, il faut obligatoirement commenter l'autre avec un caractère dièse ('#'). Le choix dépend de

l'architecture de votre machine. Si vous utilisez un PC, vous devrez choisir l'option contenant l'option `-DSMALLENDIAN`.

Une fois ces modifications faites, vous pourrez compiler **ttsps** avec la simple commande suivante :

```
make
```

L'utilisation de **ttfps** est très simple. Pour convertir une police Truetype en police Adobe de Type 42, il suffit d'utiliser la ligne de commande suivante :

```
ttfps [-a fichier.afm] fichier.ttf fichier.pfa
```

où `fichier.afm` est le nom du fichier de définition des dimensions de la police PostScript, `fichier.ttf` est le nom du fichier de police Truetype à encapsuler, et `fichier.pfa` est le nom de la police PostScript résultante.

La génération du fichier .afm est facultative. Elle doit être utilisée lorsque l'on veut utiliser un logiciel comme StarOffice, qui a besoin de connaître les dimensions de la police pour l'afficher correctement et pour faire ses calculs de mise en page.

10.7.2.2.2. Installation des polices Truetype pour GhostScript

GhostScript est un interpréteur PostScript capable d'imprimer les polices Truetype. Sa configuration est très simple, puisqu'il suffit de lui donner le nom de police que les fichiers PostScript utilisent pour la référencer et le nom du fichier de police correspondant. Cette association est définie dans le fichier `Fontmap` de GhostScript, que vous pourrez trouver dans son répertoire d'installation (normalement, GhostScript se trouve dans le répertoire `/usr/share/ghostscript/version/`, où `version` est le numéro de la version installée).

Les informations que vous devez ajouter dans ce fichier sont similaires à celles stockées dans le fichier `fonts.dir` du répertoire de polices. Pour chaque police, il faut rajouter une ligne de la forme suivante :

```
(nom) (fichier);
```

où `nom` est le nom de la police d'imprimante que les programmes utilisent dans les fichiers PostScript qu'ils génèrent, et `fichier` est le chemin absolu du fichier de la police Truetype.

En pratique, le nom de la police d'imprimante peut être choisi librement. En revanche, il faudra bien utiliser ce nom lors de la déclaration des polices d'imprimantes dans les logiciels capables d'imprimer en PostScript. On veillera toutefois à ne pas utiliser des noms de polices contenant des espaces blancs, car certains programmes peuvent avoir du mal à manipuler de tels noms.

10.7.2.2.3. Configuration de StarOffice pour l'utilisation des polices Truetype

StarOffice ne gère pas les polices Truetype directement. En fait, il n'est capable d'imprimer lui-même que les polices Adobe de Type 1. De plus, il ne propose pour l'affichage que les polices qu'il

est capable d'imprimer. Cependant, il est possible d'utiliser malgré tout les autres polices en tapant directement leur nom dans le contrôle de sélection de polices, tel qu'il apparaît dans leur description logique et en minuscule. Dans ce cas, StarOffice utilisera les polices XWindow malgré tout, mais il ne pourra toujours pas les imprimer. Il faut également s'attendre à ce que la mise en page ne soit pas correcte, car il ne peut pas déterminer les caractéristiques géométriques de la police.

Pour qu'il puisse imprimer les caractères d'une police Truetype, il existe deux possibilités. La première est tout simplement de convertir les polices Truetype en polices Adobe de Type 1. Cette méthode souffre de plusieurs inconvénients :

- les polices Adobe de Type 1 sont moins précises que les polices Truetype, surtout pour les petits caractères ;
- la conversion n'est pas parfaite (c'est à dire que les caractères ne sont pas exactement les mêmes) ;
- elle n'est pas complète (les outils de conversion ne gèrent que l'encodage ISO8859-1).

La deuxième solution est de faire croire à StarOffice que l'imprimante contient la définition de la police Truetype dans sa mémoire. Cette solution a l'avantage de ne pas convertir les polices Truetype en police Adobe de Type 1, et donc de ne pas perdre en qualité. De plus, elle permet d'économiser la place disque occupée par les polices converties. Cependant, elle a également des inconvénients :

- il faut passer par un émulateur PostScript (comme GhostScript par exemple) capable de comprendre les polices Truetype, ou il faut encapsuler ces polices dans une police Adobe de Type 42 (cette encapsulation n'est pas une conversion, il n'y a donc toujours pas de perte de qualité) ;
- la configuration de StarOffice est bien plus compliquée ;
- il faut installer les polices pour l'interpréteur PostScript ou pour l'imprimante ;
- il faut générer un fichier de description de caractéristiques géométriques .afm. Ceci n'est pas très aisé, et le problème de la gestion des encodages autres que ISO 8859-1 se pose à nouveau.

> **Note:** Aucune de ces solutions ne permet d'utiliser les encodages ISO 8859-15 et microsoft-cp1252, ce qui implique que certains caractères des polices Truetype ne seront pas imprimables. En particulier, le caractère o e dans l'o français ('½') ne pourra pas être imprimé...
>
> En fait, cette limitation est une limitation des outils de conversion pour la première méthode, et une limitation de GhostScript pour la deuxième.

Seule la deuxième méthode sera décrite ici, car c'est la seule méthode qui permette réellement d'utiliser les polices Truetype avec StarOffice. De plus, c'est la méthode qui nécessite le plus d'explications, ceux qui sont intéressés par la première méthode trouveront des informations claires dans l'aide de StarOffice.

La première étape est de créer un fichier .afm pour chaque fichier de police Truetype. Ces fichiers contiennent la définition des dimensions de chaque caractère de la police, ainsi que l'encodage utilisé. StarOffice utilise ces fichiers pour effectuer ses calculs de mise en page, à partir des dimensions des caractères.

> **Note:** StarOffice n'utilise pas l'encodage indiqué dans le fichier .afm. Il utilise l'encodage de la police X11 de l'affichage. Si une police dispose de plusieurs encodages différents, il choisit celui qui lui semble le plus approprié. En général, il s'agit de l'encodage ISO8859-1. Comme on l'a vu plus haut, on est obligé de ne définir qu'un seul encodage dans le fichier `fonts.dir` si l'on veut être sûr que StarOffice utilise l'encodage que l'on désire.

La création des fichiers .afm est une opération assez compliquée. Elle peut être faite grâce au programme PostScript « afmmaker.ps », que l'on peut trouver sur Internet. Pour l'utiliser, il faut l'éditer et définir la police pour laquelle on désire créer un fichier .afm. Les champs qui doivent être modifiés sont marqués avec le commentaire « `MAKE CHANGES` ». On veillera à ne pas modifier le programme en dehors de ce cette zone.

Les informations à modifier sont les suivantes :

- le nom de la police d'imprimante (variable « `/fontName` »). Le nom qui doit être utilisé ici est le nom de la police qui a été entré dans le fichier `Fontmap` de GhostScript ;
    - l'encodage de la police (variable « `/encodingScheme` »). Il est recommandé de conserver l'encodage « `ISOLatin1Encoding` », car les autres encodages sont moins riches. En fait, on peut utiliser tous les encodages reconnus par GhostScript, à savoir « `StandardEncoding` », « `ISOLatin1Encoding` », « `SymbolEncoding` », ou « `DingbatsEncoding` » ;

> **Note :** Le fait de ne pas pouvoir utiliser l'encodage microsoft-cp1252 ou l'encodage ISO 8859-15 est bien une limitation de GhostScript.

- le nom complet de la police (variable « `/fullName` ») ;
- le nom de la famille de la police (variable « `/familyName` ») ;
- la graisse de la police (variable « `/weight` ») ;
- l'angle de la police si elle est italique (choisissez 0 s'il s'agit d'une police romane).

Pour la plupart des polices, il n'est nécessaire de préciser, en plus du nom de la police d'imprimante, que la graisse de la police. Les polices normales utilisent souvent « `Medium` », et les polices grasses utilisent souvent « `Bold` ».

Une fois ces modifications faites, vous pourrez créer le fichier .afm avec la commande suivante :

```
gs -q -dNODISPLAY -dBATCH afmmaker.ps > fichier.afm
```

où `fichier.afm` est le nom du fichier .afm à créer. Vous pouvez choisir n'importe quel nom, StarOffice ne s'en soucie absolument pas (il utilise tous les fichiers .afm dont il dispose de toutes façons).

La deuxième étape est d'installer tous les fichiers .afm ainsi créé dans le sous-répertoire `xp3/fontmetrics/afm/` du répertoire d'installation de StarOffice. Assurez-vous de ne pas écraser de fichiers .afm déjà présent, et renommez au besoin les fichiers .afm que vous avez créé.

La troisième étape est de faire l'association entre la police d'imprimante et la police X11 à utiliser. Cette association est faite dans le fichier `xp3/psstd.fonts`, qui contient la définition des polices

PostScript standard. La structure de ce fichier est très simple : pour chaque police installée, il existe une ligne donnant le nom de la police d'imprimante (celui qui a été utilisé dans le fichier `Fontmap` de GhostScript) et la description logique paramétrée de la police X11 correspondante, séparés par une virgule. La description logique paramétrée de la police X11 est la même description que celle que l'on peut trouver dans le fichier `fonts.dir` du répertoire d'installation des polices, à ceci près que les champs de taille de police, de taille des points, de résolution horizontale et de résolution verticale sont remplacés par la chaîne de caractère « `%d` ». Cette chaîne de caractères indique à StarOffice que les champs correspondants devront être remplacés par leur valeur en fonction de la taille de la police choisie. Faites bien attention à ne pas faire de fautes de frappe lors de la saisie de ces descriptions. Le plus simple est encore de les copier à partir du fichier `fonts.dir` et de les modifier pour remplacer les spécifications de taille et de résolution par « `%d` ». La ligne suivante donne un exemple d'association entre la police d'imprimante TimesNewRoman et la police Truetype correspondante dans X11 :

```
TimesNewRoman, -monotype-times new roman-medium-r-normal--%d-%d-%d-%d-p-0\
-iso8859-1
```

> **Note:** StarOffice n'utilise pas nécessairement l'encodage de la police que vous indiquez ici. Il est recommandé d'utiliser l'encodage ISO8859-1 pour ne pas avoir de problèmes, cependant, même si l'encodage est différent, tout fonctionnera correctement.

La dernière étape est d'indiquer à StarOffice que cette police est connue de l'imprimante. Ceci est nécessaire pour qu'il n'essaie pas de la définir (il n'y parviendrait d'ailleurs pas, puisqu'il ne sait pas lire les polices Truetype), et qu'il laisse cette imprimante utiliser sa propre définition. Pour les utilisateurs de GhostScript, il faut modifier le fichier `SGENPRT.PS` du sous répertoire `ppds` (« PostScript Printer Definition ») du répertoire d'installation de StarOffice. Vous devrez ajouter une ligne pour chaque police d'imprimante installée, dans la section « `Fonts` » de ce fichier. Le format de ces lignes doit être le suivant :

```
*Font nom : Standard "(001.002)" Standard ROM
```

où `nom` est le nom de la police d'imprimante tel qu'il apparaît, encore une fois, dans le fichier `Fontmap` de GhostScript.

Une fois ces étapes réalisées, vous pourrez redémarrer StarOffice, et vous disposerez de vos polices Truetype sous StarOffice. Le nom de ces polices sera celui de la police X11 utilisée pour l'affichage. Le nom de la police utilisée dans les fichiers d'impression sera celui du fichier `Fontmap` de GhostScript.

> **Note:** Il est nécessaire d'utiliser le même encodage pour la police X11 et dans le fichier .afm. En effet, si ce n'est pas le cas, le texte imprimé utilisera l'encodage de la police X11 et sera imprimé avec l'encodage du fichier .afm. Ceci peut provoquer quelques erreurs désagréables à l'impression. Pour éviter tous les problèmes, il est recommandé de n'utiliser que l'encodage ISO 8859-1, jusqu'à ce que GhostScript puisse utiliser l'encodage microsoft-cp1252 ou l'encodage ISO 8859-15.

10.7.3. Configuration d'un serveur de polices

X Window étant un système de fenêtrage fonctionnant en réseau, il propose des services additionnels sur le réseau en plus de l'affichage. Parmi ces services, on notera la possibilité de mettre en place des serveurs de polices de caractères. Ces serveurs permetttent à des clients situés sur d'autres machines d'accéder à la définition des polices de caractères de la machine locale.

Le serveur de police fourni par X Window se nomme « `xfs` » (abréviation de l'anglais « X Font Server »). Ce serveur utilise un fichier de configuration, qui permet de lui spécifier les options avec lesquelles il doit démarrer. Dans la suite de ce document, le nom supposé de ce fichier est `/etc/xfs.conf`. Un fichier de configuration typique est donné ci-dessous :

```
# Exemple de fichier de configuration du serveur de police :

# Limite à 10 le nombre de clients connectés à ce serveur :
client-limit = 10

# Demande le démarrage d'un autre serveur quand la limite
# précédente est atteinte :
clone-self = on

# Répertoire des polices de caractères :
catalogue = /usr/X11R6/lib/X11/fonts/truetype

# Fixe la taille par défaut (en dixièmes de points) :
default-point-size = 120

# Résolution par défaut (100 x 100 et 75 x 75) :
default-resolutions = 100,100,75,75

# N'utilise pas le mécanisme de traçage des erreurs du système :
use-syslog = off

# Utilise le fichier d'erreurs suivant à la place :
error-file = /root/xfs.errors
```

Lorsque vous aurez créé votre fichier de configuration, il ne vous restera plus qu'à tester si tout fonctionne bien. Il vous faut pour cela lancer le serveur de polices avec la ligne de commande suivante :

```
xfs -port 7100 -config /etc/xfs.conf &
```

et essayer de lui demander la liste des polices qu'il peut fournir :

```
fslsfonts -server localhost :7100
```

Si cette dernière commande échoue, il se peut que le chemin indiqué pour les répertoires de polices dans le fichier `/etc/xfs.conf` ne soit pas correct, ou que le fichier `fonts.dir` n'existe pas ou n'est pas correct dans chaque répertoire de polices.

Si vous le désirez, vous pouvez faire en sorte que le serveur de polices soit démarré automatiquement au lancement de X. Vous pourrez pour cela créer un script de lancement du serveur de polices, que vous placerez dans le répertoire `/etc/rc.d/` (ou le répertoire `/sbin/init.d/`, selon votre distribution) pour lancer et arrêter le serveur de police en suivant le mécanisme des niveaux d'exécution. Vous trouverez ci-dessous un exemple de script de lancement du serveur de police permettant d'utiliser le fichier de configuration précédent :

```
#!/bin/bash
#
# /etc/rc.d/xfs
#
# Fichier de lancement du serveur de polices.
#

PORT=7100
FILE=/etc/xfs.conf
PRGM=/usr/X11/bin/xfs

[ -f $PRGM ] || exit 0

# Analyse les paramètres :
case "$1" in
    start)
        # Vérifie si xfs est déjà lancé :
        if [ -f /var/lock/subsys/xfs ]; then
            echo -n "Redémarrage du serveur de polices"
            killall -e -TERM $PRGM
            rm -f /var/lock/subsys/xfs
        fi
        $PRGM -port $PORT -config $FILE &
        touch /var/lock/subsys/xfs
       ;;
    stop)
        if [! -f /var/lock/subsys/xfs ]; then
            echo -n "xfs n'est pas lancé"
        else
            echo -n "Arrêt du serveur de polices"
            killall -e -TERM $PRGM
            rm -f /var/lock/subsys/xfs
        fi
       ;;
    restart)
        echo -n "Relecture du fichier de configuration"
        if [ -f /var/lock/subsys/xfs ]; then
            killall -e -USR1 $PRGM
        fi
       ;;
    *)
```

```
        echo "Usage : $0 {start|stop|restart}"
        exit 1
esac
exit 0
```

Comme X11 est démarré classiquement dans le niveau d'exécution 3 ou 4, vous devrez créer les liens symboliques vers ce fichier pour le lancement et l'arrêt du serveur de police dans le répertoire `rc3.d/` ou `rc4.d/`. Attention, rappelez-vous que les noms de ces liens indiqueront le moment où le script sera appelé, aussi bien pour l'entrée que pour la sortie du niveau d'exécution. Vous devrez impérativement faire en sorte que ce script soit appelé avant que XWindow ne démarre si vous désirez utiliser le serveur de police dans les sessions X locales.

L'utilisation du serveur de polices est très simple. Il suffit d'y accéder comme à un répertoire de polices de caractères normal, en utilisant la syntaxe suivante :

```
tcp/serveur :port
```

où `serveur` est le serveur sur lequel le serveur de police à accéder est lancé, et `port` est le port que ce serveur écoute. Par exemple, pour ajouter les polices d'un serveur de polices locales à la liste des polices du serveur X, il suffit de taper la commande suivante :

```
xset +fp tcp/127.0.0.1 :7100
```

Vous pourrez alors vérifier que ces polices sont bien disponibles avec le programme xfontsel.

Si vous le désirez, vous pouvez ajouter automatiquement les polices d'un serveur de polices en ajoutant la ligne suivante dans la section « `Files` » dans le fichier de configuration `XF86Config` du serveur :

```
FontPath "tcp/127.0.0.1 :7100"
```

Cette ligne permet d'indiquer au serveur X qu'il trouvera des polices de caractères au niveau du serveur de polices de la machine locale.

10.8. Problèmes classiques rencontrés

Il est possible que vous rencontriez quelques problèmes au lancement de XWindow après avoir modifié sa configuration. Ces problèmes peuvent passer du blocage complet de la machine à l'impossibilité de lancer XWindow.

D'une manière générale, il est recommandé de ne pas modifier la configuration dans le niveau d'exécution de XWindow (3, 4 ou 5 selon les distributions). En effet, dans ces niveaux, xdm relance automatiquement le serveur X dès qu'il détecte sa terminaison. Si le serveur X ne peut pas se lancer, votre ordinateur se bloquera dans une boucle infinie, avec en prime changement de mode graphique à chaque itération (ce qui peut endommager sérieusement votre moniteur). Si cela vous arrivait, vous

n'auriez plus qu'à tenter le redémarrage de la machine en basculant rapidement sur un terminal (avec `CTRL+ALT+F1` suivi de `CTRL+ALT+DEL`. Faites-le très rapidement, avant que xdm n'ait le temps de tenter un redémarrage du serveur X !), et à redémarrer en indiquant le niveau d'exécution 2 à l'amorçage du noyau. Par conséquent, essayez vos changement de configuration dans le niveau d'exécution 2 si possible, et faites vos tests avec la simple commande**startx**.

Des erreurs un peu plus techniques peuvent provenir de la configuration réseau elle-même (n'oubliez pas que XWindow est un système graphique basé réseau). Les erreurs les plus effroyables sont sans doute les erreurs du type « `_X11TransSocketINETConnect : Can't connect : errno = n` », où `n` est un code d'erreur numérique. Typiquement, cette erreur signale un problème dans la configuration réseau. Les deux erreurs les plus classiques sont l'erreur 101, qui signale que la machine indiquée dans le display n'a pas pu être contactée, et l'erreur 111, qui signale que cette machine est bien accessible par le réseau, mais que le serveur X ne s'y trouve pas.

En général, vous devrez vérifier toute votre configuration réseau si vous rencontrez des erreurs 101, et le problème n'est certainement pas spécifique à XWindow. Vous pouvez également vérifier la validité de votre display, car il se peut qu'il désigne une machine inaccessible sur votre réseau (ou que la résolution de nom ait échoué pour cette machine).

Pour ce qui est de l'erreur 111, c'est beaucoup plus simple. Dans la grande majorité des cas, le serveur X désigné n'existe tout simplement pas. Il faut donc tout simplement s'assurer que le serveur X en charge de gérer le display indiqué dans la variable d'environnement DISPLAY est bien lancé sur la machine désignée.

Enfin, lorsqu'une application cliente refuse de démarrer en affichant un message d'erreur « `Can't open display :` », c'est tout simplement que la variable d'environnement DISPLAY n'a pas été précisée, et que l'option en ligne de commande `-display` n'a pas été utilisée non plus. L'application ne sait donc tout simplement pas à quel display se connecter. Si le display a bien été défini, mais que le message « `Can't open display :xxx` » est complété du message « `Connection to "xxx" refused by server` » ou « `Client is not authorized to connect to Server` », c'est tout simplement que le client a été lancé dans un autre compte utilisateur que le compte que vous utilisez couramment, et que ce compte ne dispose pas des droits nécessaires pour accéder à ce display. Vous devez dans ce cas donner l'accès à votre display au compte utilisateur sous lequel vous lancez le client.

Chapitre 11. Conclusion

Vous avez pu voir dans ce document les différentes procédures à mettre en ½uvre pour installer Linux. Ces procédures peuvent paraître compliquées, et en fait elles le sont effectivement. Cependant, il faut faire un choix entre fonctionnalité et simplicité. Linux choisit la voie la plus difficile : celle sur laquelle il faut être performant et fournir le plus de possibilités. La complexité qui en découle se voit immédiatement lors de son installation, mais elle se comprend car chaque étape permet de le rendre à chaque fois plus puissant, en lui ajoutant une fonctionnalité de plus. Finalement, ce qui vous a motivé pendant toutes ces étapes, c'est sans aucun doute le désir de bénéficier de la stabilité et de la puissance de Linux. Vous ne serez pas déçu, et vous rendrez vite compte que Linux dépasse de loin les systèmes soi-disant plus ergonomiques sur ces deux points, dans une telle proportion que vous finirez par ne plus pouvoir les utiliser. Et si vous y êtes contraints, vous ne louerez plus leur facilité d'emploi, mais vous pestifèrerez bel et bien contre leur comportement aléatoire, leurs incohérences ou leurs plantages à répétition. En réalité, comme vous allez bientôt le découvrir, Linux est simple à utiliser pour les travaux du quotidien. De plus, comme il l'a déjà été dit au début de ce document, ce qui paraît compliqué au premier abord est souvent tout simplement inhabituel. C'est à l'usage que vous vous familiariserez avec les concepts Unix, et plus vous les utiliserez, plus vous les apprécierez. Aussi ne puis-je que vous souhaiter bonne continuation !

Annexe A. Compilation et mise à jour des principaux composants du système

Les paragraphes suivants contiennent les remarques et les options à choisir pour compiler les principaux composants du système. La compilation de GCC et du noyau a déjà vue et ne sera donc pas décrite ici. Les informations fournies ci-dessous ne sont pas très détaillées, car la plupart des gens n'en n'ont pas besoin. Elles relèvent plus de la création du système que de son installation, mais elles ont été conservées car elles peuvent être utiles pour les personnes désirant approfondir sérieusement le sujet et disposant d'un certain niveau en informatique.

A.1. Compilation de make 3.79.1

Le programme **make** est utilisé systématiquement lors des compilations. Certaines compilation nécessitent la dernière version de **make**, qui n'est pas forcément fournie avec votre distribution. Vous pouvez donc avoir besoin de le recompiler. La version courante de **make** est la 3.79.1, il est fortement recommandé de l'utiliser.

La compilation de **make** est tout à fait classique. Elle nécessite de :

- détarrer les fichiers sources dans `srcdir` ;
- créer un répertoire pour les objets et s'y placer ;
- taper :

```
CFLAGS=-O2 ../srcdir/configure --enable-shared --host=i686-pc-linux-gnu \
    --prefix=/usr
```

- lancer **make**.

Lorsque la compilation sera terminée, vous pourrez tester le nouveau programme avant de l'installer. Pour cela, il suffit de taper la commande suivante :

```
make check
```

Le programme **make** ainsi configuré sera installé avec la simple commande suivante :

```
make install
```

Le répertoire d'installation est le répertoire `/usr/bin` si le préfixe utilisé est `/usr/`.

A.2. Compilation des binutils 2.10.1

L'archive de GCC ne fournit en soi que les compilateurs C et C++, ainsi que quelques outils complémentaires. Elle ne contient pas en revanche les outils de génération et de manipulation des fichiers binaires. Ces outils sont fournis dans un paquetage distinct, les *binutils*.

La qualité des outils de manipulation des fichiers binaires est essentielle pour la génération des programmes à partir de leurs sources. En clair, le compilateur ne fait qu'une partie du travail, le reste revient aux binutils. Il est donc nécessaire de disposer des dernières versions de ces outils, car certaines compilations de grande taille peuvent mal se passer avec les anciennes versions. C'est en particulier le cas de la glibc.

La version courante des binutils est la 2.10.1. Les sources de ce paquetage peuvent, comme à l'acoutumée, être récupérés sur le site Internet du GNU (http ://www.gnu.org). Leur compilation ne pose pas de problème, puisqu'ils utilisent le très classique script de configuration de GNU. La configuration se fera donc avec la commande suivante :

```
CFLAGS=-O2 ./configure --prefix=/usr
```

La compilation et l'installation de ces outils pourront alors être faites simplement avec les deux commandes suivantes :

```
make
make install
```

Comme d'habitude, vous pourrez tester que tout s'est bien passé à l'aide de la commande **make check** avant de lancer l'installation.

A.3. Compilation de la librairie C 2.2

La librairie C est la librairie que tous les programmes écrits en C utilisent pour accéder aux fonctions du système. Il s'agit donc d'un composant essentiel dans le système (qui, rappelons-le, est lui-même écrit en C). Recompiler la librairie C est donc une opération très sensible. Vous êtes prévenu.

La compilation de la librairie C est une opération fastidieuse à plus d'un titre. Elle nécessite souvent de recompiler certains outils, et la compilation est longue et consomme énormément de ressources. Enfin, il faut parfois recompiler certaines parties du système après son installation, afin de résoudre des conflits de versions dus à la migration. La dernière version de la librairie C GNU est la 2.2.

Les prérequis sont les suivants :

- le compilateur GCC doit avoir été compilé et installé ;
- le programme **make** utilisé doit être de version récente. Il est fortement recommandé d'utiliser la dernière version de **make** (3.79.1 ou ultérieure), car les version antérieures posent quelques problèmes de compatibilité avec la librairie C GNU 2.2 ;
- vous devez disposer de l'archive de l'extension `linuxthreads` prenant en charge les threads sous Linux. Cette archive peut être trouvée au même endroit que l'archive de la librairie C, par exemple sur le site du GNU (http ://www.gnu.org).

> **Note:** Contrairement aux anciennes versions de la librairie C, il est inutile de télécharger un add-on pour les fonctionnalités de cryptage de la libraire. Celles-ci sont à présent directement intégrées dans les sources de la glibc 2.2.

Le processus de compilation est ensuite assez classique. Il suffit de :

- détarrer les fichiers sources dans `srcdir` ;
- se placer dans le répertoire `srcdir` et détarrer les sources de l'archive de `linuxthreads` ;
- créer un répertoire pour les objets en dehors du répertoire des sources et s'y placer ;
- taper :

```
CFLAGS=-O2 ../srcdir/configure --enable-shared --host=i686-pc-linux-gnu \
    --prefix=/usr --enable-add-ons=linuxthreads
```

Les options de la commande de configuration permettent l'emploi des librairies dynamiques et la compilation des modules additionnels pour le multithreading. Le répertoire d'installation sera `/usr/lib/` pour les librairies, et `/usr/include/` pour les fichiers d'en-têtes. Vous pouvez bien entendu utiliser une autre valeur pour l'option `--host`, selon votre configuration. Normalement, cette option est facultative, car sa valeur est automatiquement détectée par le programme de configuration.

La compilation peut ensuite être lancée avec la simple commande suivante :

```
make
```

Une fois la compilation effectuée, vous pouvez tester la nouvelle librairie avant de l'installer. Pour cela, tapez la commande suivante :

```
make check
```

Il est vivement recommandé d'effectuer ce test. Si la moindre erreur apparaît pendant l'exécution du test, n'installez surtout pas la libraire, vous détruiriez sans aucun doute votre système.

> **Note:** En général, un échec de ce test provient souvent d'une mauvaise génération de la librairie due à l'utilisation de programmes trop vieux sur votre sytème. Il est donc parfois nécessaire de mettre à jour d'autres programmes avant d'effectuer la mise à jour de la librairie C. Par exemple, votre version de **make** peut être obsolète, ainsi que celle de GCC ou encore celle des outils GNU binutils (assembleur, éditeur de liens, archiveur, etc...). Il est recommandé d'utiliser la version 3.79.1 de **make**, et les outils de génération de binaires de version supérieure à 2.10.1. Les sources de tous ces programmes peuvent être trouvées sur le site du GNU (http://www.gnu.org).
>
> La compilation de la librairie C, ainsi que celle des autres composants « lourds » du système, comme le noyau, GCC, XFree86 ou KDE, peuvent stresser votre système d'une manière qu'il n'a jamais connu. Les calculs intensifs peuvent durer plusieurs heures, ce qui fait chauffer les composants de votre ordinateur. Il n'est pas rare de voire certains de ces composants défaillir dans ces circonstances, alors qu'ils se sont toujours comportés apparemment normalement jusqu'à présent. En particulier, vous pouvez avoir des problèmes de surchauffe du processeur,

et des corruptions de données dans les barettes de mémoire. Les symptômes sont en général l'apparition d'une erreur de type « SIG 11 » (erreur de segmentation) sur des programmes très fiables, comme gcc ou le shell. Dans ce cas, il faut identifier et remplacer les composants défecteux (ce n'est pas une blague, ce problème m'est personnellement arrivé, et il est arrivé à bien d'autres personnes dans le monde). Bien entendu, il va de soi qu'il ne faut pas overclocker sa machine lorsque l'on se lance dans des opérations telles que celles-ci.

L'installation se fera enfin avec les deux commande suivante :

```
make install
```

et :

```
make localedata/install-locales
```

La compilation des paramètres internationaux utilisés par la librairie C pour la France se fera par exemple avec la commande suivante :

```
localedef -i fr_FR -f ISO-8859-15 fr_FR
```

Cette commande permet de compiler les fichiers les paramètres d'affichage des monnaies et des dates, ainsi que le jeu de caractère utilisé pour comparer et trier les chaînes de caractères. Dans l'exemple donné ci-dessus, la langue choisie est le français (« `fr` ») tel qu'il est parlé en France (« `FR` »), avec le jeu de caractères ISO-8859-15. Les fichiers compilés sont placés dans le répertoire `/usr/lib/locale/`, à raison d'un sous-répertoire pour chaque locale configurée. Si ce répertoire n'existe pas sur votre machine, vous devrez le créer manuellement, car la commande **localedef** ne le fait pas automatiquement.

Note: Une fois la librairie C compilée, vous aurez peut-être à recompiler GCC et un certain nombre d'autres programmes, car la compatibilité binaire d'une version à l'autre de la librairie C est très relative. Si la version que vous utilisez est un simple correctif de bug cependant, la compatibilité binaire sera sans doute totale, mais il faut savoir qu'à chaque version majeure un certain nombre de fonctionnalités sont modifiées et peuvent nécessité la recompilation de tous les programmes qui les utilisent.

En particulier, l'une des librairies qui pose le plus souvent des problèmes est la librairie des entrées/sorties libio. Cette librairie est utilisée par d'autres librairies d'entrées / sorties, comme par exemple les librairies ncurses et slang, ainsi que la plupart des librairies C++. Un changement dans la librairie libio implique donc une recompilation de toutes ces librairies, du compilateur GCC et de quasiment tous les programmes réalisés en C++. Un problème courant est l'impossibilité de compiler des programmes avec GCC 2.95.2 après avoir migré son système de la librairie C 2.1.3 ou antérieure vers la librairie 2.2. Ce problème est dû justement au changement dans un des fichiers de déclaration de la librairie libio. Vous pouvez corriger ce problème en appliquant un patch aux fichiers d'en-tête de GCC. Ce patch est téléchargeable sur Internet (http://clisp.cons.org/~haible/gccinclude-glibc-2.2-compat.diff). Pour l'appliquer, vous devrez vous placer dans le répertoire `/usr/include/g++/` et taper la commande **patch -p0 < fichier**, où `fichier` est le nom du fichier du patch.

La compilation de GCC a été vue dans le Chapitre 8 et ne sera pas détaillée ici. La compilation des librairies slang et ncurses nécessite, comme pour les autres librairies, de préciser le préfixe `/usr` au lieu de `/usr/local` pour remplacer les librairies de la distribution. Remarquez que ncurses nécessite l'option `--with-shared` au lieu de l'option `--enable-shared` pour utiliser le mécanisme des librairies dynamiques.

Remarquez enfin que le format des fichiers d'internationalisation de la librairie C a été modifié avec la version 2.2. Ceci peut également provoquer des incompatibilité avec les programmes internationnaux qui ont été compilés statiquement avec l'ancienne version de la librairie. Ces programmes devront également être recompilés. Enfin, certains programmes bogués qui fonctionnaient avec les anciennes versions de la librairie ne se comporteront pas correctement avec la version 2.2. Pour ces programmes, il est nécessaire de faire une mise à jour avec la version la plus récente.

A.4. Compilation de OpenSSL

Les communications sur Internet n'ont jamais été considérées comme quelque chose de sûr. C'est pour cela que bon nombre de programmes ont recours au cryptage des données pour assurer des communications fiables. Ce cryptage est une nécessité vitale si vous désirez acheter quelque chose sur Internet ou consulter vos comptes en ligne, tant pour assurer votre protection que celle des intervenants commerciaux. La plupart des sites commerciaux refuseront l'accès aux clients non sécurisés.

La plupart des distributions utilisent pour assurer le cryptage une librairie fournissant un accès standard à la plupart des algorithmes de cryptage : *OpenSSL*. Vous pourrez trouver cette librairie sur Internet (http ://www.openssl.org).

La compilation d'OpenSSL n'est pas compliquée, car un jeu de fichier makefile pour Linux est fourni avec les sources. Vous devez simplement tapez la commande suivante pour les sélectionner :

```
./Configure linux-elf
```

Note: Par défaut, OpenSSL va s'installer dans le répertoire `/usr/local/ssl/`, y compris les fichiers d'en-tête. Il faut donc ajouter le chemin `/usr/local/ssl/` dans les variables d'environnement C_INCLUDE_PATH et CPLUS_INCLUDE_PATH si on veut compiler des programmes qui utilisent SSL (c'est en particulier le cas de KDE). Une autre solution est de demander une configuration explicite et d'utiliser l'option `--prefix` :

```
./config --prefix=/usr
```

pour effectuer l'installation dans le répertoire `/usr/` de votre système.

Une fois la configuration effectuée, il suffit de taper les commandes suivantes pour effectuer la compilation et l'installation :

```
make
make install
```

Comme d'habitude, vous pouvez effectuer un test avec la commande **make test** avant de réaliser l'installation pour vérifier que tout est correct.

A.5. Compilation de XFree86 4.0.2

La compilation de XFree86 est certainement l'une des opérations les plus difficiles à réaliser, car il n'y a pas de programme de configuration automatique des sources. La plupart des sources de la distribution sont fournies par le Consortium X, et ils sont prévus pour pouvoir être modifiés librement par les différentes implémentation. XFree86 est l'une de ces implémentations, et utilise la base de ces sources. XFree86 est en fait une implémentation des serveurs X pour le système X Window fourni par le Consortium X. La configuration de XFree86 doit donc être faite comme indiquée dans la documentation de XFree86 et non pas comme indiqué dans la documentation originale du Consortium X.

Toutes les options sont indiquées dans les fichiers du répertoire `/xc/config/cf/`. Le fichier `site.def` est le fichier de configuration central des sources de X Window. Il lit les options des fichiers de configuration des autres systèmes, et en particulier le fichier `host.def`. C'est pour cela qu'il faut définir les options de configuration dans le fichier `host.def` et non pas dans le fichier `site.def`. Vous trouverez un exemple de fichier `host.def` adapté à Linux dans le fichier `xf86site.def`.

La compilation de XFree86 se déroule donc comme suit :

- détarrez l'archive `X402src-1.tgz` ;
- détarrez l'archive `X402src-2.tgz` si vous désirez compiler également les polices de caractères et la documentation ;
- allez dans `xc/config/cf` et copiez le fichier `xf86site.def` vers `host.def` ;
- éditez `host.def` pour définir les options de compilation de XFree86 ;
- allez dans le répertoire `xc/` et tapez

```
make World
```

> **Note:** XFree86 peut utiliser la librairie FreeType 2.0 pour prendre en charge le rendu de certaines polices de caractères vectorielles, comme les polices TrueType par exemple. Il est fortement recommandé d'installer cette librairie avant de lancer la compilation de XFree86, car cette librairie procure une qualité d'affichage des textes incomparable. Cette librairie peut être déjà installée sur votre système, toutefois, vous en trouverez une copie dans le répertoire `extras/freetype2/`. La compilation de cette librairie est très simple, puisque vous n'avez qu'à exécuter les deux commandes suivantes :
>
> ```
> make setup CFG="--prefix=/usr"
> make install
> ```
>
> Vous pourrez bien entendu changer le répertoire de base pour l'installation des librairies et des fichiers d'en-tête à l'aide de l'option `--prefix` si vous ne désirez pas les installer dans le réper-

toire de votre distribution.

La compilation de X Window suppose qu'il y ait un lien de `/usr/bin/cc` vers le compilateur C natif de votre système, en l'occurrence `/usr/bin/gcc`. Il en est de même pour le préprocesseur. Vous devrez donc créer un lien symbolique de `/lib/cpp` vers `/usr/bin/cpp`.

De même, il faut s'assurer qu'il y ait un lien symbolique de `/usr/lib/libtermcap.so` vers la version courante de cette librairie (allez savoir pourquoi ce lien n'existe pas sur certaines distributions...).

La compilation des documentation de XFree86 nécessite également que vous disposiez des outils de génération de documentation SMGL. Ces outils comprennent OpenJade, un traducteur SGML utilisant des feuilles de styles DSSSL, et les outils doctools de XFree86. OpenJade est normalement fourni avec toute bonne distribution, son installation ne sera donc pas traitée dans ce document. Notez toutefois qu'il faut s'assurer que des liens symboliques `jade` et `nsgmls` existent et référencent respectivement les programmes **openjade** et **onsgmls** pour que la compilation des documentations puisse bien se passer.

Pour l'installation des doctools, on pourra utiliser les sources fournis avec XFree86. La compilation et l'installation se font ensuite avec les commandes **make** et **make install**. Cette installation mettra en place un script **sgmlfmt** dans le répertoire `/usr/local/bin/`. Ce script est un script PERL, mais sur Linux, il ne peut pas fonctionner tel quel, parce qu'il recherche l'interpréteur PERL dans le mauvais répertoire. Il faut donc modifier la première ligne de ce script pour référencer l'interpréteur PERL du système, situé dans le répertoire `/usr/bin/` et non dans le répertoire `/usr/local/bin/`.

Les modifications à apporter au fichier `host.def` sont les suivantes pour Linux :

- fixer l'option `DefaultGcc2i386Opt` à « `-O2 -fno-strength-reduce -fno-strict-aliasing` » ;
- fixer la valeur de l'option `HasTk` à « YES » ;
- fixer la valeur de l'option `HasTcl` à « YES » ;
- fixer les chemins sur les librairies TCL/Tk de la distribution et leurs fichiers d'en-tête. Les librairies sont généralement installées dans le répertoire des librairies du système, à savoir `/usr/lib/`, et les fichiers d'en-tête sont généralement situés dans le répertoire `/usr/include/` ;
- de même, donner le nom des librairies TCL et Tk avec lesquelles les programmes qui les utilisent doivent être liés. Ces noms sont peuvent par exemple être `tcl8.3` et `tk8.3` pour la version 8.3 de ces librairies. Vous devrez bien entendu corriger le nom en fonction de la version dont vous disposez. La dernière version de Tcl/Tk peut être récupérée sur Internet (http ://www.scriptics.com) ;
- indiquer les drivers à compiler dans l'option `XF86CardDrivers`. Vous pouvez compiler plusieurs drivers, pour le cas par exemple où vous auriez plusieurs cartes graphiques installées sur la même machine. Il est recommandé de toujours compiler les drivers génériques `vga` et `vesa`, ainsi que le driver utilisant la fonctionnalité « `virtual frame buffer` » du noyau (driver `fbdev`). Vous pouvez également compiler le driver `AgpGartDrivers` si votre carte est une carte AGP, afin d'accélérer les performances OpenGL de votre carte ;

- décommenter si nécessaire la ligne fixant l'option `BuildFonts` à « NO » si vous ne voulez pas compiler les polices de caractères (en général, on ne les compile qu'une seule fois, lors des mises à jour, ce n'est pas nécessaire) ;
- décommenter si nécessaire la ligne fixant l'option `BuildServersOnly` à « YES » si vous ne désirez compiler que les serveurs X (par exemple pour une mise à jour) ;
- décommenter la ligne fixant la valeur de l'option `HasSgmlFmt` à « YES » si vous avez installé les outils doctools de XFree86 et si vous désirez compiler les documentations ;
- ajouter la ligne suivante pour indiquer à XFree86 d'utiliser la librairie FreeType 2 pour la gestion des polices de caractères TrueType :

  ```
  #define Freetype2Dir /usr
  ```

  Vous devez dans cette ligne indiquer le même chemin que celui que vous avez indiqué lors de la compilation de FreeType.

Vous pouvez laisser les autres options entre commentaires. La compilation de XFree86 se fera alors simplement en exécutant la commande suivante dans le répertoire d'installation des sources :

```
make World
```

comme indiqué ci-dessus.

> **Note:** Notez qu'en général, il n'est pas nécessaire de compiler les polices de caractères. En effet, celles-ci ne sont que très rarement modifiées d'une version à l'autre de X11, et les polices de l'ancienne version peuvent parfaitement convenir. Il en est de même pour la documentation, qui n'est pas toujours mise à jour à chaque version mineure.

L'installation de X Window se fait alors simplement avec les deux commandes suivantes :

```
make install
make install.man
```

A.6. Compilation de KDE 2.1

KDE utilise X Window et la librairie Qt. Il faut donc disposer des librairies et des fichiers d'en-tête de X Window, et compiler la librairie Qt au préalable pour compiler KDE. La version courante de la librairie Qt est la version 2.2.3.

Si la librairie Qt est fournie avec votre distribution, sont répertoire d'installation sera `/usr/lib/`. Dans le cas contraire, elle sera placée dans `/usr/local/lib/`. La compilation se fait sur place, ce qui permet d'éviter d'avoir à l'installer. Elle se déroule de la manière suivante :

- il faut se placer dans le répertoire `/usr/lib/` :

```
cd /usr/lib
```

- les sources doivent ensuite être extraites de l'archive de la librairie Qt :

```
tar xvfz qt-x.xx.tar.gz
```

- les sources sont extraites dans le répertoire `qt-x.xx/`. Il faut renommer ce répertoire en `qt/` (c'est à dire, supprimer les numéros de version) ou faire un lien symbolique `qt` vers ce répertoire (solution permettant de conserver les anciennes librairies, pour les anciens programmes) ;
- il faut ensuite ajouter les variables d'environnement dans le `.profile` ou `.login` (selon le shell utilisé, pour bash, il faut modifier `.profile`) comme indiqué dans le fichier `INSTALL` du répertoire `qt`. Ces variables d'environnement sont utilisées par le processus de compilation. Vous devrez bien entendu corriger les chemins définis par défaut pour référencer le répertoire d'installation des sources de Qt ;
- il faut recharger le fichier `.profile` ou `.login` afin de définir ces variables ;
- lancer la commande **configure** avec les options suivantes :

```
./configure -sm -gif -system-jpeg -no-opengl -thread
```

- lancer la commande **make** pour construire la librairie :

```
make
```

- lancer **ldconfig**.

> **Note:** Il n'est pas nécessaire d'effectuer la compilation des librairies multithreadées pour KDE 2.1. Cependant, on utilisera quand même l'option `-thread` lors de la configuration de Qt, car certains autres programmes utilisant l'environnement de bureau KDE peuvent utiliser les libriaires multithreadées de Qt. C'est par exemple le cas de KWinTV, un programme permettant de regarder la télévision sous Linux.

Une fois la librairie Qt compilée, on peut s'attaquer à KDE. La compilation de KDE suppose que les variables d'environnement de la librairie Qt sont toujours définies, il ne faut donc pas les supprimer après avoir compilé celle-ci.

KDE se configure classiquement, avec le programme de configuration **configure**. Pour chaque composant de KDE, il faut procéder comme suit :

- extraire les fichiers sources de l'archive ;
- appeler **configure** avec les bonnes options ;
- lancer la compilation avec « **make** » ;
- installer les fichiers binaires avec « **make install** ».

En général, KDE est installé dans le répertoire `/opt/kde/`. Il faut donc spécifier les bons préfixes lors de l'appel à **configure**. La ligne de commande à utiliser est donc la suivante :

```
./configure --prefix=/opt/kde
```

Comme d'habitude, il faut éventuellement indiquer le type de machine avec l'option `--host` et demander l'utilisation des librairies dynamiques avec l'option `--enable-shared`.

L'ordre des opérations lors de la compilation de KDE est important, parce que certaines parties de KDE en utilisent d'autres. En particulier, il faut impérativement compiler et installer les librairies. Il est donc recommandé d'opérer dans l'ordre suivant :

- kdesupport ;
- kdelib ;
- kdebase ;
- les autres composants de KDE.

> **Note:** Contrairement aux autres parties du système, il faut compiler KDE dans le répertoire des sources. En effet, certaines librairies fournies avec les sources sont nécessaires pour la compilation correcte de quelques modules non essentiels de KDE.
>
> D'autre part, si KDE est déjà présent sur votre système, il faut le supprimer avant de le recompiler (ou au moins renommer son répertoire si vous n'êtes pas sûr de vous...). En effet, les fichiers d'en-tête et les librairies de l'ancienne version peuvent faire échouer la compilation de la nouvelle version.
>
> Si vous le supprimez complètement, il se réinstallera avec la configuration par défaut. Les scripts de configuration de votre distribution pourront alors très bien ne plus fonctionner sur la configuration de KDE. Dans ce cas, vous aurez certainement quelques problèmes avec les droits fixés par défaut sur certains programmes. En particulier, vous ne pourrez jouer des fichiers son que sous le compte root, car le serveur de son « artsd » de KDE et ses outils connexes ont besoin de privilèges pour utiliser les fichiers spéciaux de périphérique de la carte son. La manière la plus propre pour régler ce problème est de créer un groupe d'utilisateurs nommé `audio` et contenant tous les utilisateurs ayant le droit d'utiliser la carte son, et d'attribuer à ce groupe les fichiers spéciaux de périphérique `/dev/dsp`, `/dev/audio` et `/dev/mixer`. Les droits d'accès en lecture et écriture sur ces fichiers devront ensuite être donnés au propriétaire (`root`) et au groupe d'utilisateurs (`audio`). Normalement, les autres utilisateurs ne doivent avoir aucun droit sur ces fichiers.
>
> Enfin, sachez que KDE 2.1 nécessite la version 1.2 ou plus de l'environnement d'exécution java pour permettre l'utilisation de java et de javascript dans le navigateur Konqueror. Cet environnement peut être téléchargé gratuitement sur le site de Sun (http://www.sun.coml). Les binaires fournis par Sun ne sont pas des logiciels libres, mais ce sont les seuls qui respectent réellement la norme Java (ce qui est logique, puisqu'elle est imposée par Sun). Vous devez télécharger la version 1.2.2 de l'environnement d'exécution Java, et non la version 1.3, qui est incompatible et qui n'est pas encore utilisée par les sites Web. Quoi qu'il en soit, l'installation de cet environnement d'exécution se fait extrêmement simplement. Il est recommandé d'installer les binaires dans le répertoire `/usr/lib/jre1.2.2` et de créer un lien symbolique `/usr/lib/java` vers ce répertoire, afin de permettre l'installation de plusieurs versions de l'environnement d'exécution

java tout en sélectionnant la version 1.2.2 par défaut. Après installation, vous devrez ajouter le répertoire `/usr/lib/java/bin` aux répertoires de votre variable d'environnement PATH pour permettre l'utilisation de Java par les programmes tels que Konqueror. Notez que le support de Java doit également être activé manuellement dans la configuration de Konqueror. Enfin, le script de lancement de la machine virtuelle utilise les commandes **cut** et **head**, et suppose qu'elles sont installées dans le répertoire `/usr/bin/`, alors que la plupart des distributions les placent directement dans le répertoire `/bin/`. Vous aurez donc peut-être à faire des liens symboliques `cut` et `head` sur ces deux commandes dans le répertoire `/usr/bin/`.

A.7. Compilation de Samba 2.0.7

La compilation de Samba 2.0.7. pose quelques problèmes avec la glibc 2.2. D'autre part, le choix des répertoires par défaut utilisés par Samba n'est pas très judicieux sur une machine Linux.

La configuration se fait classiquement avec configure :

```
CFLAGS="-O2 -pipe" ./configure --prefix=/usr --sysconfdir=/etc --with-smbmount
```

L'option `--with-smbmount` permet de compiler également les utilitaires **smbmount**, **smbmnt** et **smbumount**, qui permettent de monter les volumes partagés comme des systèmes de fichiers classiques. Notez que ces programmes ne font pas partie officiellement de Samba et ne sont pas maintenus par l'équipe de développement de Samba. D'autres outils de remplacement sont en cours de développement, et peuvent être utilisés avec l'option `--with-smbwrapper`. Cependant, ces outils ne peuvent pas encore être compilés avec la librairie C GNU 2.2.

Un fois la configuration faite, il est recommandé de modifier les chemins déclarés dans le fichier `makefile`. En effet, ces chemins ne sont pas ceux utilisés par la plupart des distributions, et ne sont pas très cohérents dans la hiérarchie du système de fichiers de Linux. Les variables suivantes devront être redéfinies :

```
# Répertoire des données variables :
VARDIR=/var

# Fichiers de traces des démons smbd et nmbd :
SMBLOGFILE = $(VARDIR)/log/log.smb
NMBLOGFILE = $(VARDIR)/log/log.nmb

# Fichier de configuration de Samba :
CONFIGFILE=/etc/smb.conf

# Répertoire des fichiers de verrouillage :
LOCKDIR=$(VARDIR)/lock

# Répertoire privé pour les fichiers de mots de passe :
PRIVATEDIR=/etc/samba
```

La compilation se fait simplement avec la commande suivante :

```
make
```

et l'installation avec :

```
make install
```

Notez cependant que les outils **smbmnt** et **smbumount** ne sont pas setuid. Leurs droits doivent donc être modifiés avec les deux commandes suivantes :

```
chmod +s /usr/bin/smbmnt
chmod +s /usr/bin/smbumount
```

Annexe B. Formulaire pour la création des lignes de mode de XFree86

Les formules données dans cette annexe donnent les relations entre les différentes valeurs utilisées dans les lignes de mode de XFree86. Elles peuvent vous aider à comprendre les relations entre les principaux paramètres de ces lignes.

Ces formules utilisent des variables qui représentent les valeurs des lignes de mode et les caractéristiques techniques de votre écran. Le tableau donné ci-dessous liste les variables utilisées :

`DC`

« Dot Clock », fréquence de base du balayage. Représente le nombre de pixels balayés par seconde et sert de base de temps pour tous les calculs.

`BW`

« Bandwidth », bande passante de fonctionnement du moniteur. Représente la limite supérieure de la fréquence de base du balayage que le moniteur peut accepter. Cette valeur fait partie des caractéristiques techniques du moniteur. Elle peut être extrapolée à partir de la résolution maximale que le moniteur peut afficher.

`GCMF`

« Graphic Card Maximum Frequency », la plus grande valeur de fréquence de balayage que la carte graphique peut générer. Cette valeur fait partie des caractéristiques techniques de la carte graphique.

`MHF`

« Max Horizontal Frequency », fréquence maximale du balayage horizontal. C'est la plus forte valeur de la plage de fréquences horizontales pour les moniteurs multisynchrones. Cette donnée fait partie des caractéristiques techniques du moniteur.

`MVF`

« Max Vertical Frequency », fréquence maximale du balayage vertical. C'est la plus forte valeur de la plage de fréquences verticales pour les moniteurs multisynchrones. Cette donnée fait partie des caractéristiques techniques du moniteur.

`RR`

« Refresh Rate », taux de rafraîchissement des images du mode graphique. C'est le nombre d'images par secondes générées pour ce mode graphique.

`HR`

« Horizontal Resolution », résolution horizontale du mode graphique. C'est le nombre de pixels visibles dans ce mode graphique.

HFL

« Horizontal Frame Length », longueur totale d'une ligne dans ce mode. C'est le nombre de pixels total du mode graphique, visibles ou non. Les pixels non visibles sont les pixels de la zone de blanking horizontale.

HST

« Horizontal Sync Time », durée minimale des signaux de synchronisation du balayage horizontal. Cette donnée fait partie des caractéristiques techniques du moniteur.

HSL

« Horizontal Sync Length », longueur des signaux de synchronisation du balayage horizontal, exprimée en pixels. Cette donnée peut être déduite directement de la durée des signaux et de la fréquence de base du balayage.

HBT

« Horizontal Blanking Time », durée minimale du blanking horizontal dans ce mode graphique. C'est la durée pendant laquelle le faisceau électronique doit être éteint. Cette donnée fait partie des caractéristiques techniques du moniteur.

HBL

« Horizontal Blanking Length », longueur totale du blanking en pixels. Cette donnée peut être déduite directement de la durée du blanking et de la fréquence de base du balayage.

VR

« Vertical Resolution », résolution verticale du mode graphique. C'est le nombre de lignes visibles dans ce mode graphique.

VFH

« Vertical Frame Height », hauteur totale d'une image dans ce mode. C'est le nombre de lignes total du mode graphique, lignes non visibles comprises. Les lignes non visibles sont les lignes de la zone de blanking verticale.

VST

« Vertical Sync Time », durée minimale des signaux de synchronisation du balayage vertical. Cette donnée fait partie des caractéristiques techniques du moniteur.

VSH

« Vertical Sync Height », nombre de lignes balayées pendant le signal de synchronisation du balayage vertical. Cette donnée peut être déduite de la durée des signaux de synchronisation du balayage vertical, de la longueur totale d'une ligne et de la fréquence de base du balayage.

VBT

« Vertical Blanking Time », durée minimale du blanking vertical dans ce mode graphique. C'est la durée pendant laquelle le faisceau électronique doit être éteint avant et après un retour de balayage vertical. Cette donnée fait partie des caractéristiques techniques du moniteur.

VBH

« Vertical Blanking Height », nombre totale de lignes du blanking vertical. Cette donnée peut être déduite de la durée du blanking vertical, de la longueur totale d'une ligne et de la fréquence de base du balayage.

Certaines des variables mentionnées dans ce tableau peuvent être directement déduites des autres. C'est en particulier le cas de RR, HSL, HVL, VSH et VBH.

Le taux de rafraîchissement dépend bien entendu de la fréquence de base du balayage et du nombre total de pixels à balayer. Il peut donc être calculé directement avec la formule suivante :

$$RR = DC \div (HFL \times VFH)$$

La longueur des signaux de synchronisation horizontale et la longueur du blanking dépendent de la fréquence de base du balayage et de leurs durées respectives. Elles peuvent donc être calculées avec les formules suivantes :

$$HSL = HST \times DC$$
$$HBL = HBT \times DC$$

Le même raisonnement peut être fait pour le nombre de lignes parcourues pendant le signal de synchronisation du balayage vertical et le blanking. Cependant, il ne faut pas utiliser la fréquence de base directement quand on calcule sur les lignes, car cette fréquence est exprimée en pixels par secondes et les lignes contiennent HFL pixels. Il faut donc diviser cette fréquence par HFL pour obtenir la fréquence en nombres de lignes par secondes. Les formules à utiliser seront donc les suivantes :

$$VSH = VST \times (DC \div HFL)$$
$$VBH = VBT \times (DC \div HFL)$$

Par ailleurs, la longueur totale des lignes doit bien entendue être supérieure à la résolution plus la taille de la zone de blanking horizontal. Le même raisonnement peut être appliqué aux nombres de lignes total, à la résolution verticale et au nombre de lignes utilisées pour le blanking. On disposera donc des relations suivantes :

$$HFL \geq HR + HBL$$
$$VFH \geq VR + VBH$$

Le choix des valeurs des lignes de mode est soumis à un certain nombre de contraintes, qui expriment les limitations techniques du moniteur et de la carte graphique. Ces contraintes sont exprimées ci-dessous.

Il va de soi que les signaux de synchronisation de balayages doivent être compris dans le blanking. Les deux premières contraintes sont donc les suivantes :

$$HSL \leq HBL$$
$$VSH \leq VBH$$

Par ailleurs, la fréquence de balayage de base doit bien entendu être tolérée à la fois par le moniteur et par la carte graphique. Elle doit donc vérifier les deux inégalités suivantes :

$$DC \leq BP$$
$$DC \leq GCMF$$

Enfin, il faut que les fréquences de balayage horizontales et verticales utilisées soient inférieures aux fréquences de balayage maximales que le moniteur peut gérer. Cette règle n'est valide que pour les moniteurs multisynchrones. Pour les moniteurs à fréquences fixes, il faut que les fréquences de balayage horizontale et verticale soient strictement égales à celles du moniteur.

La contrainte horizontale donne une troisième condition sur DC :

$$(DC \div HFL) \leq MHF$$

Or :

$$(DC \div HFL) \leq (DC \div (HR + HBL))$$

donc l'inégalité précédente est forcément vérifiée si on s'assure que :

$$DC \div (HR + HBL) \leq MHF$$

soit :

$$DC \leq MHF \times (HR + HBL)$$
$$DC \leq MHF \times (HR + DC \times HBT)$$

d'où la condition suivante :

$$DC \leq (MHF \times HR) \div (1 - MHF \times HBT)$$

(le dénominateur étant positif en pratique).

La contrainte verticale porte sur le taux de rafraîchissement, qui doit être inférieur à la fréquence verticale maximale. Elle permet d'obtenir une troisième condition sur DC :

$$RR = DC / (HFL \times VFH) \leq MVF$$

soit :

$$DC \leq MVF \times HFL \times VFH = MVF \times (HR + HBL) \times (VR + VBH) =$$
$$MVF \times (HR + DC \times HBT) \times (VR + DC \times VBT)$$

ce qui donne l'inéquation suivante :

$$\mathrm{MVF} \times \mathrm{HBT} \times \mathrm{VBT} \times \mathrm{DC}^2 + (\mathrm{MVF} \times \mathrm{HR} \times \mathrm{VBT} + \mathrm{MVF} \times \mathrm{VR} \times \mathrm{HBT} - 1) \times \mathrm{DC} + \mathrm{MVF} \times \mathrm{HR} \times \mathrm{VR} \geq 0$$

Cette inéquation impose une limite inférieure en dessous de laquelle `DC` peut se trouver, ou une limite supérieure au delà de laquelle il peut se trouver. En pratique, la limite supérieure ne peut jamais être atteinte. Il faut donc que la fréquence de base soit inférieure à la première racine de l'équation associée à l'inéquation donnée ci-dessous. Cette équation est du second degré et sa résolution ne pose pas de problèmes particuliers. Certains moniteurs ont un temps nul pour les signaux de synchronisation verticaux. Comme le second membre de cette équation est négatif en pratique, la limite à respecter pour la fréquence de balayage de base est exprimée dans ce cas par l'inéquation suivante :

$$\mathrm{DC} \leq (\mathrm{MVF} \times \mathrm{HR} \times \mathrm{VR}) \div (1 - \mathrm{MVF} \times (\mathrm{HR} \times \mathrm{VBT} + \mathrm{VR} \times \mathrm{HBT}))$$

Appendix C. GNU Free Documentation License

Version 1.1, March 2000

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and

publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same

name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.