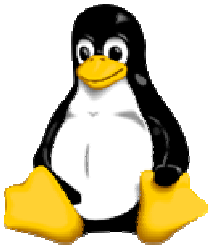
SECTEUR TERTIAIRE
SOUS SECTEUR NTIC

TECHNIQUES DE RESEAUX INFORMATIQUES *(TRI)*

Niveau Technicien spécialisé

Support de cours
Module "Systèmes d'exploitation 'Open Source"'
Juillet 2007

SUPPORT DE COURS



GNU/Linux

PREAMBULE

L'objectif de ce support de cours est de donner les connaissances de base nécessaires à l'installation et l'administration de base 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'information seront donnés 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.

SUPPORT DE COURS

TABLE DES MATIERES

PREAMBULE 1
TABLE DES MATIERES 2
INTRODUCTION 8
CHAPITRE I : LINUX ET LES LOGICIELS LIBRES 9
I.1 Les logiciels libres et la Free Software Foundation 9
I.2 DROITS D'AUTEURS ET LA LICENCE GPL 13
I.3 LA MOTIVATION DES AUTEURS ET LE FINANCEMENT DES LOGICIELS LIBRES 14
I.4 LE PROJET GNU ET LINUX 16
I.5 QU'EST-CE QUE GNU/LINUX ? 17
I.6 POURQUOI LINUX ? 20
CHAPITRE II : INSTALLATION LINUX 23
II.1 CHOIX DE LA DISTRIBUTION 23
II.2 PARTITIONNEMENT ET SYSTEMES DE FICHIERS 24
II.2.1. Notion de partition 24
II.2.2. Notion de système de fichiers 29
II.2.3. Identification des partitions 30
II.3. Choix du plan de partitionnement 32
II.4 PROCEDURE D'INSTALLATION GENERIQUE 35
II.5 EXEMPLE D'INSTALLATION :LA FEDORA CORE 7 36
II.5.1 Ce qu'il faut savoir avant d'installer Fedora 36
II.5.2 Configuration requise pour installer Fedora 37
II.5.2.1 Version PPC 37
II.5.2.2 Version x86 37
II.5.2.3 Version x86_64 37
II.5.3 Installation de Fedora 37
II.5.3.1 Vérification du disque 37

II.5.3.2 Paramétrage des langues 38
II.5.3.3 Choix du partitionnement 39
II.5.3.4 Installation de GRUB 41
II.5.3.5 Configuration réseau 41
II.5.3.6 Choix du mot de passe root 42
II.5.3.7 Installation des programmes 42
II.5.4 Configuration au premier démarrage 43
II.5.4.1 Grub 43
II.5.4.2 Page d'accueil de l'assistant de première configuration 44
II.5.4.3 Configuration du pare-feu 44
II.5.4.4 configuration de SELinux 45
II.5.4.5 Réglage de la date et de l'heure 45
II.5.4.6 Hardware profile 46
II.5.4.7 Création des utilisateurs 46
CHAPITRE III: LES CONCEPTS DE BASE 48
III.1. Introduction aux systèmes d'exploitation 48
III.2 Historique d'Unix 50
III.2.1 Les origines 50
III.2.2 Unix sur Micro 52
III.3 Architecture 53
III.4 Fichiers 54
III.5 Les Commandes. 55
III.6 Connexion et changement de mot de passe. 56
III.7 Vos premiers pas sous Linux 57
III.8 Afficher des fichiers avec la commande ls 59
III.9 Touches spéciales de la console. 60
III.10 Création de fichiers 61
III.11 Caractères permis pour les noms de fichiers. 61
III.12 Répertoires. 62

SUPPORT DE COURS

CHAPITRE IV : SYSTEME ET GESTION DE FICHIERS.....65
IV.1. Définition.....65
IV.2. Les divers types de fichiers 65
IV.2.1. Fichiers ordinaires (ordinary files).....65
IV.2.2. Catalogues (les répertoires ou directory) 65
IV.2.3. Fichiers spéciaux.....66
IV.3. Nomenclature des fichiers 66
IV.4. Chemins 67
IV.4.1. Structure et nom de chemin.....67
IV.4.2. Chemin relatif 68
IV.4.3. Répertoire personnel 70
IV.4.4. ls et quelques commandes intéressantes 70
IV.5. Gestion des fichiers et répertoires.....72
IV.5.1. Création de répertoires.....72
IV.5.2. Suppression de répertoires.....73
IV.5.3. Copie de fichiers.....73
IV.5.4. Déplacer et renommer un fichier.....74
IV.5.5. Supprimer un fichier ou une arborescence 75
IV.5.6. Les liens : plusieurs noms pour un fichier 75
IV.5.7. Critères de recherche sur noms de fichier 78
IV.5.8. Verrouillage de caractères.....79
CHAPITRE V : LES COMMANDES DE BASE, L'EDITEUR VI ET LES REDIRECTIONS.....81
V.1 Les commandes de base. 81
V.1.1 La commande ls, fichiers cachés, options des commandes. 81
V.1.2 Messages d'erreurs. 82
V.1.3 Les pages du manuel. 83
V.1.4 Les pages d'info. 83
V.1.5 Commandes fondamentales. 84

V.1.6 Recherche de fichiers. 88
V.1.7 Recherche dans les fichiers. 89
V.1.8 Le PATH où les commandes sont recherchées. 91
V.1.9 L'option −.− 92
V.2 L'éditeur vi 93
V.2.1 Commandes de saisie 93
V.2.2 Quitter 94
V.2.3 Déplacement en mode commande 94
V.2.4 Correction 95
V.2.5 Recherche dans le texte 96
V.2.6 Copier-Coller 98
V.2.7 Substitution 98
V.2.8 Autres en ligne de commande 99
V.3 Redirections 100
V.3.1. En sortie 100
V.3.2. En entrée 101
V.3.3. Les canaux standards 102
V.3.4. Filtre : définition 103
V.3.5. Pipelines / tubes 103
CHAPITRE VI : GESTION DES UTILISATEURS, DES GROUPES ET LES DROIT D'ACCES 105
VI.1 Droits associés aux fichiers. 105
VI.2 Le fichier des mots de passe: /etc/passwd. 106
VI.3 Le fichier /etc/shadow. 107
VI.4 La commande groups et /etc/group. 109
VI.5 Création manuelle d'un compte utilisateur. 111
VI.6 Méthode automatique: useradd et groupadd. 112
VI.7 Les droits d'accès 112
VI.7.1. Signification 114

VI.7.2. Modification des droits 115
VI.7.3. Masque des droits 118
VI.7.4. Changement de propriétaire et de groupe 119
CHAPITRE VII : GESTION DES PROCESSUS 121
VII.1. Définition et environnement 121
VII.2. Etats d'un processus 122
VII.3. Lancement en tâche de fond 123
VII.3.1. wait 125
VII.4. Liste des processus 125
VII.5. Arrêt d'un processus / signaux 128
VII.6. nohup 130
VII.7. nice et renice 131
VII.8. time 132
CHAPITRE VIII: LE SHELL 135
VIII.1. Structure et exécution d'un script 135
VIII.2. Les variables 136
VIII.2.1. Nomenclature 136
VIII.2.2. Déclaration et affectation 137
VIII.2.3. Accès et affichage 137
VIII.2.4. Suppression et protection 140
VIII.2.5. Exportation 141
VII.2.6. Accolades 141
VIII.2.7. Accolades et remplacement conditionnel 142
VIII.3. variables système 143
VIII.4. Variables spéciales 145
VIII.5. Paramètres de position 146
VIII.5.1. Description 146
VIII.5.2. redéfinition des paramètres 148
VIII.5.3. Réorganisation des paramètres 149

VIII.6. Sortie de script....150
VIII.7. Environnement du processus....150
VIII.8. Substitution de commande....152
VIII.9. Tests de conditions....152
VIII.9.1. tests sur chaîne....153
VIII.9.2. tests sur valeurs numériques....153
VIII.9.3. tests sur les fichiers....154
VIII.9.4. tests combinés par critères ET OU NON....156
VIII.9.5. syntaxe allégée....157
VIII.10 Les alias....157
VIII.11 Options du shell....159
RESSOURCES....160

SUPPORT DE COURS

INTRODUCTION

Linux est un système d'exploitation moderne bénéficiant de l'ensemble des fonctionnalités d'Unix. Ce n'est pas un produit commercial : c'est un logiciel libre que l'on peut obtenir gratuitement. Il est livré avec toutes les fonctionnalités, les outils et les utilitaires habituellement livrés avec les variantes commerciales d'Unix :

- c'est un système 32 bits (64 bits sur certaines plates-formes) ;
- il est multi-utilisateurs ;
- il est multitâche (multitâche préemptif et non coopératif comme Windows 98) ;
- dans le domaine des réseaux, il prend parfaitement en charge la famille des protocoles TCP/IP et possède bien plus de caractéristiques que la plupart des variantes commerciales d'Unix ;
- il dispose de shells très performants ainsi que de XFree86, une implémentation complète du système X-Window.

Linux possède les caractéristiques idéales pour implémenter un serveur Internet stable, performant, sécurisé et flexible. Les parties pratiques de cette formation s'appuient sur la distribution de Linux : la Fedora Core.

La Fedora est faite pour le confort de l'utilisateur final. Elle est à recommandé à tous ceux qui veulent utiliser leur machine rapidement sans passer trop de temps à jouer le rôle de l'ingénieur système.

SUPPORT DE COURS

CHAPITRE I : LINUX ET LES LOGICIELS LIBRES

I.1 Les logiciels libres et la Free Software Foundation

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 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. Cela étant dit, les logiciels libres sont souvent de facto gratuits, car ils sont librement redistribuables par quiconque en possède une copie.

La liberté de modifier les logiciels libres implique naturellement que leur code source, c'est à dire le texte de leur programme tel qu'il a été saisi par ses auteurs, soit librement accessible et modifiable. Les logiciels libres sont donc qualifiés de logiciels « **Open Source** », ce qui signifie en anglais que les sources du programme sont disponibles.

Attention cependant, tous les logiciels Open Source ne sont pas forcément libres, car il n'est pas toujours possible de modifier ce code source et de le redistribuer librement (éventuellement gratuitement). Ainsi, nombre d'éditeurs de logiciels propriétaires publient leur code source sans pour autant donner de droits supplémentaires à ceux qui les lisent. Certains d'entre eux jouent d'ailleurs explicitement sur cette confusion.

Les logiciels libres disposent d'avantages indéniables par rapport aux logiciels « propriétaires » ou « fermés ». Une liste non exhaustive de ces avantages :

- Les programmes distribués sous licence libre ont souvent été écrits par des passionnés du domaine applicatif auquel ils appartiennent. Les logiciels libres disposent donc souvent des dernières fonctionnalités à

la mode et sont donc généralement extrêmement compétitifs sur ce plan.

- Du fait du grand nombre possible d'intervenants sur les sources des logiciels libres, un grand nombre de possibilités techniques peuvent être explorées, et c'est souvent la meilleure qui est sélectionnée. C'est une forme de sélection naturelle de la meilleure solution. Ainsi, sur le long terme, les logiciels libres sont les plus efficaces en terme de performances.
- Toujours du fait du grand nombre d'intervenants, et surtout de par la possibilité de consulter et de modifier librement le code source, le cycle de détection/identification/correction des bogues est très court. Les logiciels libres sont donc parmi les plus fiables qui se font. On peut considérer qu'un logiciel libre utilisé par un grand nombre de personnes est virtuellement « sans bogue » connu, puisque si tel était le cas il serait immédiatement corrigé.
- La possibilité de repartir d'une base de source existante permet de réaliser des développements beaucoup plus rapidement que dans un modèle fermé. Les logiciels libres sont donc également ceux qui se développent le plus rapidement à coût fixe, et sont certainement les plus rentables en terme de coût global pour la collectivité.
- Afin de garantir l'interopérabilité entre les différents intervenants du monde du logiciel libre, chacun s'évertue à respecter les standards. Les logiciels libres sont donc les plus ouverts, non seulement en terme de code source, mais également au niveau des formats de fichiers et des protocoles de communication. Cela garantie une interopérabilité optimale et l'absence de mauvaise surprise.
- Professionnellement parlant, la disponibilité du code source fournit une garantie de fonctionnement que l'on ne peut pas retrouver ailleurs. En

cas de problème, il est toujours possible de s'en sortir, éventuellement en recourant à des compétences externes pour adapter le logiciel à ses propres besoins.

- Enfin, la disponibilité du code source garantit une pérennité absolue du logiciel, ce qu'aucune société commerciale vendant des logiciels propriétaires ne peut ou ne veut faire.

Il faut admettre également que les logiciels libres ont également des inconvénients. La liste suivante en présente quelques-uns :

- La diversité des logiciels libres a également un revers. L'utilisateur peut avoir à choisir entre plusieurs logiciels, ce qui ne simplifie pas forcément l'apprentissage ou la communication entre les différents utilisateurs de logiciels libres. Prenez par exemple ce guide : il fait la présentation de l'installation de trois distributions Linux, qu'il a fallu choisir parmi les centaines de distributions existantes... De plus, ce n'est pas le seul document traitant du sujet de l'installation et de la configuration de Linux (mais c'est sans doute le plus meilleur, n'est-ce pas ?). Cela ne simplifie pas les choses pour l'utilisateur.
- La diversité des bibliothèques et des outils, ainsi que le nombre d'applications susceptibles de communiquer entre elles, implique une complexité accrue dans le travail d'intégration de tous ces logiciels. Les distributions s'assurent que les logiciels qu'elles fournissent fonctionnent bien ensemble, mais la redondance existe malgré tout et a un coût non négligeable au final, aussi bien pour les distributions que les programmeurs et les éditeurs de logiciels. La description de l'installation de trois distributions dans ce document a également un coût pour son auteur (pfff !). Enfin, même l'utilisation simultanée de plusieurs logiciels peut amener à charger en mémoire de nombreuses bibliothèques ayant pourtant la même fonction, alourdissant le système inutilement.

- Certaines fonctions des logiciels ne seront pas forcément implémentées, si ses auteurs n'y voient pas d'intérêt. Si le logiciel est développé par une seule personne ou une petite équipe, ils peuvent ne pas en avoir les moyens financiers ou temporels. Toutefois, si un nombre suffisant d'utilisateurs la réclament, il est probable qu'une personne ayant les compétences nécessaires pour ajouter la fonctionnalité se manifeste. Mais il est également possible qu'un autre projet soit démarré, ajoutant encore une fois un élément à la complexité de l'écosystème des logiciels libres. Il n'est donc pas rare d'avoir plusieurs logiciels réalisant la même chose, mais qu'aucun ne soit complet !
- Plus spécifiquement, le marché monopolistique de Windows est beaucoup plus grand que celui de Linux. De ce fait, même les éditeurs de logiciels propriétaires rechignent à faire l'effort du portage de leurs logiciels pour Unix/Linux. Ainsi, la logithèque pour les systèmes libres s'en trouve d'autant plus réduite. Cela est particulièrement vrai pour les jeux et les logiciels professionnels, et malheureusement également pour les pilotes de périphériques de certains constructeurs de matériel.
- Quand bien même les éditeurs de logiciels voudraient publier leurs logiciels sous licence libre, ils n'en ont pas toujours le droit, en raison d'accords de licence avec des tiers ou de brevets qu'ils utilisent, et parfois même en raison de la réglementation locale de certains pays. De même, les vendeurs de matériel ne peuvent pas toujours fournir de pilotes libres, ni même d'informations ou de spécifications sur le matériel.

Ces inconvénients sont parfois incontournables, car inhérents à la nature même des logiciels libres. Des reproches injustifiés sont parfois réalisés envers les auteurs des logiciels libres, alors qu'ils ne sont pas en mesure

d'éviter ces problèmes. L'utilisateur doit donc également se faire une raison : après tout, il reçoit déjà beaucoup de la communauté des logiciels libres.

Mais, et c'est là l'essentiel, l'aspect le plus important des logiciels libres est le fait qu'ils garantissent la liberté des utilisateurs par rapport aux éditeurs de logiciels. Le respect des standards, l'ouverture des formats de documents et des protocoles de communication garantissent une interopérabilité absolue, qui permet ainsi à chacun de rester libre de ses choix pour sa solution informatique. Il n'est que trop courant de voir les éditeurs de logiciels enfermer leurs clients dans une dépendance vis à vis d'eux, simplement en leur faisant utiliser des produits fermés et inutilisables sans leur concours.

I.2 Droits d'auteurs et la licence GPL

Il faut bien comprendre que le fait de diffuser un logiciel sous une licence libre ne prive absolument pas son auteur de ses droits. Il en reste l'auteur et, en tant que tel, conserve les droits d'auteurs sur son travail. Il ne fait que concéder la liberté d'exploiter ce travail aux autres. C'est en cela que les logiciels libres se démarquent du domaine publique, dont les logiciels ne sont plus soumis à aucun droit.

Afin de protéger les logiciels libres et leurs auteurs, la **Free Software Foundation** a rédigé la licence **GPL** (abréviation de l'anglais « **General Public License** »). 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. En d'autres termes, elle garantit que la liberté des uns s'arrête là où commence celle des autres.

Cette licence empêche donc l'aliénation du logiciel et sa transformation en logiciel propriétaire, de quelque manière que ce soit. Cela implique que tout logiciel libre sous licence GPL 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. Si vous désirez lire la licence GPL, vous pouvez en trouver une copie dans le fichier /usr/src/linux/COPYING une fois que vous aurez installé Linux.

La FSF a également rédigé d'autres licences plus adaptées aux bibliothèques de programmes et aux documentations libres. Ainsi, la licence **LGPL** (« **Lesser General Public License** ») permet d'utiliser les bibliothèques de programmes dans des programmes propriétaires, et la licence **FDL** (« **Free Documentation License** ») permet de diffuser des documentations libres.

Précisons que **la licence GPL** 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 BSD** (un **autre système Unix libre**) exige également la distribution des sources, mais permet l'appropriation des sources par des sociétés commerciales. De même, **la licence X**, sous laquelle est diffusée l'environnement graphique **X11** qu'utilise Linux, est une licence libre. Quelques outils fournis avec Linux sont distribués avec d'autres licences plus rares.

I.3 La motivation des auteurs et le financement des logiciels libres

Bien que cela ne se situe pas au même niveau philosophique, la question de la motivation des auteurs de logiciel libres et, pour les entreprises, de leur financement, se pose également de manière récurrente. Il n'est en effet pas évident, en première analyse, de déterminer les raisons qui poussent un auteur ou une entreprise à rendre ainsi public son savoir-faire, au risque de se le faire tout simplement voler.

Pour ce qui est des auteurs bénévoles, la motivation provient généralement de l'amusement qu'ils ont à développer ces logiciels et à les partager avec la communauté du logiciel libre. Cette communauté, constituée de l'ensemble des auteurs et des utilisateurs des logiciels libres, leur apporte en général la

reconnaissance, des rapports de bogues ou des idées d'amélioration de leur logiciel, des conseils ou même de l'aide sur des sujets qu'ils ne maîtrisent pas (traduction, nouvelles fonctionnalités, etc.).

De plus, en s'intégrant à la communauté du logiciel libre, les programmeurs amateurs peuvent également récupérer des bibliothèques de programme ou des morceaux complets d'autres programmes, et ainsi voir leur logiciel progresser plus vite. Certains peuvent même se distinguer des autres et se voir approcher par une entreprise pour un emploi sur le logiciel qui leur servait à l'origine de passe-temps ! Il est toujours plus agréable de travailler sur quelque chose qui nous plaît...

Les sociétés quant à elles peuvent financer le développement des logiciels libres qu'elles éditent ou auxquels elles contribuent de plusieurs manières. Quelques sociétés vivent de manière dérivée des logiciels libres qu'elles développent (par vente de produits dérivés, de contrat de support, ou de services complémentaires). C'est notamment le cas des sociétés qui éditent des distributions Linux et des sociétés de service en logiciel libre. D'autres sociétés proposent une double licence, constituée d'une licence libre telle que la GPL pour bénéficier des avantages des logiciels libres, et d'une licence propriétaire, permettant d'obtenir une rémunération de la part des clients qui ne veulent pas se plier aux exigences de la GPL (obligation de redistribution des sources en particulier).

Enfin, de nombreuses sociétés contribuent à des logiciels libres tout simplement parce qu'elles en ont besoin. Il est en effet souvent préférable d'adapter un logiciel libre qui satisfait ses besoins à 80% et de développer les 20% restants, quitte à redistribuer les modifications et améliorations qui y sont apportées, que de redévelopper l'intégralité d'un logiciel équivalent ou d'en financer le développement par une société tiers. Le coût total de développement d'une solution complètement propriétaire est en effet

généralement beaucoup plus élevé. On voit bien que dans ce cas, les développeurs de logiciels libres sont avant tout leurs propres utilisateurs...

Cela dit, il faut être clair sur ce sujet : le logiciel libre rapporte moins que le logiciel propriétaire, tout simplement parce qu'il n'est pas question de monopole ici et qu'on ne peut pas pressurer le client aussi facilement qu'avec une offre logicielle fermée.

Comme on le voit, le logiciel libre est très loin d'être l'apanage de quelques fanatiques, et les mauvaises langues qui considèrent ce modèle économique comme du communisme n'ont assurément rien compris. Bien au contraire, il s'agit là de capitaliser les développement et de réduire les coûts, deux objectifs fondamentaux dans une économie de marché et très souvent mal appliqués à l'informatique ! Quant à ceux qui prétendent que les logiciels libres constituent une forme de dumping dans le domaine informatique, ils devraient plutôt analyser la pertinence de ce modèle économique et voir si, finalement, ce n'est pas tout simplement un modèle plus rentable sur le long terme que le modèle propriétaire. Il suffit de considérer le temps consacré, et le coût probablement pharaonique, du développement de Windows Vista, pour constater que le rapport qualité/prix est loin d'être reluisant, même par rapport son aîné Windows XP...

I.4 Le projet GNU et Linux

La **licence GPL** a été écrite initialement pour le projet **GNU de la Free Software Foundation**, dont le but est de réaliser 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 proposent d'y accéder de la même manière.

Le projet GNU est toujours en cours, puisque **la Free Software Foundation** a déjà écrit la plupart des utilitaires Unix, mais que le cœur 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 GPL**, bien que n'ayant rien avoir avec la **Free Software Foundation**.

Cela signifie qu'il serait en fait plus exact de parler du système « **GNU/Linux** » que de « **Linux** » tout court. Sous cette dénomination, il est clair que ce système est constitué des outils **GNU** fonctionnant sur 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, qui devient très susceptible à ce sujet.

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. Cela doit être attribué au goût des développeurs de **la Free Software Foundation** pour ce genre de définition infiniment récursive. Vous ne saurez sans doute jamais les raisons qui les ont poussés à choisir la lettre 'G' dans leur définition. Cela étant, « GNU » se prononce « gnou » en anglais, et vous trouverez donc souvent la représentation d'un gnou sur les sites Internet de GNU.

I.5 Qu'est-ce que GNU/Linux ?

Linux est le noyau d'un système d'exploitation libre de type Unix, écrit initialement par **Linus Torvalds** en 1991 et auquel un grand nombre de

programmeurs ont contribué par Internet depuis. Les origines de tous les systèmes Unix remontent à la première version d'un système d'exploitation expérimental développé par Dennis Ritchie et Ken Thompson dans les laboratoires **AT&T's Bell Laboratories** en 1969.

Ce système a avant tout été développé par des programmeurs, pour des programmeurs, et reprenait un certain nombre de concepts qui avaient été développés auparavant pour le système d'exploitation **Multics** (abréviation de « **Multiplexed Information and Computing Service** »), dont le rôle était de fournir des services informatiques centralisés à un grand nombre de personnes (un peu comme le Minitel a tenté de le faire par la suite). **Multics** n'a jamais réellement vu le jour, en revanche, le système Unix initial a engendré un grand nombre d'autres systèmes plus ou moins compatibles. Récemment, les différents fournisseurs de systèmes Unix se sont accordés pour définir l'ensemble des fonctionnalités que tous les systèmes Unix doivent supporter, afin de résoudre les problèmes engendrés par les incompatibilités existantes entre ces différents systèmes. Le terme **Unix** est donc un terme générique pour représenter l'ensemble de tous ces systèmes, dont Linux fait partie.

Pour l'anecdote, la dénomination Unix provient de la contraction de « **Unics** » (abréviation de « **Uniplexed Information and Computing Service** »), terme forgé ironiquement pour bien signaler qu'Unix était une version allégée de ce que Multics devait être.

Bien que compatible avec les dernières spécifications Unix, Linux ne contient pas une ligne du code source du système Unix original, ce qui en fait ce que l'on appelle un « **clone** ». Cela dit, il s'agit réellement d'un système Unix à part entière. En tant que tel, il dispose des fonctionnalités fournies par les systèmes Unix : il est multitâche, multi-utilisateur et relativement orienté réseau. Vous aurez donc, avec Linux, un système fiable, fonctionnel et performant.

Comme nous l'avons dit, Linux n'est que le noyau d'un système d'exploitation. Ce n'est donc que le composant de base qui prend en charge toute la gestion du matériel.

Mais quel est donc ce système d'exploitation ? Il s'agit du système GNU/Linux, comprenant donc, outre le noyau Linux, plusieurs autres couches logicielles développés par la Free Software Foundation et d'autres organisations. Ces couches prennent en charge différentes fonctionnalités, telles que l'utilisation de l'ordinateur en ligne de commande, l'affichage graphique, et la gestion complète de l'environnement utilisateur en mode graphique. Parler de Linux en tant que système d'exploitation est donc, encore une fois, un abus de langage. Cela étant dit, nous nous autoriserons à le faire dans la suite de ce document, par souci de simplicité.

Contrairement aux idées reçues, il existe un grand nombre d'applications pour Linux. La plupart de ces applications peuvent être installées avec le système GNU/Linux, ce qui fait qu'en pratique ce système forme un ensemble complet et parfaitement utilisable pour la plupart des tâches courantes.

Les systèmes Linux se présentent généralement sous la forme de « distributions », que l'ont peut acheter dans le commerce ou télécharger sur Internet (de manière tout à fait légale, comme nous le verrons dans le chapitre suivant). Une distribution n'est rien d'autre que le regroupement de l'ensemble des programmes qui constituent le système d'exploitation et des logiciels les plus utiles et les plus connus pour Linux. Une distribution Linux est donc réellement bien plus qu'un système d'exploitation : c'est un tout qui vous permettra réellement d'utiliser complètement votre ordinateur, généralement sans même à avoir à insaller de logiciels complémentaires ! De ce point de vue, Linux est beaucoup plus fonctionnel que les autres systèmes d'exploitation propriétaires, qui sont en pratique livrés « nus ».

I.6 Pourquoi Linux ?

L'installation de Linux peut être une opération relativement compliquée, et l'usage d'un système Unix en général n'est pas à la portée de tout le monde. Même si la qualité des distributions actuellement disponibles s'est grandement accrue ces derniers temps, au point que n'importe qui peut installer un système Linux viable sans trop de problèmes, 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 distributions 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 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. 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 disposent de machines hétéroclites et absolument non standards. 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.

Il faut donc 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 l'ont jamais effectuée ;
- mais le jeu en vaut la chandelle, parce que le système est réellement solide.

CHAPITRE II : INSTALLATION LINUX

II.1 Choix de la distribution

Comme vous le savez peut-être déjà, il existe un grand nombre de distributions Linux. Si l'une d'entre elles était réellement meilleure que toutes les autres, la question du choix ne se poserait pas, et il n'y aurait aucun problème. Mais en réalité, chaque distribution a son public, et le choix doit se faire en fonction de ses besoins et de ses désirs. Chose qui, évidemment, n'est pas forcément évidente quand on n'a jamais installé Linux !

Pour faire concis, et sans vouloir froisser les partisans de chaque distribution, on peut en première approche distinguer les distributions selon leur procédure d'installation. Si vous êtes un débutant total, il est probable que vous cherchiez à installer une distribution « grand public », telle que la Mandriva de la société éponyme, la Fedora Core soutenue par Redhat, ou la SuSE de Novell. D'autres préféreront des distributions « moins commerciales », plus simples et plus fiable, qui ne s'appuient pas forcément sur les toutes dernières avancées technologiques. Ce sera le cas avec les distributions Debian, Ubuntu ou Slackware par exemple. Enfin, les intégristes de la pire espèce s'orienteront vers des distributions « orientées source », dont le principe est essentiellement de reconstruire tout le système lors de l'installation. Ceux-ci seront sans doute ravi avec la distribution Gentoo ou, s'ils veulent réellement tout refaire à la main, avec la distribution LFS. Je pense que vous êtes le plus à même de choisir la distribution qui vous conviendra le mieux, en fonction de vos compétences ou de vos appréhensions...

Heureusement, il existe souvent la possibilité d'utiliser les versions « Live » des distributions. Ces versions sont simplement des versions allégées et conçues pour tenir sur un CD amorçable. Il est donc possible de démarrer son ordinateur sur ces CD et, sans faire la moindre installation sur disque dur, d'évaluer une distribution.

La plupart des distributions fournissent un CD Live (parfois même plusieurs, selon le niveau de fonctionnalité proposé). Vous pouvez télécharger sur Internet leurs fichiers ISO (format des images de CD et de DVD), à partir desquels tout bon logiciel de gravage peut graver un CD ou un DVD. Vous n'avez donc plus qu'à faire vos emplettes et tester...

Le choix de d'une distribution ne préjuge bien sûr en rien de leur valeur, il a fallu tout simplement faire un choix. En pratique toutefois, ce choix a été orienté sur les différents systèmes d'installation existants. En effet, de nombreuses distributions sont dérivées d'autres distributions plus connues ou originelles, ce qui fait qu'il n'y a finalement pas autant de variantes de systèmes d'installation qu'il y a de distributions sur le marché.

Bien que c'est la Distribution Fédora Core,projet supporté par la Redhat, qui soit présentée dans ce document et il sera aisé d'adapter les procédures d'installation de ce document aux autres distributions de même nature.

II.2 Partitionnement et systèmes de fichiers

L'installation d'un système d'exploitation tel que Linux sur un PC touche aux structures de données fondamentales du disque dur et est de ce fait une opération très sensible. Il est donc nécessaire de connaître certaines notions de base afin de savoir ce que l'on est en train de faire. Cette section a donc pour but de vous présenter ce qu'est une partition et un système de fichiers, comment choisir un plan de partitionnement, et comment l'accès aux partitions se fait dans un système Linux.

II.2.1. Notion de partition

Une « partition » est, comme son nom l'indique, une partie d'un disque dur. Les partitions permettent de diviser l'espace de stockage des disques durs en zones indépendantes de taille restreinte. La notion de partition permet de réserver certaines portions du disque dur à un usage particulier, et de bien séparer les données qui se trouvent dans chaque partition. L'opération de « partitionnement » est l'opération de création des différentes partitions d'un disque dur.

L'installation de plusieurs systèmes d'exploitation nécessite souvent d'allouer une partition à chaque système, car les systèmes d'exploitation ne comprennent généralement pas le format des partitions des autres systèmes. Il est également parfois nécessaire, pour un même système, de définir plusieurs partitions, qui seront utilisées à des fins spécifiques. Par exemple, Linux fonctionne nettement mieux si on lui attribue une partition de « swap » (dite aussi « partition d'échange ») pour stocker des données peu utilisées qui se trouve en mémoire, lorsqu'il a besoin de plus de mémoire qu'il n'en est physiquement installée sur la machine. De même, il est possible de créer plusieurs partitions pour séparer les données utilisateurs des programmes, ce qui permet de faciliter les mécanismes de sauvegarde d'une part, et d'assurer une plus grande sécurité des données lors des opérations de maintenance du système d'autre part.

Sur les machines de type PC, chaque disque dur peut être découpé en quatre partitions dites « primaires ». La position, la taille et le type de ces partitions sont enregistrées dans le premier secteur du disque dur, que l'on appelle souvent le « Master Boot Record » (« MBR » en abrégé). Le MBR ne contient que quatre entrées pour la définition des partitions, d'où la limite de quatre partitions primaires. Le type des partitions est un code numérique qui indique le système d'exploitation capable de l'utiliser et sa nature (partition de swap ou système de fichiers par exemple). À titre d'exemple, Linux utilise principalement deux types de partition : les partitions de swap (numéro 82) et les partitions pour les systèmes de fichiers (type 83).

La définition des partitions se fait donc en donnant leur point de départ, leur taille et leur type. Le point de départ et la longueur des partitions sont exprimées en secteurs. Un « secteur » est l'unité de base pour les données des disques durs, qui correspond à un bloc de 512 octets utiles (auxquels s'ajoutent bien entendu d'éventuels octets de contrôle d'erreur, mais qui ne sont manipulés que par le disque dur lui-même et par son contrôleur, et que l'on ne peut donc pas utiliser pour y stocker des données). Cela dit, certains systèmes (ceux de Microsoft) ne permettent

pas une telle finesse dans la définition des partitions et nécessitent de travailler au niveau du cylindre.

Pour comprendre ce qu'est un cylindre, il faut savoir que les données des disques durs sont stockées sur les faces magnétiques de plateaux en rotation, au dessus (et en dessous) desquelles les têtes de lecture/écriture du disque se déplacent radialement. Les données sont donc écrites en cercles concentriques sur les différents plateaux en raison de leur rotation sous les têtes de lecture (contrairement aux microsillons et aux CD, il s'agit bien ici de cercles et non d'une spirale car les têtes de lecture/écriture restent à une position fixe pendant la rotation des plateaux). On appelle ces cercles des « pistes » (« track » en anglais). Chaque tête accède donc à une piste et une seule à un instant donné, sur laquelle les secteurs sont enregistrés. Comme toutes les têtes sont solidaires (elles se déplacent ensemble lorsque l'une d'entre elles doit changer de piste), les différentes pistes des différents plateaux sont accédées simultanément. L'ensemble de ces pistes, situés à un rayon donné pour tous les plateaux, constitue ce que l'on appelle un « cylindre ». Les paramètres des disques durs sont donc exprimés en termes de nombre de têtes, de cylindres et de secteurs par piste.

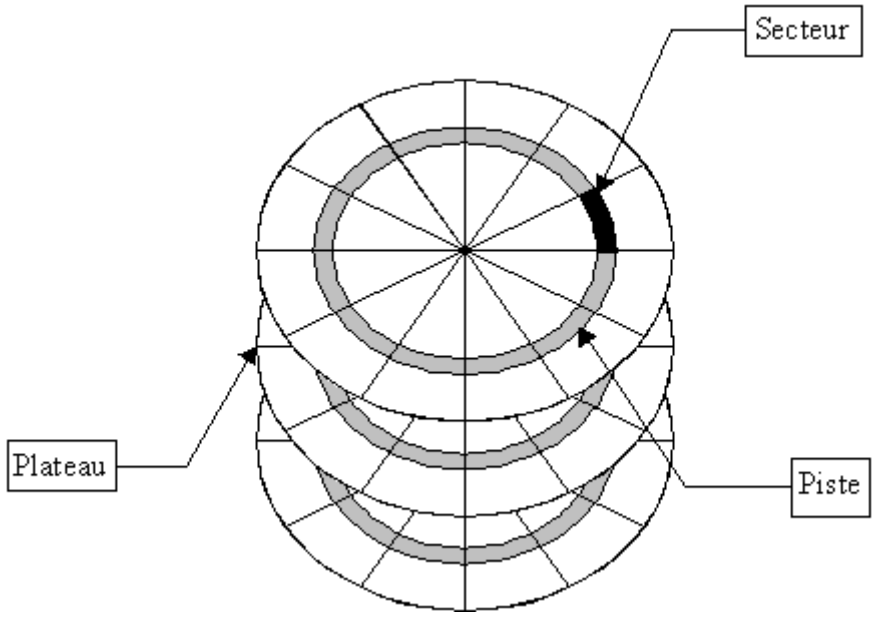


Figure 2-1. Pistes et secteurs d'un disque dur

Vous remarquerez souvent que le nombre de têtes est impair, alors qu'en général, un plateau a deux faces... Cela est dû au fait que les fabricants de disques durs conservent toujours une face d'un plateau pour y écrire des données de contrôle permettant aux têtes de lecture/écriture de se positionner ou pour y placer des pistes complémentaires en cas de zones défectueuses sur la surface de l'un des autres plateaux.

Bien entendu, la limitation à quatre partitions seulement est extrêmement contraignante, aussi la notion de partition étendue a-t-elle été introduite. Une « partition étendue » est une partition primaire spéciale, dans laquelle il est possible de définir jusqu'à 64 sous-partitions. Ces sous-partitions sont appelées des « partitions logiques ». Les données ne sont jamais stockées dans la partition étendue elle-même, mais dans ses partitions logiques. On ne peut définir qu'une seule partition étendue sur un disque donné, mais cela n'empêche pas d'avoir des partitions primaires normales à côté de celle-ci. Il est donc recommandé, lorsque l'on crée la quatrième partition, de créer une partition étendue et non une partition primaire, afin de se réserver la possibilité de créer de nouvelles partitions ultérieurement. Il faut toutefois savoir que certains systèmes ne peuvent pas être installés sur des partitions logiques (notamment DOS et Windows 9x/Millenium), bien qu'ils soient capables d'y accéder une fois qu'ils ont démarré.

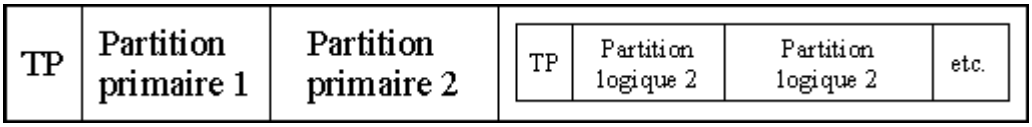


Figure 2-2. Partitions primaires et partitions logiques

Outre la table des partitions primaires, le MBR contient un petit programme appelé le « bootstrap loader » qui permet de charger le premier secteur d'une des partitions primaires. Ce secteur est communément appelé le « secteur de boot », parce qu'il contient le programme capable de charger le système d'exploitation. La partition dont

le secteur de boot est chargé par le bootstrap loader est appelée la « partition active ». Il ne peut y avoir qu'une seule partition active à chaque instant : celle du système d'exploitation principal.

Généralement, le programme stocké sur le secteur de boot d'une partition a pour but de charger le système d'exploitation qui y est installé. Cependant, pour certains systèmes d'exploitation, ce programme est très évolué et permet de lancer d'autres systèmes d'exploitation, éventuellement installés sur d'autres partitions ou d'autres disques durs. Ces programmes sont alors appelés des « gestionnaires d'amorçage ».

Linux dispose de deux gestionnaires d'amorçage très puissants : le **GRUB** et **LILO**. Windows NT, 2000, XP disposent également d'un gestionnaire d'amorçage capable de lancer d'autres systèmes d'exploitation : **NTLDR**. Windows Vista fournit son propre gestionnaire d'amorçage, **bcdedit**, qui est relativement difficile à utiliser.

Pour résumer, lors du démarrage d'un PC, le programme d'amorçage de la machine (communément appelé le « BIOS ») charge le MBR du premier disque dur en mémoire et exécute le bootstrap loader. Celui-ci cherche ensuite à charger le secteur de boot de la partition active, et exécute le gestionnaire d'amorçage qui s'y trouve. Ce gestionnaire peut donner accès aux différents systèmes d'exploitation, qu'ils soient situés sur d'autres partitions ou même d'autres disques durs.

La manière dont chaque système est lancé dépend ensuite du système. Il faut donc, en général, lancer chaque système d'exploitation avec son propre chargeur. Cependant, on peut toujours utiliser un gestionnaire d'amorçage d'un autre système en rusant quelque peu : il suffit d'indiquer à ce gestionnaire de charger le secteur de boot de la partition d'installation du système que l'on désire lancer si celui-ci n'est pas pris en charge directement. Dans ce cas, le gestionnaire d'amorçage ne fait que passer la main au chargeur de l'autre système.

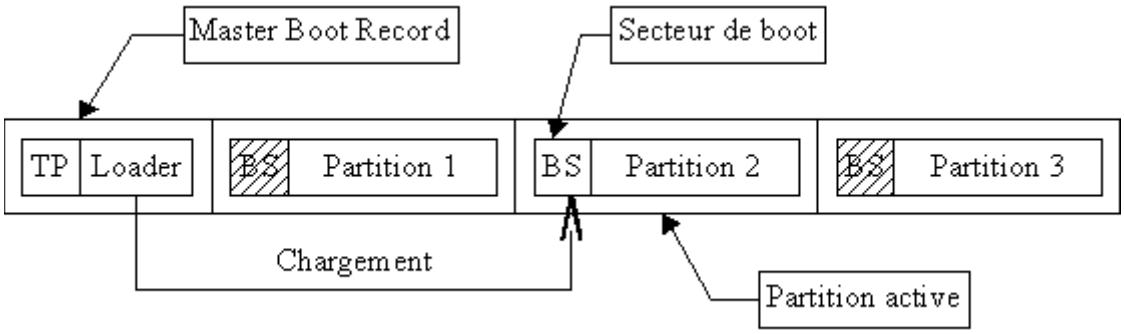


Figure 2-3. Master boot record et secteurs de boot

II.2.2. Notion de système de fichiers

Les systèmes d'exploitation utilisent généralement les partitions pour y stocker ce que l'on appelle des « systèmes de fichiers ». Nous allons voir à présent ce qu'est un système de fichiers, et comment ils sont manipulés sous Unix.

Un système de fichiers est une structure de données sur disque contenant des fichiers, dans lesquels applications peuvent y stocker leurs données. Les fichiers eux-même sont des entités capable de contenir 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.

La plupart des systèmes de fichiers sont structurés hiérarchiquement, et regroupent les fichiers dans des répertoires. Cette organisation permet de classer les fichiers et de pouvoir les retrouver plus facilement. Il existe donc un répertoire racine, qui contient tous les fichiers soit directement, soit indirectement dans ses sous-répertoires. 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.

Il existe de nombreux types de systèmes de fichiers, pour chaque système d'exploitation. Les plus connus dans le monde Windows sont la FAT (système de fichier originel du DOS) et la FAT32, ainsi que NTFS (système de fichiers de Windows NT4, 2000, XP et Vista) et ISO9660 (système de fichiers des CD-ROM et DVD). Sous Linux, on retrouve, outre les systèmes de fichiers Windows, les systèmes Ext2 et Ext3, ReiserFS v3, XFS, JFS, etc. Chacun de ces systèmes de fichiers a ses avantages et ses inconvénients, aussi bien en termes de performances et de limites que de fiabilité face aux défaillances matérielles. En pratique, les systèmes de fichiers les plus utilisés sous Linux sont Ext3 et ReiserFS v3.

II.2.3. Identification des partitions

Comme nous le verrons plus tard, sous Linux, tous les périphériques sont considérés comme des fichiers. Cela signifie qu'ils peuvent être manipulés via des fichiers spéciaux, sur lesquels les opérations classiques des fichiers peuvent être réalisées (et notamment la lecture et l'écriture). Les disques durs, et même leurs partitions, ne font pas exception à la règle, aussi est-il utile de préciser les conventions de nommage utilisées par Linux pour représenter ces entités.

Hormis pour les communications réseau, les fichiers spéciaux de périphériques sont par conventions tous situés dans le répertoire **/dev/.** C'est donc dans ce répertoire que l'on trouve les fichiers spéciaux de périphériques pour les disques et autres périphériques de stockage de masse. Malheureusement, leur nom dépend de leur interface de connexion, ce qui fait que l'on ne peut pas écrire de procédure générique pour leur manipulation.

Ainsi, historiquement, les périphériques IDE sont accessibles via des fichiers spéciaux nommés avec des noms de la forme **hdX**, où **'X'** est une lettre identifiant le disque sur le bus IDE. Par exemple, le lecteur maître du premier contrôleur IDE est accessible via le fichier spécial de périphérique **/dev/hda** (qu'il s'agisse d'un disque dur ou d'un lecteur de CD/DVD), tandis que le lecteur esclave de ce même contrôleur sera accessible via le fichier spécial de périphérique **/dev/hdb**. Les lecteurs maître et

esclave du deuxième contrôleur IDE seront quant à eux accessibles via les fichiers spéciaux de périphérique **/dev/hdc** et **/dev/hdd**, et ainsi de suite.

Inversement, les périphériques SCSI sont accessibles via des fichiers spéciaux de périphériques dont le nom est de la forme sdX. On pourra donc accéder aux divers disques SCSI via les fichiers spéciaux de périphérique **/deb/sda, /dev/sdb**, etc.

Afin d'uniformiser la manière dont les périphériques sont accédés, la tendance est de considérer les nouveaux périphériques comme des périphériques SCSI. De ce fait, la terminologie sdX est de plus en plus employée. Ainsi, les lecteurs de connectique Serial ATA, USB ou Firewire sont tous accédés sous cette terminologie. De plus, un nouveau jeu de pilotes pour les périphériques IDE est en cours de développement, et il utilise déjà cette terminologie. Il n'y aura donc plus lieu de faire la distinction entre les interfaces utilisées sous peu.

Les fichiers spéciaux de périphérique des disques peuvent être utilisés directement, même si ce n'est pas l'usage de le faire. Par exemple, il est possible de lire la totalité du contenu d'un disque en lisant son fichier spécial de périphérique. En pratique toutefois, ces fichiers spéciaux de périphériques ne sont utilisés que pour manipuler la table des partitions. Ils ne sont généralement utilisés que par les programmes de partitionnement, donc que lors de l'installation du système ou lors de l'installation d'un nouveau disque.

Les partitions d'un disque quant à elle sont accessibles sous la terminologie **hdXn** ou **sdXn**, où **'X'** est toujours la lettre du lecteur, et 'n' est le numéro de la partition considérée (la numérotation des partitions commence à partir de 1). Ainsi, la première partition du disque dur maître du premier contrôleur IDE est accessible via le fichier spécial de périphérique **/dev/hda1**, et ainsi de suite.

Ces fichiers spéciaux de périphériques peuvent également être utilisés directement, par exemple lorsque l'on veut copier une partition. Toutefois, en pratique, ces fichiers spéciaux sont surtout utilisés par les programmes permettant de créer les systèmes de fichiers.

Comme il l'a déjà été dit ci-dessus, il est tout à fait possible de créer un système de fichiers dans un fichier... Il suffit en effet de donner le nom de ce fichier en paramètre à la commande de création du système de fichiers, en lieu et place du nom du fichier spécial de périphérique de la partition ou du disque !

Certains utilitaires non spécifiques à Linux utilisent d'autres conventions de nommage pour identifier les disques et les partitions. C'est notamment le cas pour le gestionnaire d'amorçage GRUB. En effet, ce gestionnaire ne fait pas d'hypothèse sur le système utilisé, et ne peut donc pas reprendre la terminologie Linux.

II.3. Choix du plan de partitionnement

Le partitionnement du disque peut généralement être réalisé automatiquement par les programmes d'installation des systèmes d'exploitation. Cependant, les choix faits par ces programmes ne sont pas toujours très judicieux, et peuvent ne pas convenir si votre disque contient des partitions déjà existantes. Nous verrons donc comment réaliser un partitionnement manuellement lors de l'installation de Linux.

Toutefois, avant de se lancer dans cette opération, il faut établir un plan de partitionnement. Cela consiste tout simplement à déterminer la taille et la nature de chaque partition dans le système. Il est normal, sur un système où seul Linux sera utilisé, de disposer d'au moins trois partitions :

- Une partition d'échange, que Linux utilisera pour y stocker temporairement des données lorsqu'il aura besoin de récupérer un peu de place en mémoire. Il est recommandé de placer cette partition au début du disque (c'est à dire au plus près du cylindre 0, là où le taux de transfert est le plus rapide). La taille de cette partition peut empiriquement être définie comme étant égale à deux fois la quantité de mémoire vive installée sur la machine, sans toutefois dépasser 512Mo. Cette limite part du principe que si le système a besoin de plus de mémoire virtuelle (somme de la quantité de mémoire vive et de la taille des zones d'échange), c'est que de toutes façons il n'est pas dimensionné

pour faire ce qu'on lui demande. Sachez qu'on peut rajouter des fichiers d'échange a postériori très facilement sous Linux.

- Une partition pour le système de fichiers racine dans laquelle se trouvera l'ensemble des fichiers du système. Selon l'âge de la machine et la version du BIOS, il peut y avoir des limitations en ce qui concerne le début de cette partition. Il est recommandé qu'elle se trouve dans les 1024 premiers cylindres pour que le BIOS puisse y accéder et charger le gestionnaire d'amorçage du système. Sa taille devra être de 4 à 8Go, afin de pouvoir installer le système d'exploitation, les environnements graphiques et la plupart des applications sans problème. Il est inutile de dépasser les 8Go : si cette partition vient à se remplir, c'est que les fichiers temporaires du système, situés dans les répertoires **/tmp/** et **/var/,** prennent trop de place, auquel cas l'activité de la machine justifie l'utilisation d'une partition dédiée pour ces données.
- Une partition devant contenir les données des utilisateurs (qui se trouveront dans le répertoire **/home/**). Elle pourra prendre le reste de l'espace disponible sur le disque moins, bien entendu, l'espace nécessaire aux partitions des éventuels autres systèmes d'exploitation et celui d'une éventuelle partition de sauvegarde ou d'échange de données.

L'avantage d'avoir une partition séparée pour toutes les données des utilisateurs est considérable, puisque dans ce cas on peut mettre à jour le système ou le réinstaller complètement sans avoir à faire de sauvegarde de ses données. De plus, les fichiers de configuration importants peuvent être sauvegardés sur cette partition avant la réinstallation, ce qui est extrêmement pratique.

Ce type de partitionnement est donc à prendre sérieusement en considération, surtout pour les machines de particuliers, sur lesquelles un grand nombre de programmes peuvent être installés simplement pour les tester. Il n'est donc pas rare, dans ces conditions, d'avoir à refaire une installation complète pour « nettoyer » rapidement le système.

Toutefois, si l'on ne dispose pas de beaucoup de place sur le disque, il est possible de regrouper la partition racine et la partition contenant les données utilisateurs. Un mauvais dimensionnement de ces partitions aura dans ce cas de moins lourdes conséquences. En effet, lorsqu'une partition est pleine, on ne peut pas facilement utiliser l'espace restant sur les autres partitions pour l'agrandir, car il faut déplacer toutes les données au préalable.

Si votre machine est destinée à accueillir plusieurs systèmes d'exploitation, il est peut être intéressant de créer au moins une partition FAT ou FAT32. Cette partition permettra en effet d'échanger des données entre Linux et les autres systèmes d'exploitation, car les systèmes de fichiers FAT sont reconnus par tous les systèmes d'exploitation courants. Notez que si Windows NT4 doit être installé, vous devrez créer une partition FAT plutôt qu'une partition FAT32, car Windows NT ne reconnaît pas, sans programmes additionnels, les partitions FAT32. Ce problème ne se pose plus pour Windows 2000 et les suivants.

Notez également que bien que Linux sache parfaitement lire les partitions NTFS (utilisées par Windows NT4, 2000, XP et Vista), l'écriture sur ces partitions n'est pas complètement implémentée (il n'est possible d'écrire que dans un fichier existant). Inversement, aucun système Windows ne sait et ne saura lire les systèmes de fichiers Linux, quels qu'ils soient. De même, les systèmes Windows ne savent pas accéder à un système de fichiers stocké dans un autre fichier, sans installer d'outils complémentaires (pour ceux qui disposent de Windows XP, l'outil filedisk sera sans doute relativement utile). Avoir une partition FAT est donc souvent la solution la plus simple pour échanger des informations entre les deux systèmes.

Sachez cependant que la taille maximum des fichiers sur les partitions FAT32 est limitée à 4Go. Cette limitation est en dessous de la taille d'un DVD, cela pourra donc être assez gênant à l'occasion pour transférer de gros fichiers tels que des fichiers images de DVD ou des fichiers vidéo non compressés. De plus, Microsoft ne recommande pas que des systèmes de fichiers FAT de plus de 32 Go soient utilisés,

même s'il est possible de créer de tels systèmes de fichiers avec des utilitaires spécifiques. Les systèmes de fichiers Linux n'ont pas ce genre de limites.

Dans ce cas également, vous devrez prévoir une ou deux partitions pour le deuxième système d'exploitation, afin de séparer les données de ce système et les données des utilisateurs. Vous aurez alors certainement à créer une partition étendue et des partitions logiques, pour éviter d'être limité aux quatre partitions primaires. On constate ici que le fait que les données des utilisateurs ne puissent pas être partagées entre les systèmes est très pénalisant...

Rien ne vous empêche de créer d'autres partitions si vous le désirez. Par exemple, si la machine doit être d'une fiabilité absolue ou si vous êtes soumis à des contraintes d'exploitation fortes, vous pouvez opter pour des solutions radicales qui consistent à séparer les données d'exploitation (normalement situées dans le répertoire ***/var/***) des fichiers des programmes, et de les placer dans une partition dédiée. Vous pourrez alors ne monter que cette partition en lecture/écriture. Ainsi, en cas de crash système, seule la partition contenant les données d'exploitation devra être réparée, ou à l'extrême rigueur réinitialisée complètement par les scripts de démarrage.

Vous voyez que définir un plan de partitionnement n'est pas une chose facile, et il n'existe pas de solution générique qui convienne à tous les usages. Cela est d'autant plus vrai qu'il est impératif de bien déterminer ses besoins en espace disque, aussi bien pour les programmes que pour les données et le swap, puisque les partitions ne peuvent pas facilement être redimensionnées. Quoi qu'il en soit, dans tous vos choix, gardez à l'esprit que ce qui est le plus important pour un particulier, ce sont ses données.

II.4 Procédure d'installation générique

Si vous avez fait la sauvegarde de vos données, que vous savez comment partitionner vos disques et où installer Linux, vous pouvez vous lancer effectivement dans l'installation de Linux.

De manière générale, une installation de Linux se déroule selon les étapes suivanes :

- amorçage du système ;
- création ou redimensionnement des partitions du disque dur ;
- création des systèmes de fichiers et de la partition d'échange ;
- installation du système proprement dite ;
- installation du gestionnaire d'amorçage ;
- configuration du système.

II.5 Exemple d'Installation :La Fedora Core 7

Nous allons présenter dans cette section la procédure d'installation de la distribution Fedora Core 7. Comme nous allons le voir, cette distribution simplifie réellement la procédure d'installation et vise à être particulièrement conviviale.

L'installation peut être débutée simplement en plaçant le CD/DVD d'installation dans le lecteur de CD/DVD de l'ordinateur et en configurant le BIOS pour qu'il démarre sur ce CD/DVD.

II.5.1 Ce qu'il faut savoir avant d'installer Fedora

Pour installer Fedora il vous faut d'abord récupérer un DVD ou les CD d'installation. Pour cela vous avez différents moyens, soit acheter le DVD dans un magazine consacré à GNU/Linux dans les librairies, soit télécharger les images des CD et du DVD à l'aide des liens disponibles sur la page d'accueil du site. Ces liens vous proposeront différentes versions de Fedora :

- une version x86_64 qui correspond au processeur en 64 bits,
- une version x86 qui correspond à la version i386 du noyau Linux pour les processeurs en 32 bits,
- une version ppc qui correspond aux versions PowerPC 32 et 64 bits (ç.-à-d., non Mac Intel) pour les mac.

II.5.2 Configuration requise pour installer Fedora

- Disque Dur : Taille nécessaire variant en fonction des composants installés mais une installation complète prend près de 9Go sur le disque.

II.5.2.1 Version PPC

- PowerPC G3 / POWER3 a 233 MHz et 128 Mio de memoire vive pour une utilisation en mode texte;
- PowerPC G3 / POWER3 a 400 MHz et 256 Mio de memoire vive pour une utilisation en mode graphique.

Fedora supporte egalement : IBM pSeries, IBM iSeries, IBM RS/6000, Genesi Pegasos II, IBM Cell Broadband Engine machines (Playstation 3)...

II.5.2.2 Version x86

- pentium ou equivalent a 200 MHz et 128 Mio de memoire vive pour une utilisation en mode texte;
- pentium II ou equivalent a 400 MHz et 192 Mio de memoire vive (256 Mio recommande) pour une utilisation en mode graphique.

II.5.2.3 Version x86_64

- Un processeur 64 bit et 256 Mio de memoire vive pour une utilisation en mode texte;
- Un processeur 64 bit et 384 Mio de memoire vive (512 Mio recommande) pour une utilisation en mode graphique.

Une fois que vous avez le DVD de Fedora, introduisez le dans le lecteur et redémarrez votre ordinateur.

II.5.3 Installation de Fedora

II.5.3.1 Vérification du disque

La première image que vous obtenez lorsque votre ordinateur démarre sur votre CD/DVD est la suivante (choisissez bien l'option pour booter sur le lecteur optique, cf la documentation de la carte mère) :

Plusieurs possibilités s'offrent alors à vous :

- Installer ou mettre à jour Fedora mode graphique ;
- Installer ou mettre à jour Fedora en mode texte ;
- Dépanner une installation éxistente ;
- Booter depuis le disque dur.

La suite de ce tutoriel considèrera une installation en mode graphique.

Avant de poursuivre l'installation, le système vous demande si vous voulez vérifier l'intégrité de vos/votre CD/DVD. Il est fortement recommandé d'effectuer cette vérification, cela vous évitera la mauvaise surprise de fichiers illisibles lors de l'installation par exemple.

II.5.3.2 Paramétrage des langues

Après la vérification de l'intégrité de votre support, le système vous demande de choisir votre langue puis la disposition de votre clavier.

Pour un clavier "azerty" français vous avez le choix entre le Latin 1 et le Latin 9, ce deuxième vous donnant plus de caractère pour la langue française (notamment le signe €), il est donc fortement conseillé de le choisir. Avec fedora 7, latin 9 est devenu le choix par défaut.

Si vous avez un clavier avec une disposition différente (belge, luxembourgeoise, québecoise, etc), merci de nous donner la meilleure disposition par un message sur le forum.

II.5.3.3 Choix du partitionnement

Après avoir choisi votre configuration linguistique vous pouvez choisir la manière dont Fedora est installée sur votre disque dur. Les 3 paragraphes suivants permettront de vous guider suivant votre système actuel et votre niveau.

II.5.3.3.1 C'est votre première installation et votre premier contact avec un système GNU/Linux

Il est alors conseillé aux débutants de laisser le partitionnement par défaut (2e ou 3e option en fonction de vos possibilités). Le partitionnement par défaut va adopter la structure suivante :

- **/boot** : partition sur laquelle sont installées les informations nécessaires au démarrage,
- **/** : partition sur laquelle est installée le système,
- **SWAP** : partition dont le système se sert pour décharger la mémoire (RAM) lorsqu'elle atteint un certain niveau.

II.5.3.3.2 Si vous avez déjà un système d'exploitation sur votre ordinateur

Si vous avez un autre système d'exploitation présent sur votre ordinateur et que vous désirez le conserver, il est nécessaire que vous prépariez le partitionnement. La meilleure solution est de partitionner depuis un logiciel de partitionnement (Gparted - logiciel libre et gratuit -, partition magic - logiciel non libre et payant) pour créer un espace non partitionné, dans lequel seront créées et formatées les partitions lors de l'installation de Fedora. La dernière version du live cd de Gparted (juin 2007) gère les partitions NTFS de Windows Vista.

II.5.3.3.3 Si vous êtes un utilisateur avancé

Dans ce cas vous pouvez choisir un partitionnement personnalisé, où vous pouvez choisir les partitions que vous souhaitez créer/formater/supprimer.

Exemple de partitionnement communément admis :

- **/boot** partition sur laquelle va se lancer Fedora (utile en cas de multiboot et ne prends que très peu place). Optez pour une taille d'environs 100Mo,
- **/** partition sur laquelle s'installe le système. Variez la taille de cette partition en fonction de votre disque dur mais sachez qu'une installation complète du DVD/CD prendra environ 9 Go,
- **SWAP** correspond à la partition SWAP, historiquement elle était fixée à deux fois la valeur de la RAM de l'ordinateur. Aujourd'hui il est admis que même s'il

vaut mieux une SWAP supérieure à la RAM présente dans l'ordinateur, il n'est plus nécessaire de mettre une valeur aussi élevée,

- **/home** Partition sur laquelle sont conservées les données de l'ensemble des utilisateurs. Faire une partition /home séparée permet de simplifier la transition des données utilisateurs, notamment lors des changements de version de Fedora.

II.5.3.4 Installation de GRUB

Vous devez choisir l'emplacement d'installation de "**GRUB**" (**Grant Unified Boot loader**). "**GRUB**" est ce qu'on appelle un chargeur de démarrage, en d'autres termes un programme qui permet de choisir et de lancer un système d'exploitation. Si vous n'avez qu'un seul disque, a priori, les options par défauts conviennent. Si vous avez plusieurs disques, faites attention au disque sur lequel GRUB va s'installer.

II.5.3.5 Configuration réseau

Vous pourrez y saisir les caractéristiques de votre réseau pour que Fedora se connecte à internet. Vous n'avez rien à faire si le PC est branché directement au modem en ethernet. Si vous possédez plusieurs ordinateurs en réseau, il est peut être nécessaire de spécifier les paramètres manuellement. La connexion de votre ordinateur au modem en USB est déconseillée, des pilotes non présents sur le DVD peuvent être requis.

L'écran suivant vous permet de choisir le fuseau horaire de votre lieu de résidence de manière à régler l'horloge de votre système d'exploitation.

II.5.3.6 Choix du mot de passe root

L'utilisateur ***root*** est un super utilisateur qui sert à administrer le système, son usage doit être réservé à des cas bien particuliers car cet utilisateur possède les pleins pouvoirs sur l'ensemble de votre système.

II.5.3.7 Installation des programmes

Lors de l'installation, Fedora vous propose trois catégories de programmes que vous pouvez sélectionner en fonction de l'usage que vous aller avoir de votre ordinateur. Vous pouvez aussi décider d'augmenter le nombre de programmes disponible en ajoutant des dépots externes.

Si vous en avez un usage plus particulier ou que vous désirez installer des programmes spécifiques vous avez la possibilité de les choisir en personnalisant l'installation. Fedora va maintenant copier les fichiers sur votre disque dur, puis vous demandera de redémarrer votre ordinateur après vous avoir invité à retirer votre DVD.

II.5.4 Configuration au premier démarrage

II.5.4.1 Grub

Lors du démarrage de Fedora, GRUB commence par se lancer :

Vous avez 5 secondes (temps par défaut) pour appuyer sur une touche si vous ne souhaitez pas démarrer sur Fedora.

Dans ce cas vous arrivez sur cet écran :

II.5.4.2 Page d'accueil de l'assistant de première configuration

Lors du premier démarrage un certain nombre de paramètres sont à configurer.

II.5.4.3 Configuration du pare-feu

Après l'acceptation de la licence, le système vous demande de configurer le Pare-feu. Il est conseillé de laisser le pare-feu activé. En revanche les options à cocher sont fonction de l'utilisation de votre ordinateur. Pour un poste de travail, aucune ouverture de port n'est nécessaire.

II.5.4.4 configuration de SELinux

Ensuite, vous arrivez sur l'écran de configuration de **SELinux** (**Security Enhanced Linux**), logiciel de sécurité de Fedora. Ce n'est pas un pare-feu, **SELinux** s'occupe de vérifier les autorisations des applications et contrôle leur exécution dans un environnement sécurisé. Par défaut il est recommandé de laisser **SELinux** en mode Strict (ou enforced en anglais).

Trois niveaux vous sont proposés :

- Strict,
- Permissif (ce mode n'est utile qu'à des fins de debogage),
- Désactivé.

II.5.4.5 Réglage de la date et de l'heure

Il vous faut ensuite régler votre horloge :

II.5.4.6 Hardware profile

Cette étape permet d'envoyer la configuration matérielle aux développeurs, aucune données nominatives n'est envoyées. Ce profil est renvoyé tous les mois pour suivre les modifications du matériel. Cette base de données permet aux développeurs de repérer le matériel populaire pour y consacrer plus d'efforts. L'utilisateur peut également s'en servir pour vérifier que son matériel est compatible. Les statistiques sont disponibles à cette page : http://smolt.fedoraproject.org/stats

II.5.4.7 Création des utilisateurs

Puis vous devez créer votre premier utilisateur (nom d'utilisateur et mot de passe) :

L'installation de Fedora 7 est terminée, vous pouvez maintenant vous identifier et commencer à profiter de Votre distribution Linux

CHAPITRE III: LES CONCEPTS DE BASE

III.1. Introduction aux systèmes d'exploitation

L'histoire de l'informatique est très brève – les ordinateurs sont nés avec la seconde guerre mondiale – et pourtant, elle a connu des grandes évolutions. À leur apparition, les ordinateurs étaient très coûteux et réservés aux grandes entreprises; celles-ci n'en possédaient au départ que quelques exemplaires. Ces ordinateurs « centraux » sont rapidement devenu un auxiliaire d'administration et ils se sont diffusés dans les différents services: financier, comptabilité, etc. Pour rendre l'informatique plus adaptée et plus abordable, des fabricants se sont alors mis à produire des mini-ordinateurs « départementaux ». Ces ordinateurs fonctionnaient avec des systèmes d'exploitation qui leur étaient propres, à chaque machine ou à chaque constructeur, par exemple, MVS pour IBM ou VMS pour DEC.

Aujourd'hui, l'informatique, aussi bien dans les entreprises, que dans la recherche ou l'enseignement, utilise des machines plus petites, fonctionnant avec des systèmes d'exploitation à caractère universel. Parmi ces systèmes d'exploitation, deux se distinguent particulièrement, un système mono-utilisateur, Windows, et un autre multi-utilisateurs et multitâches, Unix. D'une manière grossière – et contestable avec l'apparition des réseaux – on peut affirmer que le premier système est destiné à des ordinateurs individuels, tandis que l'autre est réservé au travail en groupe. Les systèmes actuels gèrent, par ailleurs une interface graphique, avec comme pionnier le Finder du Macintosh. Les systèmes d'exploitation actuels ont intégré de façon généralisée le multitâches et le service à plusieurs utilisateurs avec la généralisation des architectures Client-Serveur, par exemple avec OS/2 d'IBM et Windows/NT.

Parmi ces systèmes, Unix, qui est le plus ancien, est celui qui offre le plus de richesses, le plus d'homogénéité et le plus de souplesse; il dispose, dans les versions standards, d'extensions pour les réseaux et pour le graphique. Pour cette

raison, nous l'avons choisi comme le centre de ce cours. Par ailleurs, le système MS-DOS puis Windows, en évoluant, ont incorporé beaucoup de caractéristiques de leur prédécesseur. Les noyaux de ces systèmes se modifieront certainement avec l'évolution des techniques. Cependant, les principes sur lesquels ils se fondent, et à plus forte raison, leur « décor », devraient rester relativement stables, au moins pour les quelques années à venir.

L'étude des systèmes d'exploitation forme une part très importante de l'informatique comme discipline et, à la différence des ses autres domaines, c'est une part qui lui est propre. Ceci au contraire de l'algorithmique ou de la logique, par exemple, qui se partagent avec les mathématiques. C'est aussi une discipline technique qui plus encore que les autres est sujette au renouvellement.

On peut diviser les systèmes d'exploitation classiques en quatre parties principales :

1. Les **processus**, qui correspondent à l'exécution des programmes. Ces processus pouvant s'exécuter simultanément dans un système multitâche. Le système a pour fonction de les créer, de les gérer, de les synchroniser, ainsi que de leur permettre de communiquer entre eux;
2. La **gestion de la mémoire**, qui permet de transférer les programmes et les données nécessaires à la création des processus, d'un support secondaire, par exemple un disque, vers un support central, où a lieu l'exécution des processus. Le système devra garder la trace des parties utilisées et libres de la mémoire ainsi que gérer les transferts entre les mémoires principale et secondaire;
3. Le **système de fichiers**, qui offre à l'utilisateur une vision homogène et structurée des données et des ressources : disques, mémoires, périphériques. Le système gère la création des fichiers, leur destruction, leur correspondance avec les dispositifs physiques, ainsi qu'un certain nombre d'autres caractéristiques, telles que la protection. Il les organise enfin, en général, en une structure arborescente;
4. Les **entrées-sorties**, qui correspondent aux mécanismes qu'utilisent les processus pour communiquer avec l'extérieur. Ces entrées-sorties font largement

appel aux couches les plus proches du matériel, et dont le système tente de masquer les particularités aux utilisateurs.

Le système d'exploitation correspond à l'interface entre les applications et le matériel. Le programmeur d'applications n'aborde que rarement – sinon jamais – son code interne. Il l'utilise par l'intermédiaire d'« appels système ». Les appels systèmes sont souvent accessibles à partir d'un langage de programmation, notamment en C avec le système Unix. Ces appels permettent d'effectuer la plupart des opérations sur les entités du système d'exploitation et, par exemple, de créer et détruire des processus, des fichiers, de réaliser des entrées-sorties, etc. Une terminologie tend à s'imposer pour dénommer l'ensemble des appels système, qu'ils concernent un système d'exploitation ou n'importe quelle d'application informatique : les API (*Application Programming Interface*).

Un utilisateur peut lui aussi – dans une certaine mesure – manipuler un système d'exploitation, sans pour autant avoir à créer un programme. Il le fait par l'intermédiaire d'un interprète de commandes (un « Shell » en anglais) muni d'une syntaxe et éventuellement programmable. Cet interprète peut accepter les lignes de commandes comme sous MS-DOS ou sous Unix. Il peut aussi gérer les « métaphores » graphiques comme avec les Macintoshes, Windows ou X-Window.

III.2 Historique d'Unix

III.2.1 Les origines

- **1969** : Bell Laboratories, centre de recherches commun à AT&T et Western Electrics, Ken Thompson travaille sur MULTICS (Multiplexed Information and Computing Service). Bell se retire du projet, Multics est abandonné. Ken Thompson décide de développer son propre OS, en s'éloignant volontairement de tout existant et écrit UNICS (Unified Information and Computing System) sur DEC PDP-7. Équipe : Dennis Ritchie, Rudd Canaday, puis Brian Kernighan.

- **1970** : Premier portage sur DEC PDP-11/20, avec le premier compilateur C, conçu spécialement pour rentre cet OS portable.
- **1971** : Version 1 d'Unix sur PDP/11-20 avec un système de fichiers, fork(), roff, ed, suite à la demande de AT&T qui avait besoin d'un système de traitement de textes pour l'aide à l'écriture de ses brevets.
- **1973** : La V2 intègre les tubes (pipes)
- **1974** : AT&T ne voyant pas d'avenir commercial à Unix, décide de distribuer le code source aux universités selon quatre critères de licence. Unix gagne donc la faveur des universitaires. Entre 1974 et 1977 les versions de la V3 à la V6 voient le jour.
- **1978** : La V7 est annoncée, développée afin de pouvoir être portée sur d'autres architectures matérielles. AT&T rattrape le coup et décide de distribuer les sources sous licence. Diffusion d'un manuel système et de développement par Brian Kernighan et Rob Pike. Apparition du Bourne Shell. Taille du noyau : 40Ko ! La V7 est la base commune à tous les Unix.
- **1979** : Le coût des licences Unix incite l'université de Californie à Berkeley à continuer ses travaux sur les sources diffusées avant la licence, et créée sa propre variante : BSD Unix. Le DARPA décide d'utiliser Unix pour ses développements, notamment BSD Unix.
- **1983** : AT&T met en vente la version commerciale de Unix SYSTEM V.
- **1986** : Première ébauche des normes POSIX sur la standardisation des appels systèmes et des fonctions.
- **1987** : Création de X-Window, interface C/S graphique développée au sein du MIT. System V v3, premiers Unix propriétaires de HP et IBM suite à la modification de la licence de SYSTEM V. BSD 4.3, Unification de BSD et SYSTEM V (Sun et AT&T), d'où abandon des particularités de chaque système.

- **1988** : Troisième version de X/Open Portability Guide, servant de référence pour tous les développements d'Unix ultérieurs (commandes, appels système, langages, requêtes, graphique, internationalisation, réseau).
- **1990** : System V v4 de AT&T, nouveaux standards d'unification avec Sun. Les autres constructeurs se sentent menacés et fondent OSF (Open Software Foundation).
- **1991** : OSF/1. Apparition des premiers clones Unix comme Linux et FreeBSD.
- **1992** : Sun sort Solaris (SunOS), dérivé de System V v4, avec la gestion des threads. AT&T créée USL (Unix Software Laboratories) et transfert toutes les licences à cettre société.
- **1993** : **Novell** rachète **USL**, puis transfert les droits de licences à **X/Open**.
- **Depuis 1993** : S'il existe un grand nombre d'Unix propriétaires, la plupart restent conformes aux normes et standards établis (X/Open, Posix). On distingue deux grandes branches SYSTEM V et BSD. Les deux sont compatibles. L'arrivée de Linux (dérivé de System V mais avec pas mal d'améliorations issues de BSD) a changé la donne.
- Les code source d'Unix appartient aujourd'hui à la société **Caldera** issue de Novell, mais les droits et la force de proposition sont transférés à l'**Open Group**.

III.2.2 Unix sur Micro

Le premier Unix disponible sur PC a été porté par Microsoft en 1979, **Xenix**, disponible jusqu'en 1984. Il fonctionnait sur 8086 à l'aide de très lourdes modifications. L'essor a eu lieu avec l'apparition du 80286, premier processeur Intel à posséder un mode protégé et des instructions de commutation de tâches. A l'arrivée du 80386 en 1987, la société Santa Cruz Operations qui diffusait dèja Xenix modifia Xenix et l'appela UNIX System V/386.

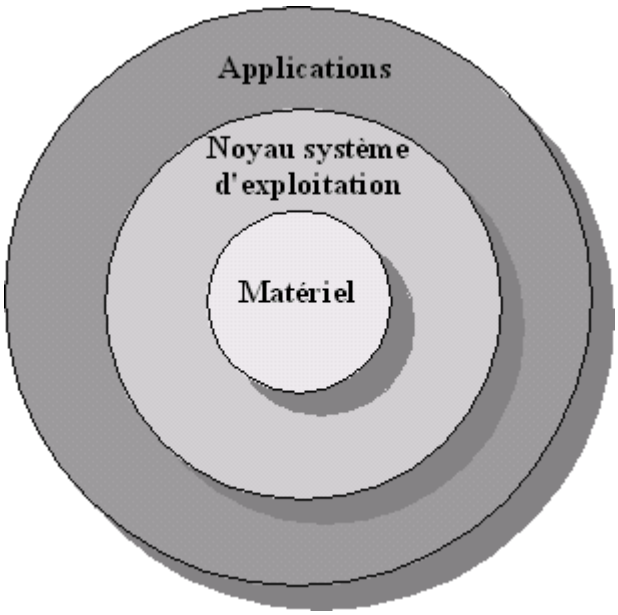
Depuis on trouve plusieurs portages Unix sur PC, dont les principaux sont SCO Openserver / Unixware, Sun Solaris, Linux, FreeBSD, OpenBSD, NetBSD.

Le PC n'est pas le seul micro supportant Unix : on trouve les Mac (anciens et nouveaux modèles), les Atari et Amiga, les machines à base de processeurs Alpha, Sur PC et dans les écoles, on utilise généralement Linux.

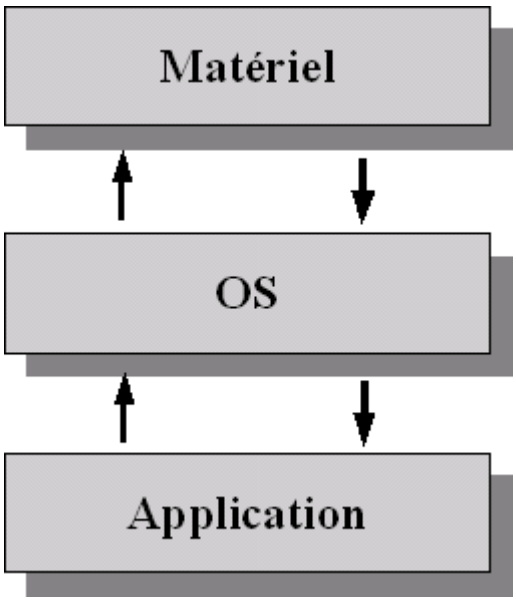
III.3 Architecture

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

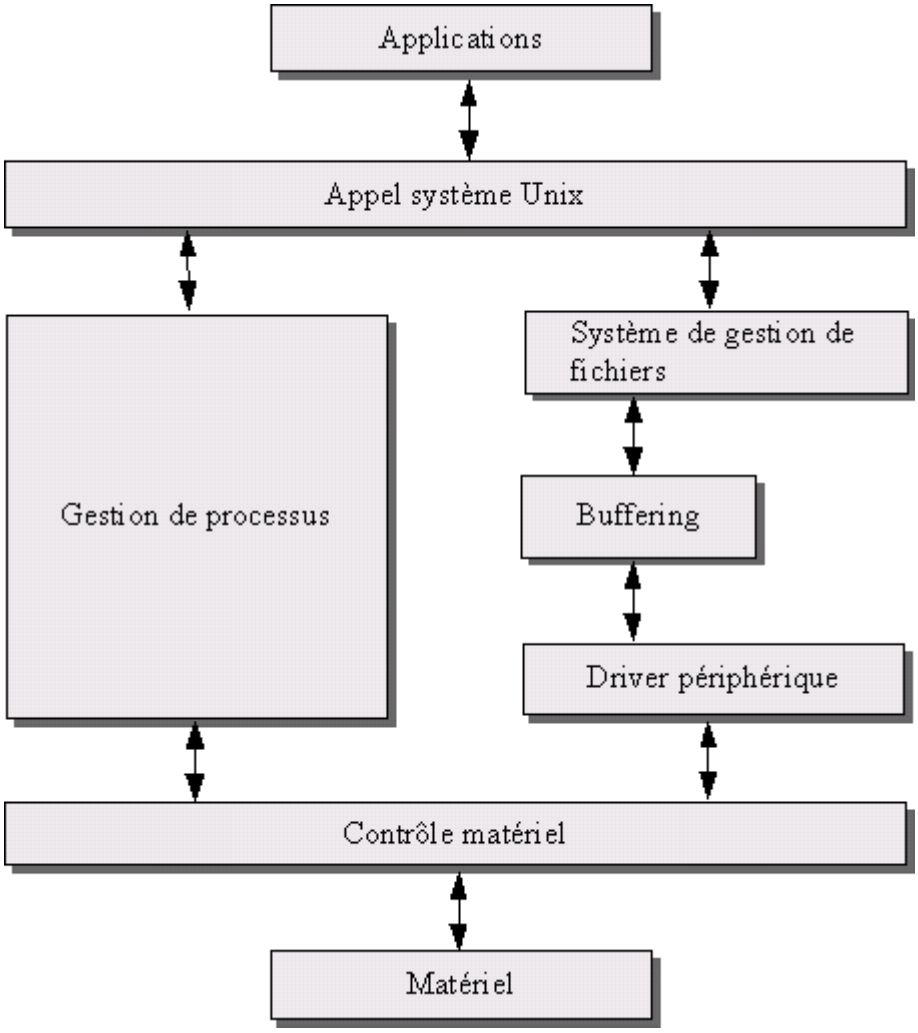
- le noyau Linux ;
- les programmes en ligne de commande et le shell ;
- le serveur XWindow ;
- le gestionnaire de fenêtres et le gestionnaire de bureau ;
- les applications XWindow.



Structure d'un SE UNIX



Rôle d'un OS



III.4 Fichiers

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

Chaque fichier possède un nom. Un nom de fichier sur une machine LINUX ou UNIX peut comporter jusqu'à 256 caractères.

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

Quel que soit le type de fichier, toutes les données contenues dans ce dernier consistent en une suite de nombres. La taille du fichier n'est que la longueur de la liste de ces nombres. Chaque nombre est appelé un octet (byte). Chaque octet contient 8 bits.

III.5 Les Commandes.

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

III.6 Connexion et changement de mot de passe.

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

Une invite de mot de passe (***password prompt***) s'affiche, derrière laquelle vous devez saisir votre mot de passe. Eventuellement, l'identifiant et le mot de passe peuvent être identiques. Notez que votre mot de passe n'apparaît pas à l'écran: à mesure que vous le tapez, il demeure invisible. Après cette saisie, enfoncez la touche ***"Entrée"*** ou ***"Return"*** à nouveau.

Login : <tapez ici votre nom d'utilisateur>

Password : <tapez ici votre mot de passe>

L'écran peut afficher un message et vous présenter à nouveau une autre invite de connexion. Cela signifie que vous avez tapé quelque chose d'incorrect. Faites un nouvel essai.

Le mot de passe n'apparaît pas en clair et doit être tapé en aveugle. En cas d'erreur, un message indiquera :

Login incorrect

Suivant la version d'Unix et la configuration, plusieurs lignes de message peuvent apparaître, qu'il est possible d'ignorer. Puis le prompt du Shell devrait apparaître, quelque chose du genre

user@machine$

ou

$

ou

%

III.7 Vos premiers pas sous Linux

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

Le premier exercice consistera à changer votre mot de passe. Tapez la commande ***passwd***. L'ordinateur vous demande le mot de passe qui vous est actuellement attribué. Si vous ne voulez pas poursuivre, faites ***Ctrl+C***. Le Shell vous rend une nouvelle invite. Introduisez ce mot de passe de manière à ce que le Shell vous invite à donner le nouveau mot de passe. Confirmez ce dernier. Notez que ce mot de passe doit comprendre des lettres, des nombres et certains signes de ponctuation. Vous verrez plus tard pourquoi il est important d'assurer une bonne sécurité sur le mot de passe. Le nouveau mot de passe prend cours immédiatement en remplaçant le mot précédent avec lequel vous vous êtes connecté.

$ passwd

Old password:

New password:

Reenter password:

La commande " ***passwd*** " peut aussi retourner des messages qu'éventuellement vous ne comprenez pas encore. Essayez de déterminer la connotation négative ou positive de ceux−ci.

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

Pour se familiariser avec la saisie de commandes, nous pouvons tester quelques programmes d'information :

- ***date : affiche la date et l'heure***
- ***who : liste des utilisateurs connectés (who am i : qui suis-je)***
- ***cal : affiche le calendrier (cal 12 1975)***
- ***man : mode d'emploi (man cal)***

Pour interrompre une commande on peut utiliser la combinaison ***Ctrl+C***. Pour lier les commandes, on sépare les commandes par le caractère « **;** »

who ; date

Pour se déconnecter, il suffit de taper

exit

On peut aussi utiliser la combinaison de touches ***Ctrl+D.***

Le système permet dans certains cas à un utilisateur connecté de changer de nom en cours de travail avec la commande su. Le mot de passe du nouvel utilisateur sera demandé.

su [-] utilisateur [-c commande]

Si **–** est précisé, l'environnement du nouvel utilisateur est chargé, et si -c est précisé les commandes qui suivent sont exécutées.

La commande ***logname*** affiche le nom de login de l'utilisateur, en principe toujours le nom utilisé lors de la première connexion.

$ logname

tariq

Une première commande : « echo »

En principe cette commande n'est pas utile tout de suite, mais la première chose que l'on apprend généralement avec un Shell ou un langage quelconque est d'afficher un message du genre « Hello, world ! ». la commande echo est une commande centrale du Shell : elle transmet tous ses paramètres sur écran (ou canal de sortie standard).

$ echo texte

Le texte est quelconque mais peut aussi admettre quelques caractères de formatage.

Caractère	***Effet***
\n	Saut de ligne (newline)
\b	Retour arrière (backslash)
\t	Tabulation
\c	Pas de retour à la ligne (carriage)
\\	Affiche \
\$	Affiche $
\valeur	Affiche le caractère spécial de code octal valeur

$ echo "Bonjour\nComment ça va ?\c"

III.8 Afficher des fichiers avec la commande ls

Derrière votre invite, tapez ***ls***. La commande **ls** est l'abréviation en deux lettres (comme c'est souvent le cas de nombreuses commandes UNIX) pour list. ls affiche tous vos fichiers courants et les répertoires qui sont des dossiers regroupant des fichiers. Vous pourriez trouver qu'il ne fait pas grand−chose. Alors, retournez dans le

Shell en appuyant sur la touche "***ENTREE***". Comme il n'y a pas encore beaucoup de fichiers, cela semble bien vide. Notez que, pour la plupart, les commandes UNIX ne retournent pas de messages. La commande ***passwd*** est une exception. Si vous aviez des fichiers, ceux−ci seraient disposés en lignes ou en colonnes sans autre renseignement.

III.9 Touches spéciales de la console.

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

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

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

III.10 Création de fichiers

Il y a différentes manières de créer un fichier. Tapez: ***cat > Fiche.lettre*** et saisissez quelques lignes de texte. Vous utiliserez ce fichier par après. La commande ***cat*** est utilisée pour écrire depuis le clavier dans le fichier ***Fiche.lettre***.

A la fin de la dernière ligne, pressez "***ENTRER***" et ensuite ***Ctrl−D*** pour clore la saisie. A présent, si vous exécutez **ls**, vous apercevrez ***Fiche.lettre*** parmi d'autres fichiers. Tapez ***cat Fiche.lettre*** sans > et vous verrez que la commande ***cat*** affiche le contenu du fichier donné en argument à la commande, vous permettant ainsi de consulter votre lettre.

Une autre façon pour créer un fichier consiste à utiliser la commande:

$ touch Fiche.lettre

Cette commande sert en général à changer la date de dernière modification d'un fichier. Si le fichier ***Fiche.lettre*** n'existe pas il sera créer.

III.11 Caractères permis pour les noms de fichiers.

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

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9 . _ −

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

III.12 Répertoires.

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

Par exemple, votre lettre pourrait être regroupées avec d'autres lettres dans un dossier idoine. Sur un ordinateur, un dossier est appelé un répertoire. Ceci est la troisième caractéristique commune des ordinateurs: tout fichier peut faire partie d'un répertoire.

Pour comprendre comment fonctionnent les répertoires, tapez la commande ***mkdir lettres***, le terme ***mkdir*** signifiant ***make directory***. A présent, tapez ***ls***. Ceci devrait vous retourner une liste avec aussi bien ***Fiche.lettre*** que lettres. Le ***fichier lettres*** ne contient encore rien sauf le nom d'un répertoire dans lequel d'autres fichiers pourront être placés.

Pour se déplacer dans **le répertoire lettres**, il faut taper la commande ***cd lettres*** où ***cd*** signifie ***change directory***. Etant donné que ce répertoire vient d'être créé, il ne contient pas de fichiers et cela peut se vérifier en tapant ***ls***.

Vous pouvez créer un fichier en exécutant la commande ***cat*** comme vous l'avez fait précédemment (essayez donc). Pour revenir dans le répertoire original, vous utiliserez la commande ***cd ..*** où **..** a la signification spéciale de vous faire sortir du répertoire courant. Tapez à nouveau ls pour vérifier que vous êtes remonté d'un cran dans la hiérarchie de vos répertoires.

Cependant, il est ennuyeux que vous ne puissiez voir la différence entre fichier et répertoire. La différenciation se fait en utilisant la commande **ls −l** où **−l** est une option signifiant format long.

Si vous appliquez cette commande vous observerez de nombreux détails qu'éventuellement vous ne comprenez pas encore. Les trois informations qui d'emblée attireront votre attention sont:

(i)	tout−à−fait à droite, le nom de fichier;
(ii)	la taille mentionnée dans la cinquième colonne (le nombre d'octets que contient le fichier) et
(iii)	le type de fichier, information se trouvant tout−à−fait à gauche. Parmi la chaîne de caractères, vous trouverez à l'extrême gauche soit **−** ou **d**. Un **−** signifie que vous avez à faire à un fichier tandis qu'un **d** montre qu'il s'agit d'un répertoire (**directory**). La commande ***ls −l Fiche.lettre*** affichera seulement le fichier **Fiche.lettre**; ceci est utile pour connaître la taille d'un fichier particulier.

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

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

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

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

SUPPORT DE COURS

CHAPITRE IV : SYSTEME ET GESTION DE FICHIERS

IV.1. Définition

Le système de fichiers **/ FileSystem / FS**: comment dont sont gérés et organisés les fichiers par le système d'exploitation. Le **FS** de Linux est hiérarchique.

IV.2. Les divers types de fichiers

On distingue principalement trois types de fichiers : ordinaires, catalogue, spéciaux.

IV.2.1. Fichiers ordinaires (ordinary files)

Ce sont soit des fichiers contenant du texte, soit des exécutables (ou binaires), soit des fichiers de données. Par défaut, rien ne permet de différencier les uns des autres, sauf à utiliser quelques options de certaines commandes (ls -F par exemple) ou la commande **file**.

```
$ file nom_fic
nom fic : 32 Bits ELF Executable Binary (stripped)
```

IV.2.2. Catalogues (les répertoires ou directory)

Les répertoires permettent d'organiser le disque dur en créant une hiérarchie. Un répertoire peut contenir des fichiers normaux, des fichiers spéciaux et d'autres répertoires, de manière récursive.

IV.2.3. Fichiers spéciaux

Ce sont le bien souvent des fichiers servant d'interface pour les divers périphériques. Ils peuvent s'utiliser, suivant le cas, comme des fichiers normaux. Un accès en lecture ou écriture sur ces fichiers est directement dirigé vers le périphérique (en passant par le pilote Linux associé s'il existe

IV.3. Nomenclature des fichiers

On ne peut pas donner n'importe quel nom à un fichier, il faut pour cela suivre quelques règles simples. Ces règles sont valables pour tous les types de fichiers.

Sur les anciens systèmes un nom de fichier ne peut pas dépasser 14 caractères. Sur les systèmes récents, on peut aller jusqu'à 255 caractères. Il est possible d'utiliser des extensions de fichiers mais cela ne modifie en rien le comportement du système (un exécutable n'a pas besoin d'une extension particulière).

Linux fait la distinction entre les minuscules et majuscules. Toto, TOTO, ToTo et toto sont des noms de fichiers différents.

La plupart des caractères (chiffres, lettres, majuscules, minuscules, certains signes, caractères accentués) sont acceptés, y compris l'espace (très déconseillé). Cependant quelques caractères sont à éviter :

& ; ( ) ~ <espace> \ | ` ? - (en début de nom)

Quelques noms valides :

Fichier1
Paie.txt

123traitement.sh
Paie_juin_2002.xls
8
...

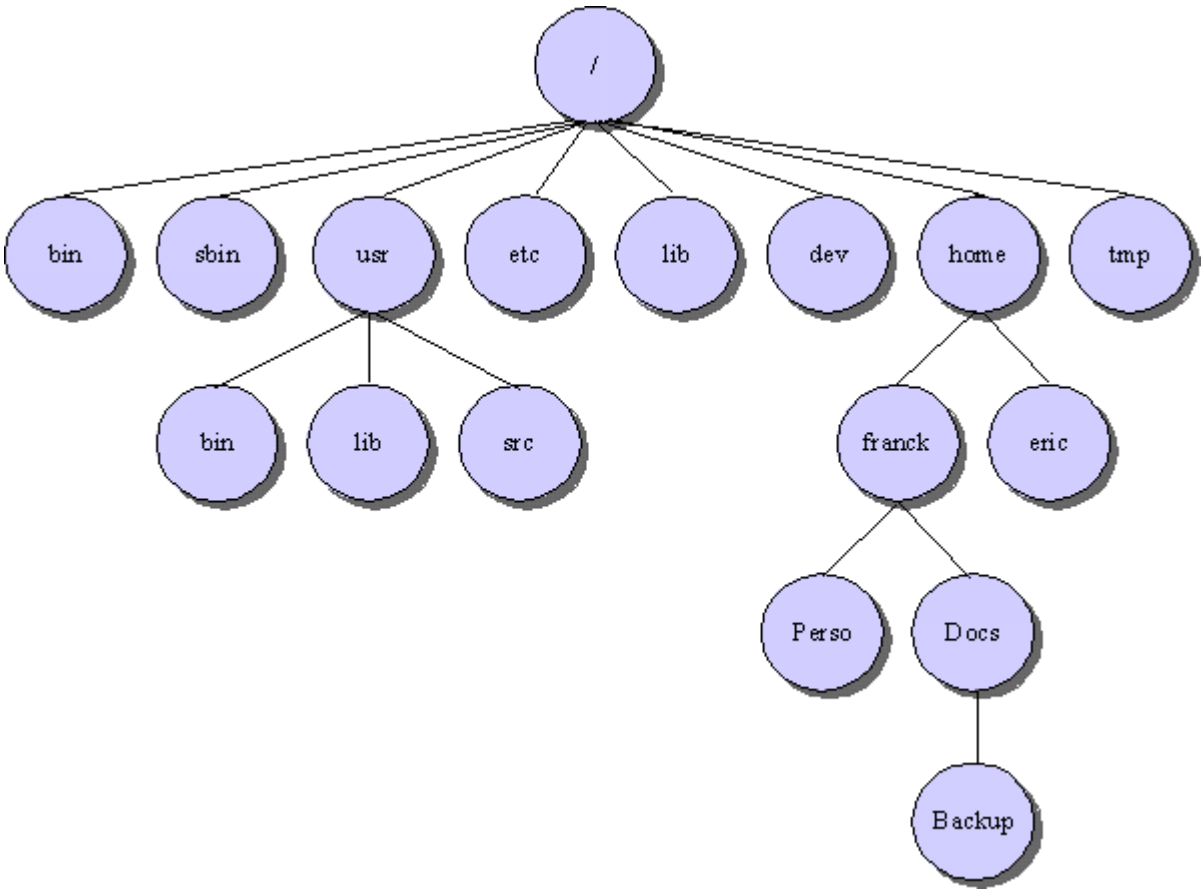
Quelques noms pouvant poser problème :

Fichier*
Paie(decembre)
Ben&Nuts
Paie juin 2002.xls
-f
...

IV.4. Chemins

IV.4.1. Structure et nom de chemin

Les chemins permettent de se déplacer dans le FileSystem. Un nom de fichier est ainsi généralement complété de son chemin d'accès. C'est ce qui fait que le fichier « toto » du répertoire « rep1 » est différent du fichier « toto » du répertoire « rep2 ». Le FS de Linux étant hiérarchique, il décrit une arborescence.



Le schéma précédent représente une arborescence d'un FS Linux. Le « ***/*** » situé tout en haut s'appelle la **racine** ou **root directory** (à ne pas confondre avec le répertoire de l'utilisateur root).

Le **nom de chemin** ou **path name** d'un fichier est la concaténation, depuis la racine, de tous les répertoires qu'il est nécessaire de traverser pour y accéder, chacun étant séparé par le caractère « / ». C'est un **chemin absolu**.

/home/toto/Docs/Backup/fic.bak

IV.4.2. Chemin relatif

Un nom de chemin peut aussi être relatif à sa position courante dans le répertoire.

Le système (ou le shell) mémorise la position actuelle d'un utilisateur dans le système de fichier, le répertoire actif. On peut donc accéder à un autre répertoire de l'arborescence depuis l'emplacement actuel sans taper le chemin complet.

Pour se déplacer dans les répertoires, on utilise la commande **cd**. Le « **..** » permet d'accéder au répertoire de niveau supérieur. Le « **.** » définit le répertoire actif (répertoire courant). La commande **ls** permet de lister le contenu du répertoire. La commande **pwd** (print working directory) affiche le chemin complet du répertoire actif.

```
$ cd /
$ ls
bin
sbin
usr
etc
lib
dev
home
tmp
$ cd /usr/lib
$ pwd
/usr/lib
$ cd ../bin
$ pwd
/usr/bin
$ cd ../../etc
$ pwd
/etc
```

IV.4.3. Répertoire personnel

Lors de la création d'un utilisateur, l'administrateur lui alloue un répertoire utilisateur. Après une connexion, l'utilisateur arrive directement dans ce répertoire, qui est son **répertoire personnel**. C'est dans ce répertoire que l'utilisateur pourra créer ses propres fichiers et répertoires. La commande **cd** sans argument permet de retourner directement dans son répertoire utilisateur.

Login : toto
Password :
$ pwd
/home/toto

IV.4.4. ls et quelques commandes intéressantes

La commande **ls** permet de lister le contenu d'un répertoire (catalogue) en lignes ou colonnes. Elle supporte plusieurs options.

Option	***Signification***
-l	Sortie de chaque information des fichiers
-F	Ajoute un « / » au nom d'un répertoire, un « * » au nom d'un exécutable, un « | » pour un tube nommé et « @ » pour un lien symbolique.
-a	Affiche toutes les entrées, y compris « . », « .. » et les fichiers cachés (qui commencent par un .)
-d	affiche le nom (et les attributs) des répertoires et pas leur contenu.

-i	Affiche les numéros d'inode.
-R	Mode récursif. Rentre dans les répertoires et affiche leur contenu.
-r	Inverse l'ordre du tri (à l'envers)
-t	Tri par date de modification
-c	Affiche la date de création (si -l) ou tri par date de création (si -t)
-C	Les noms sont affichés sur plusieurs colonnes
-u	Affiche la date d'accès (-l) ou tri par date d'accès (-t)
-1	Liste sur une seule colonne.

Sortie de la commande ls -l :

-rw-r--r--	1	oracle	dba	466	Feb 8 2001	input.log
1	2	3	4	5	6	7

1. Le premier caractère représente le type de fichier, les autres, par blocs de trois, les droits pour l'utilisateur, le groupe et tous (expliqué plus loin).
2. Compteur de liens (expliqué plus loin)
3. Propriétaire du fichier.
4. Groupe auquel appartient :le fichier
5. Taille du fichier en octets
6. Date de dernière modification (parfois avec l'heure)
7. Nom du fichier

Deux autres commandes utiles :

- **cat** : concaténation de fichiers, le résultat étant affiché par défaut sur la sortie standard (écran)
- **touch** : permet de créer un fichier s'il n'existe pas, et s'il existe de modifier sa date d'accès et sa date de modification, touch toto crée le fichier toto s'il n'existe pas

IV.5. Gestion des fichiers et répertoires

IV.5.1. Création de répertoires

La commande **mkdir** (make directory) permet de créer un ou plusieurs répertoires, ou une arborescence complète.

mkdir rep1 [rep2] ... [repn]

$ mkdir documents
$ mkdir documents/texte documents/calcul documents/images

La commande ***mkdir*** accepte un paramètre « -p » permettant de créer une arborescence. Dans l'exemple précédent, si je veux créer documents/texte et que documents n'existe pas, alors :

$ mkdir -p documents/texte

va créer à la fois documents et texte. C'est valable pour tous les répertoires de niveau supérieur :

$ mkdir -p documents/texte/perso

va créer les répertoires documents, texte et perso s'ils n'existent pas. S'il existent ils ne sont pas modifiés.

IV.5.2. Suppression de répertoires

La commande **rmdir** (remove directory) supprime un ou plusieurs répertoires. Elle ne supprime pas une arborescence. Si des fichiers sont encore présents dans le répertoire, la commande retourne une erreur. Le répertoire ne doit donc contenir ni fichiers ni répertoires.

rmdir rep1 [rep2] ... [repn]

$ cd documents
$ rmdir texte/perso

IV.5.3. Copie de fichiers

La commande **cp** (copy) copie un ou plusieurs fichiers vers un autre fichier ou vers un répertoire.

cp fic1 fic2
cp fic1 [fic2 ... ficn] rep1

Dans le premier cas, fic1 est recopié en fic2. Si fic2 existe, il est écrasé sans avertissement (sauf droit particulier). Dans le second cas, fic1, fic2 et ainsi de suite sont recopiés dans le répertoire rep1. Les chemins peuvent êtres absolus ou relatifs. La commande peut prendre les options suivantes :

- **-i** : demande une confirmation pour chaque fichier avant d'écraser un fichier existant.
- **-p** : préservation des permissions, dates d'accès de modification
- **-r** : Récursif. Si la source est un répertoire copie de ce répertoire et de toute son arborescence. Les liens symboliques (voir plus loin) ne sont

pas recopiés tels quels, mais seulement les fichiers pointés (avec le nom du lien cependant).

- **-R** : comme -r mais la recopie est identique à l'original (les liens symboliques sont copiés tels quels)

$ cd
$ pwd
/home/toto
$ touch cv.txt calc.xls image.jpg
$ ls
cv.txt calc.xls image.jpg documents
$ cp cv.txt documents/texte
$ cp calc.xls documents/calcul/calcul_paie.xls
$ cd documents
$ touch fichier
$ cp fichier ..
$ cp texte/cv.txt .

IV.5.4. Déplacer et renommer un fichier

La commande **mv** (move) permet de déplacer et/ou de renommer un fichier. Elle a la même syntaxe que la commande cp. On peut à la fois déplacer et changer de nom.

$ mv cv.txt cv_toto.txt
$ mv image.jpg documents/images/photo_toto_cv.jpg

IV.5.5. Supprimer un fichier ou une arborescence

La commande **rm** (remove) supprime un ou plusieurs fichiers, et éventuellement une arborescence complète, suivant les options. La suppression est définitive (à moins d'avoir un utilitaire système propre au filesystem).

rm [Options] fic1 [fic2...]

Options :

- **-i** : la commande demandera une confirmation pour chacun des fichiers à supprimer. Suivant la version d'Linux, le message change et la réponse aussi : y, Y, O, o, N, n, parfois toutes.
- **-r** : le paramètre suivant attendu est un répertoire. Dans ce cas, la suppression est récursive : tous les niveaux inférieurs sont supprimés, les répertoires comme les fichiers.
- **-f** : force la suppression. Si vous n'êtes pas le propriétaire du fichier à supprimer, rm demande une confirmation, mais pas avec l'option -f. Aucun message n'apparaîtra si la suppression n'a pu avoir lieu.

$ cd
$ rm -rf documents

IV.5.6. Les liens : plusieurs noms pour un fichier

Un lien permet de donner plusieurs noms à un même fichier, ou de faire pointer un fichier sur un autre. Plutôt que de faire plusieurs copies d'un même fichier pour plusieurs utilisateurs, on peut par exemple permettre à ceux-ci d'accéder à une copie unique, mais depuis des endroits et des noms différents. On utilise la commande **ln**.

ln [options] fic1 fic2

ln [options] fic1 rep1

ln [options] rep1 fic2

Il existe deux types de liens : les liens en dur « **hard links** » et les liens symboliques « **symbolic links** ».

IV.5.6.1. Hard link

Un **hard link** permet d'ajouter une référence sur un inode.

Sous Linux chaque fichier est en fait référencé au sein de deux tables : une table d'**inode** (information >node, noeud d'information, une par filesystem) qui contient outre un numéro de fichier, des informations comme des droits, le type et des pointeurs sur données, et une table catalogue (une par répertoire) qui est une table de correspondance entre les noms de fichiers et les numéros d'inodes. Le hard link rajoute donc une association dans cette seconde table entre un nom et un inode. Les droits du fichier ne sont pas modifiés.

Un hard link ne permet pas d'affecter plusieurs nom à un même répertoire, et ne permet pas d'effectuer des liens depuis ou vers un autre filesystem. De plus, faites attention au compteur de lien fourni par la commande ls -l : un 1 indique que ce fichier ne possède pas d'autres liens, autrement dit c'est le dernier. Si vous le supprimez, il est définitivement perdu. Par contre, tant que ce compteur est supérieur à 1, si un lien est supprimé, il reste une copie du fichier quelque part.

```
$ cd
$ touch fic1
$ ln fic1 fic2
```

```
$ ls
fic1 fic2
$ ls -l
-rw-r--r-- 2 oracle system 0 Jul 25 11:59 fic1
-rw-r--r-- 2 oracle system 0 Jul 25 11:59 fic2
$ ls -i
 484 fic1 484 fic2
```

L'exemple précédent montre que les hard links n'ont pas de type particulier et sont considérés comme des fichiers ordinaires. On constate que chacun a 2 liens. Logique puisque deux fichiers pointent sur le même inode. Enfin nous voyons bien en résultat du ls -i que fic1 et fic2 ont le même inode, à savoir 484.

IV.5.6.2. Symbolic link

Un lien symbolique ne rajoute pas une entrée dans la table catalogue mais est en fait une sorte d'alias, un fichier spécial contenant une donnée pointant vers un autre chemin (on peut le concevoir comme une sorte de fichier texte spécial contenant un lien vers un autre fichier ou répertoire).

De par cette nature, un lien symbolique ne possède pas les limitations du hard link. Il est donc possible d'effectuer des liens entre plusieurs **FileSystems,** et vers des répertoires. Le cas échéant le lien se comportera à l'identique du fichier ou du répertoire pointé (un **cd** nom_lien est possible dans le cas d'un répertoire).

La suppression de tous les liens symboliques n'entraîne que la suppression de ces liens, pas du fichier pointé. La suppression du fichier pointé n'entraîne pas la suppression des liens symboliques associés. Dans le cas le lien pointe dans le vide.

```
$ rm fic2
$ ln -s fic1 fic2
$ ls -l
-rw-r--r-- 1 oracle system 0 Jul 25 11:59 fic1
lrwxrwxrwx 1 oracle system 4 Jul 25 12:03 fic2 -> fic1
$ls -i
  484 fic1 635 fic2
$ ls -F
  fic1 fic2@
```

Cet exemple montre bien qu'un lien symbolique est en fait un fichier spécial de type « l » pointant vers un autre fichier. Attention, les droits indiqués sont ceux du fichier spécial. Lors de sont utilisation, ce sont les droits du fichier ou du dossiers pointés qui prennent le dessus. On distingue le caractère « @ » indiquant qu'il s'agit d'un lien symbolique. On remarque aussi que les inodes sont différents et que les compteurs sont tous à 1.

IV.5.7. Critères de recherche sur noms de fichier

Lors de l'utilisation de commandes en rapport avec le système de fichier, il peut devenir intéressant de filtrer la sortie de noms de fichiers à l'aide de certains critères, par exemple avec la commande ls. Au lieu d'afficher toute la liste des fichiers, on peut filtrer l'affichage à l'aide de divers critères et caractères spéciaux.

Caractère spécial	*Rôle*
*	Remplace une chaîne de longueur variable, même vide
?	Remplace un caractère unique quelconque

[]	Une série ou une plage de caractères
[!...]	Inversion de la recherche

Ainsi,

- **ls a*** : tous les fichiers commençant par a
- **ls a??** : tous les fichiers de trois caractères commençant par a
- **ls a??*** : tous les fichiers d'au moins trois caractères et commençant par a
- **ls [aA]*** : tous les fichiers començant par a ou A
- **ls [a-m]?*txt** : tous les fichiers commençant par les lettres de a à m, possédant au moins un second caractère avant la terminaison txt.

C'est le shell qui est chargé d'effectuer la substitution de ces caractères avant le passage des paramètres à une commande. Ainsi lors d'un

cp * documents

cp ne reçoit pas le caractère * mais la liste de tous les fichiers et répertoires du répertoire actif.

IV.5.8. Verrouillage de caractères

Certains caractères spéciaux doivent être verrouillés, par exemple en cas de caractères peu courants dans un nom de fichier.

Le **backslash ** permet de verrouiller un caractère unique. ***ls paie\ *.xls*** va lister tous les fichiers contenant un espace après paie.

Les **guillemets "..."** les guillemets permettent l'interprétation des caractères spéciaux, variables, au sein d'une chaîne.

Les **apostrophes '...'** verrouillent tous les caractères spéciaux dans une chaîne ou un fichier.

SUPPORT DE COURS

CHAPITRE V : LES COMMANDES DE BASE, L'EDITEUR VI ET LES REDIRECTIONS

V.1 Les commandes de base.

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

V.1.1 La commande ls, fichiers cachés, options des commandes.

En plus des fichiers textes ordinaires et des répertoires, il existe d'autres types de fichiers, quoique tous contiennent une liste d'octets. Un fichier caché est un fichier qui n'apparaît pas lorsque la commande **ls** est exécutée pour afficher le contenu d'un répertoire. Pour voir un fichier caché, il faut que vous exécutiez la commande **ls −a** où **−a** est une option pour désigner tous les fichiers (all files). Une autre variante est **ls −l** qui affiche le contenu d'un répertoire dans le format long. Le tiret ``**−**'' permet d'utiliser des variantes d'une commande, appelées options ou arguments de commandes. La plupart des commandes UNIX/Linux admettent diverses options. Elles peuvent être combinées afin d'apporter de la concision : par exemple, ls −a −l, ls −l −a ou ls −al produisent une liste longue de tous les fichiers, y compris cachés.

Les commandes sous la licence des logiciels libres GNU sont, à ce point de vue, meilleures: elles disposent d'un plus grand nombre d'options que les commandes traditionnelles UNIX et apportent davantage de souplesse.

Toutes les commandes GNU prennent −h ou −help comme arguments. Vous pouvez saisir une commande suivie de cet argument afin d'obtenir un résumé de l'usage de cette commande. Ceci consitue une aide brève des options que

vous pourriez avoir oubliées si vous êtes déjà familier avec la commande (il ne s'agit pas d'une description exhaustive).

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

V.1.2 Messages d'erreurs.

Bien que la plupart des commandes ne renvoient pas de messages lors d'une exécution réussie, les commandes devraient afficher d'éventuels messages d'erreur de manière cohérente. Le format varie d'une commande à l'autre mais souvent, il apparaît ainsi:

nom_de_commande : what was attempted : error message.

Par exemple, la commande **ls −l azerty** revoie une erreur ***ls: azerty: No such file or directory.***

- Que se passe−t−il réellement quand la commande **ls** tente de lire le fichier **azerty?** Vu que le fichier n'existe pas, un code d'erreur 2 survient. Ce code d'erreur correspond à un cas pour lequel un fichier ou un répertoire n'est pas trouvé.
- Ce code d'erreur est automatiquement traduit en No such file or directory. Il est important de comprendre la différence entre un message d'explication qu'une commande retourne et un code d'erreur traduit sous forme de texte. Le fait est qu'il y a beaucoup de raisons à l'obtention d'un code d'erreur identique (il existe une centaine de codes d'erreur). L'expérience vous montrera que les messages d'erreur ne vous disent pas ce que vous faites, mais seulement ce qui va mal.
- Le fichier ***/usr/include/asm/errno.h*** contient une liste complète des erreurs fondamentales. Par ailleurs, plusieurs autres fichiers d'en−tête (qui se terminent par un .h) peuvent définir leurs propres erreurs de code. Sous UNIX, cependant, cela constitue 99% des erreurs

susceptibles d'être produites. Pour l'heure, la plupart d'entre elles n'ont pas beaucoup de signification mais elles sont répertoriées dans le tableau 6 à titre de référence.

V.1.3 Les pages du manuel.

La commande ***man [<section> | −a] <commande>*** fournit une aide sur un sujet particulier et, son nom est une contraction pour manuel. Chaque commande du système est documentée dans les pages de man.

Par exemple: man 1 −a cp affiche les explications sur l'usage de la commande cp.

Au cours des dernières années est apparu un nouveau format de documentation: les pages d'info. Celles−ci sont considérées comme un moyen moderne de description des commandes. Toutefois, la majorité de la documentation du système est seulement disponible, à l'aide de man.

Les pages de man constituent une référence faisant autorité quant au fonctionnement des commandes parce que ces pages sont écrites par les développeurs des commandes, eux−mêmes.

Sous UNIX, toute documentation imprimée sera considérée comme de "deuxième main". En revanche, les pages de man ne contiennent souvent pas les notions de base nécessaires pour comprendre le contexte d'utilisation d'une commande. Donc, il n'est pas possible à une personne d'apprendre UNIX à partir des seules pages de man. Cependant, une fois que vous avez acquis les bases nécessaires, les pages du manuel deviennent une indispensable source d'informat

V.1.4 Les pages d'info.

Les pages d'info contiennent d'excellentes références et de l'information tutorielle selon un format de type hypertexte. Saisissez info tel quel pour obtenir le menu général d'info. Vous pouvez également taper

info <commande> pour obtenir de l'aide à propos des nombreuses commandes de base. Cependant, certains paquets logiciels n'ont pas de pages d'info et d'autres systèmes UNIX ne contiennent pas de pages d'info du tout.

info est un programme interactif avec des touches de navigation et de la documentation d'aide.

V.1.5 Commandes fondamentales.

Vous pouvez vous entraîner à utiliser les commandes suivantes:

- ***bc*** : est un programme de calcul manipulant des nombres à grande précision. Il est utile pour réaliser tout type de calcul en ligne de commande. Son utilisation vous est laissée à titre d'exercice.
- ***cal [[0−12] 1−9999]*** : retourne un calendrier agréablement formaté, du mois courant, d'un mois sélectionné, ou des mois d'une année donnée. Essayez cal 1 pour le plaisir et cal 9 1752, année où le Pape a effacé quelques jours pour compenser une erreur d'arrondi.
- ***cat <nom de fichier> [<nom de fichier> ...]*** : écrit le contenu des tous les fichiers affichés à l'écran. cat peut regrouper des fichiers lorsque l'expression suivante est utilisée:
- ***cat <fichier1> <fichier2> ... > <nouveau_fichier>***. Le fichier intitulé <nouveau_fichier> contiendra le résultat d'une concaténation (c'est−à−dire une mise bout−à−bout) du contenu des fichiers donnés en argument.
- ***clear :*** nettoye le texte du terminal courant.
- ***date :*** retourne à l'écran la date et l'heure. (La commande time exécute un programme entièrement différent).
- ***df*** : correspond à disk free et vous indique la quantité d'espace libre sur votre système. L'espace disponible est usuellement exprimé en

kilooctets (1024 octets) (bien que sur certains systèmes UNIX les unités sont de 512 ou 2048 octets). La colonne la plus à droite mentionne les répertoires avec leur espace libre correspondant, dans la colonne juste à côté.

- ***du <directory>*** : correspond à disk usage et renvoie la quantité d'espace occupé par un répertoire. Cette commande est appliquée récursivement dans tous les sous−répertoires du répertoire désigné en argument.
- ***du −s <répertoire>*** peut ne renvoyer qu'un résumé. Testez aussi:
- ***du −−max−depth=1 /va***r et du ***−x /*** sur un système avec ***/usr*** et ***/home*** sur des partitions séparées.
- ***dmesg*** : renvoie à l'écran le journal des messages affichés durant la phase d'initialisation du système. Pour l'instant, ces messages ne vous sont peut−être pas encore explicites.
- ***echo*** : affiche un message au terminal. Essayez ***echo 'bonjour', echo $[10*3+2], echo '$[10*3+2]'.*** La commande ***echo −e*** permet l'interprétation de certaines séquences incluant un backslash. Par exemple, ***echo −e "\a"*** affiche une cloche ou fait émettre un son via votre terminal. echo −n fait la même chose mais sans pratiquer de saut de ligne. En fait, elle ne provoque pas de saut entre la ligne de saisie et l'affichage de votre invite. ***echo −e −n "\b"*** applique un retour en arrière sur une lettre (back−space), ce qui écrase le dernier caractère imprimé.
- ***exit*** : permet la déconnexion de la session en mode console et ferme un terminal en mode graphique tout en vous laissant dans votre session.
- ***expr <expression>*** : calcule l'expression numérique expression. La plupart des opérations arithmétiques que vous connaissez seront

effectuées. Essayez ***expr 5 + 10 '*' 2***. Notez les blancs entre les paramètres et la préséance des opérations (* est exécuté avant +).

- ***file <fichier>*** : affiche le type de données contenues dans un fichier. La commande ***file portrait.jpg*** vous dira que c'est une donnée image JPEG, standard JFIF. La commande file détecte une quantité très importante de types de fichier, sur chaque plate−forme. file fonctionne en vérifiant que les premiers octets d'un fichier correspondent aux séquences d'octets appelés nombres magiques. La liste complète de ces nombres est stockée dans **/usr/share/magic**.

Le terme ``nombre magique'' sous UNIX désigne normalement les séquences d'octets ou de nombres qui ont une signification déterminée. Ces nombres magiques sont créés pour le code source, les formats de fichiers et le système de fichiers.

- ***free :*** renvoie à l'écran la mémoire libre disponible. Vous noterez deux listes: l'espace de swap et la mémoire physique. Elles sont contiguës pour l'utilisateur. L'espace de swap est une extension de la mémoire, extension installée sur le disque. Bien sûr, son accès est lent par rapport celui de la mémoire vive, mais elle fournit l'illusion de plus de mémoire RAM disponible. Elle évite la possibilité d'un dépassement de mémoire RAM (ce qui est ``fatal'').
- ***head [−n <lignes>] <fichier>*** : affiche la liste des n premières lignes d'un fichier ou les 10 premières lignes par défaut si l'option −n n'est pas renseignée (voir aussi la commande ***tail***).
- ***hostname [<nouveau_nom>]*** : sans options, **hostname** affiche le nom de votre machine. Autrement, elle remplace le nom de machine par ***<nouveau_nom>.***
- ***more*** : est un afficheur (pager en anglais) qui formate à l'écran un fichier par page entière (sans défilement). Exécutez la commande ***ls −l***

/bin > bin−ls et, ensuite ***more bin−ls***. La première commande crée le fichier ***bin−ls*** qui contient le résultat de ***ls*** exécutée sur le répertoire ***/bin***. Ceci donnera une liste suffisamment longue pour ne pas tenir sur un seul écran car ***/bin*** contient de nombreuses entrées. La seconde commande permet de visualiser le fichier ***bin−ls***. Utilisez la barre d'espacement pour passer d'une page à l'autre. Pour sortir, pressez ***q***. Vous pouvez aussi tester ***ls −l /bin | more*** qui réalisera la même opération, mais en une seule commande composée.

- ***less*** : est la version GNU de more, mais avec des caractéristiques spéciales. Sur votre système les deux commandes sont peut−être identiques. Avec **less**, vous pouvez utiliser les touches fléchées pour naviguer dans les pages. Vous pouvez également faire des recherches.
- ***sort <fichier>*** : affiche le contenu d'un fichier avec les lignes éditées par ordre alphabétique. Créez un fichier appelé telephone, chaque ligne contenant une courte entrée d'un annuaire téléphonique. Ensuite tapez ***sort telephone***, ou ***sort telephone | less*** et constatez ce qui se produit. Sort présente beaucoup d'options intéressantes comme effectuer un affichage en ordre inverse (***sort −r***), éliminer des entrées multiples (***sort −u***), ignorer les espaces blancs en début de lignes (sort −b), etc..
- **split ...** : divise un fichier en plusieurs fichiers séparés. Cette commande peut être utilisée lorsqu'un fichier est trop long pour être copié directement sur une disquette. Il est alors nécessaire de le morceler en fichiers d'1 Mo, par exemple. Sa commande soeur ***csplit*** peut diviser un fichier selon des lignes spécifiques du texte. Ces commandes sont parfois utilisées telles quelles, mais aussi, le plus souvent, par des programmes qui manipulent des textes.

- **tac <fichier> [<fichier> ...]** : écrit le contenu de tous les fichiers, à l'écran en renversant l'ordre des lignes −c'est−à−dire en affichant la dernière ligne en premier. ***tac*** est ***cat*** écrit à l'envers et se comporte comme tel.
- **tail [−f] [−n <lignes>] <fichier>** : affiche les <lignes> lignes d'un fichier ou les 10 dernières lignes par défaut si l'option /−n n'est pas donnée. L'option **−f** signifie "surveiller le fichier pour les lignes qui ont été ajoutées à la fin du fichier" (voir aussi head ci−dessus),
- ***uname*** : imprime le nom du système d'exploitation UNIX utilisé. Dans ce cas: LINUX.
- ***uniq <fichier>*** : imprime un fichier de manière à ce que les lignes multiples soient supprimées. Le fichier doit d'abord être trié.
- ***wc [−c] [−w] [−l] <fichier>*** : compte le nombre d'octets (avec −c pour caractère), ou de mots (avec −w) ou de lignes (avec −l) dans un fichier,
- ***whatis <commande>*** : donne la première ligne de la page de man correspondant à <commande>, sauf si la page n'existe pas; dans ce cas, le message affiché est nothing appropriate,
- ***whoami*** : retourne à l'écran votre identifiant.

V.1.6 Recherche de fichiers.

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

```
$ find /usr −type f −exec ls '−al' '{}' ``;"
```

La commande ***find*** présente l'inconvénient de lire de manière active les répertoires pour y trouver les fichiers. Ce processus est lent, notamment lorsque vous commencez depuis le répertoire racine. L

locate <fichier> est une commande alternative. Celle−ci recherche les fichiers dans une base de données établie au préalable. Donc, la recherche est rapide. En contre−partie, il est nécessaire d'utiliser ***updatedb*** afin de mettre à jour la base de données des fichiers utilisés par ***locate***.

Sur certains systèmes, ***updatedb*** est exécuté automatiquement tous les jours à 4H00.

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

```
$ updatedb
$ locate rpm
$ locate HOWTO
```

V.1.7 Recherche dans les fichiers.

Très souvent, vous devrez chercher dans un certain nombre de fichiers afin d'y trouver un mot ou une partie de phrase donnée. Cela pourrait être le cas de fichiers contenant une liste de numéros de téléphone avec des noms de personnes et des adresses.

La commande ***grep*** effectue une recherche **ligne−par−ligne** dans un fichier et affiche les lignes qui contiennent un mot que vous lui avez indiqué. La syntaxe de grep se présente ainsi:

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

Les mots "mot", "chaîne", "motif" (word, string, pattern en anglais) sont utilisés comme des synonymes dans ce contexte. Ils signifient une courte

énumération de lettres et/ou de nombres pour lesquels vous essayez de trouver une correspondance. Un motif peut également désigner une chaîne de caractères avec des motifs de remplacement.

Exécutez la commande ***grep*** avec pour motif le mot "le" de manière à retourner à l'écran les lignes contenant ce motif "le":

```
$ grep 'le' Fiche.lettre.
```

A présent, essayez

```
$ grep 'le' *.lettre.
```

Voyons quelques options:

- grep −n <motif> <fichier> : montre le nombre de lignes pour lequel le motif a été trouvé dans le fichier.
- grep −<nombre> <motif> <fichier> : affiche le <nombre> de lignes qui précèdent et suivent chaque ligne où le motif a été détecté dans le fichier.
- grep −A <nombre> <motif> <fichier> : affiche le <nombre> de lignes qui suivent chaque ligne où le motif a été détecté dans le fichier.
- grep −B <nombre> <motif> <fichier> : affiche le <nombre> de lignes qui précèdent chaque ligne où le motif a été détecté dans le fichier.
- grep −v <motif> <fichier> : imprime seulement les lignes qui ne contiennent pas le motif indiqué comme argument. [Vous pouvez vous dire que l'option −v ne fait plus le même type de recherche que grep est sensé faire, c'est−à−dire rechercher des chaînes de caractères. En fait, les commandes LINUX possèdent une telle versatilité dans leur fonctionnalité qu'elles recouvrent d'autres commandes. Avec LINUX, on n'arrête jamais d'apprendre de nouvelles et astucieuses méthodes

pour réaliser des choses cachées dans les coins obscurs des pages de man].

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

V.1.8 Le PATH où les commandes sont recherchées.

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

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

```
$ echo $PATH
```

Vous verrez une ligne de plusieurs répertoires séparés par des caractères" **:** " (double−point).

/bin:/usr/bin:/usr/local/bin

Notez que le répertoire courant "**.**", n'est pas affiché. Il est fondamental que le répertoire courant ne soit pas dans le PATH pour des raisons de sécurité. Ceci fait que pour exécuter une commande dans le répertoire courant, il est nécessaire de faire:

```
$./<commande>
```

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

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

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

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

Il y a une commande supplémentaire, ***which,*** permettant de vérifier qu'une commande est localisable via le **PATH**. Parfois, il y a deux commandes de même nom dans différents répertoires contenus dans le PATH. L'exécution de ***which <commande>*** à l'invite permet de localiser la commande que votre Shell exécutera. Essayez :

```
$ which ls
$ which cp mv rm
$ which which
```

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

5.1.9 L'option −.−

S'il advient qu'un nom de fichier débute par −, alors il sera impossible de l'utiliser en tant qu'argument d'une commande. Pour contourner cet problème, la plupart des commandes présentent l'option −−. Cette option indique à la

commande qu'il n'y a pas d'autre option qui suit. Tout autre indication sera traité comme un nom de fichier littéral. Par exemple:

```
$ touch -- -nom_fichier
$rm -- -nom_fichier
```

V.2 L'éditeur vi

L'éditeur Linux par défaut se nomme **vi** (visual editor). S'il n'est pas des plus ergonomiques par rapport à des éditeurs en mode graphique, il a l'avantage d'être disponible et d'utiliser la même syntaxe de base sur tous les Unix. Chaque Unix propose généralement une syntaxe étendue au-delà de la syntaxe de base. Pour en connaître les détails : ***man vi.***

```
$ vi [options] Fichier [Fichier2 ...]
```

Trois modes de fonctionnement :

- **mode commande** : les saisies représentent des commandes. On y accède en appuyant sur « ***Echap*** ».
- **mode saisie** : saisie de texte classique
- **mode ligne de commande « à la ex »** : utilisation de commandes spéciales saisies et se terminant par Entrée. Accès pas la touche « : ».

V.2.1 Commandes de saisie

En mode commande

Commande	***Action***
A	Ajout de texte derrière le caractère actif

A	Ajout de texte en fin de ligne
i	Insertion de texte devant le caractère actif
I	Insertion de texte en début de ligne
O	Insertion d'une nouvelle ligne sous la ligne active
O	Insertion d'une nouvelle ligne au-dessus de la ligne active

V.2.2 Quitter

- La commande **ZZ** quitte et sauve le fichier
- **:q!** quitte sans sauver
- **:q** quitte si le fichier n'a pas été modifié
- **:w** sauve le fichier
- **:wq** ou **x** sauve et quitte

V.2.3 Déplacement en mode commande

Commande	***Action***
H	Vers la gauche
i	Vers la droite
K	Vers le haut
j	Vers le bas
0 (zéro)	Début de ligne (:0 première ligne)
$	Fin de ligne (:$ dernière ligne)

W	Mot suivant
B	Mot précédent
Fc	Saut sur le caractère 'c'
Ctrl + F	Remonte d'un écran
Ctrl + B	Descend d'un écran
G	Dernière ligne du fichier
NG	Saute à la ligne 'n' (:n identique)

V.2.4 Correction

Commande	***Action***
X	Efface le caractère sous le curseur
X	Efface le caractère devant le curseur
rc	Remplace le caractère sous le curseur par le caractère 'c'
dw	Efface le mot depuis le curseur jusqu'à la fin du mot
D$ (ou D)	Efface tous les caractères jusqu'à la fin de la ligne
dO	Efface tous les caractères jusqu'au début de la ligne.
dfc	Efface tous les caractères de la ligne jusqu'au caractère 'c'
dG	Efface tous les caractères jusqu'à la dernière ligne, ainsi que la ligne active
D1G	Efface tous les caractères jusqu'à la première ligne, ainsi que la

	ligne active
dd	Efface la ligne active

Ces commandes peuvent être répétées. 5Dd supprime 5 lignes.

On peut annuler la dernière modification avec la commande « u ».

V.2.5 Recherche dans le texte

Contrairement à un éditeur de texte classique, **vi** peut rechercher autre chose que des mots simples et fonctionne à l'aide de caractères spéciaux et de critères. La commande de recherche est le caractère « **/** ». La recherche démarre du caractère courant à la fin du fichier. Le caractère « **?** » effectue la recherche en sens inverse. On indique ensuite le critère, puis ***Entrée***.

/echo

recherche la chaîne 'echo' dans la suite du fichier. Quand la chaîne est trouvée, le curseur s'arrête sur le premier caractère de cette chaîne.

La commande « **n** » permet de continuer la recherche dans le sens indiqué au début. La commande « **N** » effectue la recherche en sens inverse.

V.2.5.1Quelques critères :

- /[FfBb]oule : Foule, foule, Boule, boule
- /[A-Z]e : Tout ce qui commence par une majuscule avec un e en deuxième position.
- /[A-Za-Z0-9] : tout ce qui commence par une majuscule,minuscule ou un chiffre

- /[^a-z] : plage négative : tout ce qui ne commence pas par une minuscule
- /vé.o : le point remplace un caractère, vélo, véto, véro, ...
- /Au*o : l'étoile est un caractère de répétition, de 0 à n caractères, Auo, Auto, Automoto, ...
- /.* : l'étoile devant le point, une chaîne quelconque de taille variable
- /[A-Z][A-Z]* : répétition du motif entre [] de 0 à n fois, recherche d'un mot comportant au moins une majuscule (en début de mot)
- /^Auto : le ^ indique que la chaîne recherchée devra être en début de ligne
- /Auto$: le $ indique que la chaîne recherchée devra être en fin de ligne

V.2.5.2 Quelques commandes de remplacement

Pour remplacer du texte, il faut se placer au début de la chaîne à modifier, puis taper l'une des commandes suivantes.

Commande	***Action***
cw	Remplacement du mot courant
C$	Remplacement jusqu'à la fin de la ligne
cO	Remplacement jusqu'au début de la ligne
cfx	Remplacement jusqu'au prochain caractère 'x' dans la ligne courante
c/Auto	Remplacement jusqu'à la prochaîne occurrence de la chaîne

(Entrée)	'Auto'

Après cette saisie, le caractère $ apparaît en fin de zone à modifier. Il suffit alors de taper son texte et d'appuyer sur Echap.

V.2.6 Copier-Coller

On utilise la commande « **y** » (Yank) pour copier du texte, elle-même devant être combinée avec d'autres indications. Pour couper (déplacer), c'est la commande « **d** ». Pour coller le texte à l'endroit choisi, c'est la commande « **p** » (derrière le caractère) ou « **P** » (devant le caractère). Si c'est une ligne complète qui a été copiée, elle sera placée en-dessous de la ligne active.

Pour copier une ligne : yy

Pour copier cinq lignes : 5yy

Pour placer les lignes copiées à un endroit donné : p

L'éditeur vi dispose de 26 tampons pour y stocker les données que l'on peut nommer comme on le souhaite. On utilise pour ça le « " ».

Pour copier cinq mots dans la mémoire m1 : m1y5w

Pour coller le contenu de la mémoire m1 à un endroit donnée : "m1p

V.2.7 Substitution

La substitution permet de remplacer automatiquement plusieurs occurrences par une autre chaîne.

:[1ere ligne, dernière ligne]s/Modèle/Remplacement/[gc]

Les numéros de lignes sont optionnels. Dans ce cas la substitution ne se fait que sur la ligne courante. En remplacement des numéros de lignes, « **.** » détermine la ligne courante, « **1** » la première ligne, « **$** » la dernière ligne.

Le modèle est l'un des modèles présenté plus tôt. Remplacement est une chaîne quelconque qui remplacera le modèle.

Par défaut seule la première occurrence est remplacée. La lettre « **g** » indique qu'il faut remplacer toutes les occurrences. Avec « **c** », vi demande une confirmation pour chacune des occurrences.

:1,$s/[Uu]nix/UNIX/g

Cet exemple remplace, dans tout le fichier, Unix ou unix par UNIX.

V.2.8 Autres en ligne de commande

Commande	***Action***
:w Nom_fic	Sauve le fichier sous Nom_fic, en l'écrasant ou en le créant
:1,10w Nom_fic	Sauve les lignes 1 à 10 dans Nom_fic
:r Nom_fic	Insère le fichier Nom_fic à partir de la ligne courante
:! commande	Exécute la commande puis retourne à l'éditeur
:r! commande	Exécute la commande et insère le résultat à partir de la ligne courante
:f Nom_fic	Affiche en bas d'écran le nom du fichier, le nombre de ligne et la position actuelle

:e Nom_fic	Le fichier est chargé. Un message indique si le précédent a été modifié
:e #	Le dernier fichier chargé est affiché. Permet de commuter entre les fichiers

V.3 Redirections

Les redirections sont l'une des plus importantes possibilités offerte par le Shell.

Par redirection, on entend la possibilité de rediriger l'affichage de l'écran vers un fichier, une imprimante ou tout autre périphérique, les messages d'erreurs vers un autre fichier, remplacer la saisie clavier par le contenu d'un fichier.

Linux utilise des canaux d'entrées/sorties pour lire et écrire ses données. Par défaut le canal d'entrée est le clavier, et le canal de sortie, l'écran. Un troisième canal, le canal d'erreur, est aussi redirigé vers l'écran.

Il est donc possible de rediriger ces canaux vers des fichiers, ou du flux texte de manière transparente pour les commandes Linux.

V.3.1. En sortie

On se sert du caractère « **>** » pour rediriger la sortie standard (celle qui va normalement sur écran). On indique ensuite le nom du fichier où seront placés les résultats de sortie.

```
$ ls -l > resultat.txt
$ cat resultat.txt
total 1
```

```
-rw-r--r-- 1 Administ ssh_user 0 Jul 4 12:04 TOTO
-rw-r--r-- 1 Administ ssh_user 0 Jul 25 15:13 resultat.txt
-rw-r--r-- 1 Administ ssh_user 171 Jul 25 15:13 test.txt
```

Si le fichier n'existe pas, il sera créé. S'il existe, son contenu sera écrasé, même si la commande tapée est incorrecte.

Le shell commence d'abord par créer le fichier puis exécute ensuite la commande.

Pour rajouter des données à la suite du fichier, donc sans l'écraser, on utilise la double redirection « **>>** ». Le résultat est ajouté à la fin du fichier.

```
$ ls -l > resultat.txt
$ date >> resultat.txt
$ cat resultat.txt
total 1
-rw-r--r-- 1 Administ ssh_user 0 Jul 4 12:04 TOTO
-rw-r--r-- 1 Administ ssh_user 0 Jul 25 15:13 resultat.txt
-rw-r--r-- 1 Administ ssh_user 171 Jul 25 15:13 test.txt
Thu Jul 25 15:20:12 2002
```

V.3.2. En entrée

Les commandes qui attendent des données ou des paramètres depuis le clavier peuvent aussi en recevoir depuis un fichier, à l'aide du caractère « **<** ». Un exemple avec la commande « **wc** » (word count) qui permet de compter le nombre de lignes, de mots et de caractères d'un fichier.

```
$ wc < resultat.txt
4 29 203
```

V.3.3. Les canaux standards

On peut considérer un canal comme un fichier, qui possède sont propre descripteur par défaut, et dans lequel on peut ou lire ou écrire.

- Le canal d'entrée standard se nomme « **stdin »** et porte le descripteur **0**.
- Le canal de sortie standard se nomme « **stdout »** et porte le descripteur **1**.
- La canal d'erreur standard se nomme « **stderr »** et porte le descripteur **2**.

On peut rediriger les canaux de sortie 1 et 2 vers un autre fichier.

```
$ rmdir dossier2
rmdir: `dossier2': No such file or directory
$ rmdir dossier2 2>error.log
$ cat error.log
rmdir: `dossier2': No such file or directory
```

On peut aussi rediriger les deux canaux de sortie dans un seul et même fichier, en les liant. On utilise pour cela le caractère « >**&** ». Il est aussi important de savoir dans quel sens le shell interprète les redirections. Les redirections étant en principe en fin de commande, le shell recherche d'abord les caractères « <, >, >> » en fin de ligne. Ainsi si nous voulons grouper les deux canaux de sortie et d'erreur dans un même fichier, il faut procéder comme suit.

```
$ ls -l > resultat.txt 2>&1
```

La sortie 2 est redirigée vers la sortie 1, donc les messages d'erreurs passeront par la sortie standard. Puis le résultat de la sortie standard de la commande ls est redirigé vers le fichier resultat.txt. Ce fichier contiendra donc à la fois la sortie standard et la sortie d'erreur.

On peut aussi utiliser à la fois les deux types de redirection.

```
$ wc < resultat.txt > compte.txt
$ cat compte.txt
4 29 203
```

On peut aussi se reporter à la commande « **tee** »

V.3.4. Filtre : définition

Un **filtre** (ou une commande filtre) est un programme sachant écrire et lire des données par les canaux standards d'entrée et de sortie. Il en modifie ou traite éventuellement le contenu. wc est un filtre.

Nous nous attarderons sur quelques filtres plus tard, mais en voici quelques uns : **more** (affiche les données page par page), **sort** (tri des données), **grep** (critères de recherche)

V.3.5. Pipelines / tubes

Les redirections d'entrée/sortie telles que nous venons de les voir permettent de rediriger les résultats vers un fichier. Ce fichier peut ensuite être réinjecté dans un filtre pour en extraire d'autres résultats. Cela oblige à taper deux lignes : une pour la redirection vers un fichier, l'autre pour rediriger ce fichier vers le filtre. Les **tubes** ou **pipes** permet de rediriger directement le canal de sortie d'une commande vers le canal d'entrée d'une autre. Le caractère permettant cela est « **|** ».

```
$ ls -l > resultat.txt
$ wc < resultat.txt
```

devient...

```
$ ls -l | wc
```

Il est possible de placer plusieurs « **|** » sur une même ligne.

```
$ ls -l | wc | wc
1 3 24
```

La première commande n'est pas forcément un filtre. L'essentiel est qu'un résultat soit délivré. Idem pour la dernière commande qui peut par exemple être une commande d'édition ou d'impression. Enfin, la dernière commande peut elle-même faire l'objet d'une redirection en sortie.

```
$ ls -l | wc > resultat.txt
```

SUPPORT DE COURS

CHAPITRE VI : GESTION DES UTILISATEURS, DES GROUPES ET LES DROIT D'ACCES

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

Lorsque vous utilisez la machine en tant que superutilisateur (root, ou administrateur), vous avez un accès total à chaque fichier du système. Le répertoire personnel du superutilisateur est /root et ce répertoire est monté sur la racine (/)

Dans la terminologie anglo−saxonne, il y a des ambiguïtés: le répertoire−parent de votre système (/) est appelé ***root***. Le superutilisateur s'appelle ***root*** et son répertoire personnel est **/root.**

Sauf en ce qui concerne le superutilisateur, les autres utilisateurs ont un accès limité aux fichiers et aux répertoires.

Utilisez toujours votre machine comme un utilisateur classique de manière à ne vous connecter en root (superutilisateur) que pour des travaux d'administration. Dans ce chapitre, nous montrons comment créer manuellement et automatiquement des comptes utilisateurs.

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

VI.1 Droits associés aux fichiers.

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

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

VI.2 Le fichier des mots de passe: /etc/passwd.

Le fichier ***/etc/passwd*** est le seul dans tout le système où le nom d'un utilisateur est enregistré.

Des exceptions à cette règle:

(i) plusieurs schémas d'authentification distribués et

(ii) le paquet logiciel Samba.

Une fois qu'un utilisateur est additionné à cette liste, on dit qu'il "existe" sur le système. Listez le fichier **passwd** bien connu des administrateurs, avec :

more /etc/passwd

Chaque utilisateur est enregistré sur une ligne différente. La plupart d'entre elles ne correspondent pas à des utilisateurs physiques mais sont associés à des programmes.

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

Ahmed:x:511:512:Ahmed Choukri : /home/ahmed:/bin/bash

- **ahmed** : est l'identifiant de l'utilisateur. Il devrait être composé de minuscule et de nombres. D'autres caractères sont permis, mais cela n'est pas souhaitable. En particulier, il ne devrait jamais y avoir deux noms d'utilisateurs ne différant que par une majuscule.
- **x** : est un signe masquant le mot de passe crypté et indiquant que ce dernier se trouve dans le fichier **/etc/shadow**. Le fichier de mots de passe cryptés (**shadow password**) est un fichier additionnel des systèmes UNIX. Il contient des informations supplémentaires à propos des utilisateurs.
- **511** : ce nombre est l'**UID** ou **User IDentifier** de l'utilisateur. il est utilisé par les programmes pour identifier tout utilisateur.

En fait, en interne, seul l'UID −et non le nom de connexion− qui est utilisé].

- **512** : ce nombre est le GID ou Group IDentifier.
- **Ahmed Choukri** : est le nom complet de l'utilisateur.
- **/home/ahmed** : représente le répertoire personnel de l'utilisateur ahmed. La variable d'environnement **HOME** sera fixée à cette valeur lorsque l'utilisateur se connectera.
- **/bin/bash :** est le Shell à démarrer lors de la connexion de l'utilisateur.

VI.3 Le fichier /etc/shadow.

Le problème des fichiers **passwd** traditionnels est qu'ils sont lisibles par tous (world readable) [n'importe qui sur le système peut lire le fichier passwd].Ceci permet aux programmes d'y extraire des informations au sujet d'un utilisateur, telles que son nom complet. Ceci veut donc dire que tout le monde peut lire le mot de passe crypté dans le second champ. Aussi, tout le monde peut−il copier le champ "mot de passe" d'un utilisateur et essayer des milliards de

mots de passe jusqu'à ce qu'un des mots cryptés ainsi composé corresponde au mot de passe crypté de l'utilisateur.

Si vous avez une centaine d'utilisateurs sur votre système, il se peut que plusieurs d'entre eux choisissent un mot du dictionnaire comme mot de passe.

Les attaques par le dictionnaire consistent à tester les 80.000 mots d'anglais courant pour trouver une correspondance. Si vous pensez qu'il est astucieux d'ajouter un nombre en préfixe à un mot du dictionnaire, sachez que les algorithmes de craquage des mots de passe savent ça aussi [comme c'est d'ailleurs le cas pour d'autres astuces].

C'est pour résoudre ce problème que les mots de passe cryptés ont été inventés. Le fichier des mots de passe cryptés est exclusivement utilisé lors de l'étape d'authentification [en vérifiant que l'utilisateur est le véritable détenteur du compte]. Il n'est pas lisible par tous et aucun utilisateur normal n'a le droit de voir le champ ``mot de passe chiffré''. Il n'y a pas d'information dans ce fichier nécessaire aux programmes. Les champs du fichier/etc/shadow sont séparés par des doubles points comme c'est le cas du fichier /etc/passwd:

ahmed:Q,Jpl.ora6474e7:10795:0:99999:7:-1:-1:134537220

- **ahmed** : est le nom d'utilisateur
- **Q,Jpl.ora6474e7**: est le mot de passe chiffré résultant d'une transformation irréversible d'un mot de passe à 8 caractères.

Un algorithme mathématique est appliqué au mot de passe de manière à produire un résultat unique pour chaque mot de passe. Pour s'en convaincre, le mot de passe Loghimin (plutôt simple) est converti en: 1Z1F.0VSRRucs dans le fichier /etc/shadow. Le mot quasi−identique loghimin donne une transformation totalement différente: CavHIpD1w.cmg. Donc pour retrouver le mot de passe à partir du mot chiffré, il faudrait essayer chaque mot possible.

Bien qu'une attaque brutale de ce type soit considérée comme coûteuse du point de vue de la charge de calcul, elle n'est pas impossible. Pour vérifier si un mot de passe est bon, il suffit de lui appliquer l'algorithme permettant de vérifier la correspondance avec le mot de passe crypté. C'est comme cela que fonctionne la commande login.

Si vous voyez un signe * à la place du mot chiffré, cela signifie que le compte a été désactivé.

- **10795** : c'est le nombre de jours entre le 1er janvier 1970 et la date de dernier changement du mot de passe.
- **0** : c'est le nombre de jours avant que le mot de passe ne puisse être changé. Usuellement, la valeur est = 0. Ce champ n'est pas souvent utilisé.
- **99999** : correspond au nombre de jours après quoi le mot de passe doit être modifié. Ce champ est rarement utilisé. Par défaut, sa valeur est 99999.
- **7** : nombre de jours pour lequel un utilisateur est averti que son mot de passe expirera.
- **−1** : est le nombre de jours après quoi le mot de passe expire de sorte que le compte est désactivé. −1 est utilisé pour indiquer un nombre infini de jours (ce qui veut dire que cette propriété du compte n'est pas considérée).
- **−1** : est le nombre de jours entre le 1er janvier 1970 et la date de désactivation du compte.
- **134537220** est un indicateur réservé à un usage ultérieur.

VI.4 La commande groups et /etc/group.

Vous désirerez peut−être donner les mêmes droits d'accès à un certain nombre d'utilisateurs de votre système LINUX. Supposons, par exemple, cinq

utilisateurs qui seront autorisés à accéder à un fichier et dix autres auxquels vous donnerez le droit d'exécuter un programme. Vous pouvez faire deux groupes, l'un appelé previl et l'autre wproc. Le fichier et les répertoires appropriés seront attribués sélectivement comme ceci, par exemple:

```
chown root:previl /home/un_fichier

chown root:wproc /usr/lib/wproc
```

Les droits imposent le type d'accès, mais le fichier ou le répertoire doit au préalable être la propriété de ce groupe avant que ne se discutent les droits d'accès.

Le fichier ***/etc/group*** présente aussi des champs séparés par des double−points. Une ligne typique se présente comme ceci:

wproc:x:524:ahmed,tariq,user,saad,soufiane

- **wproc :** est le nom du groupe. Il devrait également y avoir un utilisateur portant ce nom.
- **x** : c'est le mot de passe du groupe. Ce champ est usuellement occupé par un x et n'est pas utilisé.
- **524** : est le GID ou Group IDentifier qui doit être unique dans le fichier group.
- ***ahmed,tariq,user,saad,soufiane*** forment la liste des utilisateurs appartenant à ce groupe. Il doit y avoir une virgule (sans espace) comme séparateur.

Vous pouvez bien sûr étudier le fichier **/etc/group** pour trouver les groupes auquel un utilisateur appartient

ce qui n'est pas la même chose −et c'est facile à voir− que de déterminer quels utilisateurs forment tel ou tel groupe.

Toutefois, quand il y a de nombreux groupes, cette méthode est fastidieuse par le balayage complet du fichier qu'elle impose. La commande groups affiche cette information.

VI.5 Création manuelle d'un compte utilisateur.

Les étapes suivantes sont requises pour la création manuelle d'un compte d'utilisateur:

1. **/etc/passwd :** Pour créer une entrée dans ce fichier, modifiez simplement celui−ci et copiez une ligne existante

> Lors de la modification de fichiers de configuration, n'écrivez jamais une ligne si celle−ci présente un format particulier. Dupliquez toujours une entrée existante connue pour fonctionner correctement, et modifiez−la ensuite. Cette pratique évitera des erreurs. Ajoutez toujours les utilisateurs dans le bas du fichier et essayez de préserver le style du fichier. C'est−à−dire que si vous voyez que des nombres augmentent de ligne en ligne, poursuivez avec des nombres croissants pour la nouvelle entrée; si vous ajoutez un utilisateur normal, faites−le à la suite des lignes existantes qui se rapportent aux utilisateurs normaux. Chaque utilisateur doit avoir un UID unique et devrait aussi avoir un GID unique. Ainsi, si vous ajoutez une ligne à la fin du fichier, augmentez d'une unité les UID et GID de la dernière ligne servant de modèle.

2. **/etc/shadow** : Créez une nouvelle entrée pour le mot de passe crypté (shadow password). A cette étape, vous n'en connaissez pas la valeur. Donc, mettez un *****. Vous déterminerez le mot de passe ultérieurement avec la commande passwd.
3. **/etc/group** : Créez une nouvelle entrée pour le groupe de l'utilisateur. Vérifiez bien que le nombre du groupe corresponde à celui indiqué dans le fichier passwd.

4. **/etc/skel** : Ce répertoire contient un modèle de répertoire personnel pour les utilisateurs. Copiez le répertoire en entier et tout son contenu dans **/home**; et renommez−le pour l'utilisateur. Dans le cas de notre exemple, **saad**; vous obtiendrez ainsi un ***/home/saad***.
5. **Droits du répertoire personnel** : Vous devez, à présent, modifier les droits de propriétés du répertoire personnel pour qu'ils correspondent au nouvel utilisateur. La commande ***chown −R saad:saas /home/saad*** réalise cet objectif.
6. **Mot de passe** : Utilisez la commande ***passwd <nom_utilisateur>*** pour attribuer un mot de passe.

VI.6 Méthode automatique: useradd et groupadd.

La méthode décrite précédemment est plutôt fastidieuse. Les commandes réalisant les mises à jour automatiquement sont ***useradd, userdel*** et ***usermod***. Les pages de man expliquent ces commandes en détail.

Notez que les divers types d'UNIX présentent des commandes différentes. Certaines distributions offrent des interfaces graphiques ou web comme assistants à la création, suppression, modification des utilisateurs.

Par ailleurs, les commandes ***groupadd, groupdel*** et ***groupmod*** agissent similairement en ce qui concerne les groupes.

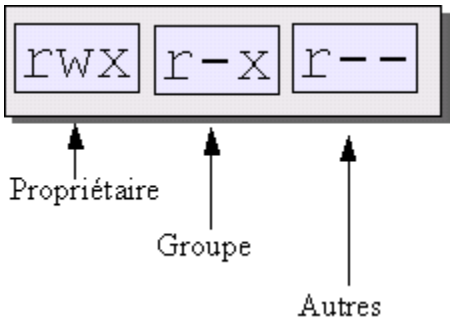
VI.7 Les droits d'accès

Nous avons indiqué que le rôle d'un système d'exploitation est aussi d'assurer la sécurité et l'accès aux données, ce qui se fait grâce au mécanisme des droits. Chaque fichier se voit attribué des droits qui lui sont propres, des autorisations d'accès individuelles. Lors d'un accès le système vérifie si celui-ci est permis.

La commande **id** permet d'obtenir ces informations. En interne, le système travaille uniquement avec les UID et GID, et pas avec les noms par eux-mêmes.

A chaque fichier (inode) sont associés un UID et un GID définissant son propriétaire et son groupe d'appartenance. On peut affecter des droits pour le propriétaire, pour le groupe d'appartenance et pour le reste du monde. On distingue de ce fait trois cas de figure

1. UID de l'utilisateur identique à l'UID défini pour le fichier : cet utilisateur est propriétaire du fichier.
2. Les UID sont différents : le système vérifie si le GID de l'utilisateur est identique au GID du fichier : si oui l'utilisateur appartient au groupe associé au fichier.
3. Dans les autres cas (aucune correspondance) : il s'agit du reste du monde (others), **ni les propriétaires, ni appartenant au groupe**.



Sortie de la commande ls -l :

d	rwxr-xr-x	1	oracle	dba	466	Feb 8 2001	oracle
-	rwxrwxrwx	1	oracle	dba	14536	Feb 8 2001	output.log

-	rwxr-x---	1	oracle	dba	1456	Feb 8 2001	launch.sh

Sur la première ligne du tableau, le répertoire oracle appartient à l'utilisateur oracle et au groupe dba, et possède les droits rwxr-xr-x.

VI.7.1. Signification

Droit	***Signification***
Général	
r	Readable (lecture)
w	Writable (écriture)
x	Executable (exécutable comme programme)
Fichier normal	
r	Le contenu du fichier peut être lu, chargé en mémoire, visualisé, recopié.
w	Le contenu du fichier peut être modifié, on peut écrire dedans. La suppression n'est pas forcément liée à ce droit (voir droits sur répertoire).
x	Le fichier peut être exécuté depuis la ligne de commande, s'il s'agit soit d'un programme binaire (compilé), soit d'un script (shell, perl, ...)
Répertoire	
r	Les éléments du répertoire (catalogue) sont accessibles en lecture. Sans cette autorisation, le ls et les critères de filtres sur le répertoire et son contenu ne sont pas possibles. **Ce droit ne suffit pas pour entrer**

	dans le catalogue.
w	Les éléments du répertoire (catalogue) sont modifiables et il est possible de créer, renommer et supprimer des fichiers dans ce répertoire. On voit donc que c'est ce droit qui contrôle l'autorisation de suppression d'un fichier **même si on est pas propriétaire des fichiers du répertoire**.
x	Le catalogue peut être accédé par cd et listé. Sans cette autorisation il est impossible d'accéder et d'agir sur son contenu qui devient verrouillé. Avec uniquement ce droit les fichiers et répertoires inclus dans celui-ci peuvent être accédés mais il faut alors obligatoirement connaître leur nom.

Ainsi pour un fichier :

rwx	***r-x***	***r--***
Droits de l'utilisateur, en lecture, écriture et exécution	Droits pour les membres du groupe et lecture et exécution	Droits pour le reste du monde en lecture uniquement

VI.7.2. Modification des droits

Lors de sa création, un fichier ou un répertoire dispose de droits par défaut. On utilise la commande **chmod** (change mode) pour modifier les droits sur un fichier ou un répertoire. Il existe de méthodes pour modifier ces droits : par la forme symbolique et par la base 8. Seul le propriétaire d'un fichier peut en modifier les droits (sauf l'administrateur système).

Le chmod sur un lien symbolique est possible comme sur tout autre fichier, mais cela ne modifie pas les droits du lien par lui-même mais les droits du fichier pointé.

VI.7.2.1. Par symboles

La syntaxe est la suivante :

chmod modifications Fic1 [Fic2...]

S'il faut modifier les droits de l'utilisateur, on utilisera le caractère « **u** ». pour les droits du groupe, le caractère « **g** », pour le reste du monde le caractère « **o** », pour tous le caractère « **a** ».

Pour ajouter des droits, on utilise le caractère « **+** », pour en retirer le caractère « **-** », et pour ne pas tenir compte des paramètres précédents le caractère « **=** ».

Enfin, le droit d'accès par lui-même : « **r** », « **w** » ou « **x** ».

On peut séparer les modifications par un espace, et cumuler plusieurs droits dans une même commande.

```
$ ls -l
total 0
-rw-r--r-- 1 oracle system 0 Aug 12 11:05 fic1
-rw-r--r-- 1 oracle system 0 Aug 12 11:05 fic2
-rw-r--r-- 1 oracle system 0 Aug 12 11:05 fic3

$ chmod g+w fic1
$ ls -l fic1
```

```
-rw-rw-r-- 1 oracle system 0 Aug 12 11:05 fic1

$ chmod u=rwx,g=x,o=rw fic2
$ ls -l fic2
-rwx--xrw- 1 oracle system 0 Aug 12 11:05 fic2

$ chmod o-r fic3
$ ls -l fic3
-rw-r----- 1 oracle system 0 Aug 12 11:05 fic3
```

VI.7.2.2. Par base 8

La syntaxe est identique à celle des symboles. A chaque droit correspond une valeur **octale c'est à dire de zéro (0) à sept (7)**, positionnelle et cumulable.

Propriétaire			*Groupe*			*Reste du monde*		
r	*w*	*x*	*r*	*w*	*x*	*r*	*w*	*x*
400	200	100	40	20	10	4	2	1

Pour obtenir le droit final il suffit d'additionner les valeurs. Par exemple si on veut rwxrw-rw- alors on fera 400+200+100+40+20+4+2=766, et pour rw-r-r- 400+200+40+4=644.

```
$ chmod 766 fic1
$ ls -l fic1
-rwxr-xr-x 1 oracle system 0 Aug 12 11:05 fic1

$ chmod 644 fic2
```

$ ls -l fic2

-rw-r--r-- 1 oracle system 0 Aug 12 11:05 fic2

VI.7.3. Masque des droits

Lors de la création d'un fichier ou d'un répertoire et qu'on regarde ensuite leurs droits, on obtient généralement rw-r-r-- (644) pour un fichier et rwxr-xr-x (755) pour un répertoire. Ces valeurs sont contrôlées par un masque, lui-même modifiable par la commande **umask**. la commande prend comme paramètre une valeur octale qui sera soustraite aux droits d'accès maximum. Par défaut, tous les fichiers sont créés avec les droits 666 (rw-rw-rw) et les répertoires avec les droits 777 (rwxrwxrwx), puis le masque est appliqué.

Sur la plupart des Unix, le masque par défaut est 022, soit ----w--w-. Pour obtenir cette valeur, on tape umask sans paramètre.

Pour un fichier :

Maximum rw-rw-rw- (666)
Retirer ----w--w- (022)
Reste rw-r--r-- (644)

Pour un répertoire :

Maximum rwxrwxrwx (777)
Retirer ----w--w- (022)
Reste rwxr-xr-x (755)

ATTENTION : la calcul des droits définitifs (effectifs) n'est pas une simple soustraction de valeurs octales ! Le masque retire des droits mais n'en ajoute pas.

Pour un fichier :

Maximum rw-rw-rw- (666)

Retirer ---r-xrw- (056)

Reste rw--w---- (620) **et PAS 610 !**

VI.7.4. Changement de propriétaire et de groupe

Il est possible de changer le propriétaire et le groupe d'un fichier à l'aide des commandes **chown** (change owner) et **chgrp** (change group).

chown utilisateur fic1 [Fic2...]

chgrp groupe fic1 [Fic2...]

Sur les UNIX récents seul root peut utiliser chown. La commande chgrp peut être utilisée par n'importe qui à condition que cet utilisateur fasse aussi partie du nouveau groupe.

En précisant le nom d'utilisateur (ou de groupe), le système vérifie d'abord leur existence. On peut préciser un UID ou un GID, dans ce cas le système n'effectuera pas de vérification.

Pour les deux commandes on peut préciser l'option **-R**, dans ce cas les droits seront changés de manière récursive. Les droits précédents et l'emplacement du fichier ne sont pas modifiés. Enfin sous certains Unix il est possible de modifier en une seule commande à la fois le propriétaire et le groupe.

chown utilisateur[:groupe] fic1 [fic2...]

chown utilisateur[.groupe] fic1 [fic2...]

$ chgrp dba fic1

```
$ ls -l fic1
-rwxr-xr-x 1 oracle dba 0 Aug 12 11:05 fic1
```

SUPPORT DE COURS

CHAPITRE VII : GESTION DES PROCESSUS

VII.1. Définition et environnement

Un **processus** représente à la fois un programme en cours d'exécution et tout son environnement d'exécution (mémoire, état, identification, propriétaire, père ...). Voir les cours de Système d'exploitation.

Voici une liste des données d'identification d'un processus :

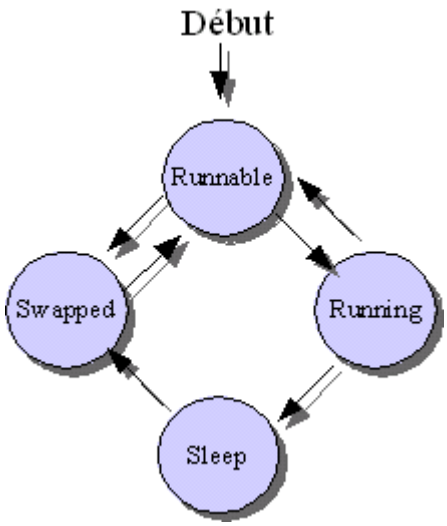
- **Un numéro de processus unique PID** (Process ID) : chaque processus Unix est numéroté afin de pouvoir être différencié des autres. Le premier processus lancé par le système est 1 et il s'agit d'un processus appelé généralement init. On utilise le PID quand on travaille avec un processus. Lancer 10 fois le même programme (même nom) donne 10 PID différents.
- **Un numéro de processus parent PPID** (Parent Process ID) : chaque processus peut lui-même lancer d'autres processus, des processus enfants (child process). Chaque enfant reçoit parmis les informations le PID du processus père qui l'a lancé. Tous les processus ont un PPID sauf le processus 0 qui est un pseudo-processus représentant le démarrage du système (créé le 1init).
- **Un numéro d'utilisateur et un numéro de groupe** : correspond à l'UID et au GID de l'utilisateur qui a lancé le processus. C'est nécessaire pour que le système sache si le processus à le droit d'accéder à certaines ressources ou non. Les processus enfants héritent de ces informations. Dans certains cas (que nous verrons plus tard) on peut modifier cet état.

- **Durée de traitement et priorité** : la durée de traitement correspond au temps d'exécution écoulé depuis le dernier réveil du processus. Dans un environnement multitâches, le temps d'exécution est partagé entre les divers processus, et tous ne possèdent pas la même priorité. Les processus de plus haute priorité sont traités en premier. Lorsqu'il est inactif sa priorité augmente afin d'avoir une chance d'être exécuté. Lorsqu'il est actif, sa priorité baisse afin de laisser sa place à un autre. C'est le scheduler du système qui gère les priorités et les temps d'exécution.
- **Répertoire de travail actif** : A son lancement, le répertoire courant (celui depuis lequel le processus a été lancé) est transmis au processus. C'est ce répertoire qui servira de base pour les chemins relatifs.
- **Fichiers ouverts** : table des descripteurs des fichiers ouverts. Par défaut au début seuls trois sont présents : 0 1 et 2 (les canaux standards). A chaque ouverture de fichier ou de nouveau canal, la table se remplit. A la fermeture du processus, les descripteurs sont fermés (en principe).
- On trouve d'autres informations comme la taille de la mémoire allouée, la date de lancement du processus, le terminal d'attachement, l'état du processus, UID effectif et Reel ainsi que GID effectif et réel.

VII.2. Etats d'un processus

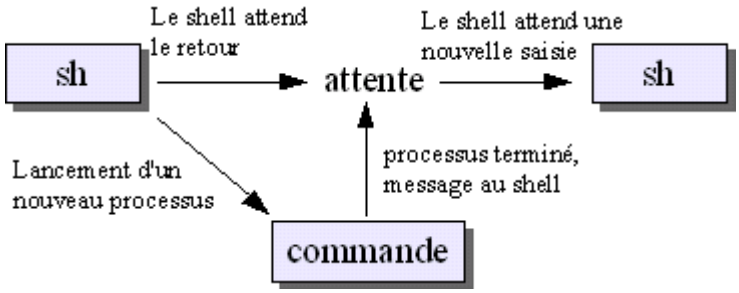
Durant sa vie (temps entre le lancement et la sortie) un processus peut passer par divers états ou **process state** :

- Exécution en mode utilisateur (**user mode**)
- Exécution en mode noyau (**kernel mode**)
- En attente E/S (**waiting**)
- Endormi (**sleeping**)
- Prêt à l'exécution (**runnable**)
- Endormi dans le swap (**mémoire virtuelle**)
- Nouveau processus
- Fin de processus (**Zombie**)



VII.3. Lancement en tâche de fond

En suivant ce que nous avons vu avant, le système étant multi-tâches un certain nombre de processus tournent déjà sur la machine sans que nous le voyons. De même le Shell que nous utilisons est lui-même un processus. Quand une commande est saisie, le Shell créé un nouveau processus pour l'exécuter, ce processus devient un processus enfant du Shell. Jusqu'à présent il fallait attendre la fin de l'exécution d'une commande pour en saisir une autre (sauf à utiliser le « ; » pour chaîner les commandes).



Rien n'empêche le Shell d'attendre le message du processus terminé pour rendre la main : de ce fait la commande une fois lancé, le Shell peut autoriser la saisie d'une nouvelle commande sans attendre la fin de l'exécution de la commande précédente. Pour cela il suffit de saisir après avoir tapé la commande le **ET Commercial « & »**. Dans ce cas le Shell et la commande lancée fonctionneront en parallèle.

```
$ ls -R / > ls.txt 2>/dev/null &
[1] 21976

$
[1] Done ls -l -R / > ls.txt 2>/dev/null
$ ls
fic1 fic3 liste ls.txt rep1 users
fic2 fic4 liste2 mypass toto.tar.gz
```

Juste après la saisie un chiffre apparaît, il est à retenir car il s'agit du PID du nouveau processus lancé. Après une autre saisie une ligne Done indique que le traitement est terminé. La valeur [1] est propre à un Shell particulier (ksh).

Quelques remarques sur l'utilisation du lancement en tâche de fond :

1. Le processus lancé ne devrait pas attendre de saisie au risque de confusion entre cette commande et le Shell lui-même.

2. Le processus lancé ne devrait pas afficher de résultats sur l'écran au risque d'avoir des affichages en conflit avec celui du Shell (par exemple apparition d'une ligne en milieu de saisie).
3. Enfin, quand on quitte le Shell, on quitte aussi tous ses fils : dans ce cas ne pas quitter le Shell pendant un traitement important.

VII.3.1. wait

Lorsqu'un processus est en tâche de fond, il est possible de récupérer et d'attendre la fin de la dernière commande lancée en arrière-plan avec la commande **wait [pid]**. S'il s'agit d'une autre commande, alors il est important d'avoir noté le PID de celle-ci lorsqu'elle a été lancée.

VII.4. Liste des processus

La commande **ps** (process status) permet d'avoir des informations sur les processus en cours. Lancée seule, elle n'affiche que les processus en cours lancés depuis l'utilisateur et la console actuels. Nous détaillerons les colonnes plus loin.

```
$ ps
PID TTY S TIME CMD
22608 ttyp0 S 0:00.04 -csh (csh)
```

Pour avoir plus d'informations, on peut utiliser l'option **-f**.

```
$ ps -f
UID PID PPID C STIME TTY TIME CMD
oracle 22608 12288 0.0 15:20:09 ttyp0 0:00.05 -csh (csh)
```

L'option -e donnes des informations sur tous les processus en cours.

```
$ ps -ef
UID PID PPID C STIME TTY TIME CMD
root 0 0 0.0 Aug 11 ?? 2:26.95 [kernel idle]
root 1 0 0.0 Aug 11 ?? 0:01.87 /sbin/init -a
root 3 1 0.0 Aug 11 ?? 0:00.27 /sbin/kloadsrv
root 51 1 0.0 Aug 11 ?? 1:30.22 /sbin/update
root 165 1 0.0 Aug 11 ?? 0:00.35 /usr/sbin/syslogd
root 169 1 0.0 Aug 11 ?? 0:00.02 /usr/sbin/binlogd
root 229 1 0.0 Aug 11 ?? 0:00.00 /usr/sbin/routed
root 260 1 0.0 Aug 11 ?? 0:01.65 /usr/sbin/rwhod
root 349 1 0.0 Aug 11 ?? 0:00.02 /usr/sbin/portmap
root 351 1 0.0 Aug 11 ?? 0:00.00 /usr/sbin/nfsiod 7
root 354 1 0.0 Aug 11 ?? 0:00.01 /usr/sbin/rpc.statd
root 356 1 0.0 Aug 11 ?? 0:00.01 /usr/sbin/rpc.lockd
root 587 1 0.0 Aug 11 ?? 0:00.23 /usr/sbin/xntpd -gx -l
root 626 1 0.0 Aug 11 ?? 0:00.06 /usr/sbin/snmpd
root 646 1 0.0 Aug 11 ?? 0:00.84 /usr/sbin/inetd
root 654 1 0.0 Aug 11 ?? 0:00.01 /usr/sbin/svrMgt_mib
root 655 1 0.0 Aug 11 ?? 0:00.03 /usr/sbin/svrSystem_mi
root 657 1 0.0 Aug 11 ?? 0:00.03 /usr/sbin/os_mibs
root 659 1 0.0 Aug 11 ?? 0:00.08 /usr/sbin/cpq_mibs
root 704 1 0.0 Aug 11 ?? 0:01.46 /usr/sbin/cron
root 718 1 0.0 Aug 11 ?? 0:00.01 /usr/lbin/lpd
...
```

L'option **-u** permet de préciser une liste d'un ou plusieurs utilisateurs séparés par une virgule. L'option **-g** effectue la même chose mais pour les groupes, **-t** pour les terminaux et **-p** pour des PID précis.

```
$ ps -u oracle
PID TTY S TIME CMD
2308 ?? S 0:02.27 /mor/app/oracle/product/8.1.7/bin/tnslsnr LISTENER -
inherit
>2311 ?? S 0:00.98 /mor/app/oracle/product/8.1.7/bin/tnslsnr DIVERS -inherit
2334 ?? S 0:00.22 ora_pmon_ORAP
2336 ?? S 0:01.79 ora_dbw0_ORAP
2338 ?? S 0:02.94 ora_lgwr_ORAP
2340 ?? S 0:20.23 ora_ckpt_ORAP
2342 ?? S 0:04.42 ora_smon_ORAP
2344 ?? S 0:00.09 ora_reco_ORAP
2346 ?? S 0:03.11 ora_snp0_ORAP
2348 ?? S 0:02.73 ora_snp1_ORAP
2350 ?? S 0:02.52 ora_snp2_ORAP
2352 ?? S 0:02.69 ora_snp3_ORAP
2354 ?? S 0:00.08 ora_arc0_ORAP
2369 ?? S 0:00.21 ora_pmon_OLAP
2371 ?? S 0:00.22 ora_dbw0_OLAP
...
```

Enfin l'option **-l** propose plus d'informations techniques.

```
$ ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
80808001 S 75 22608 12288 0.0 44 0 0 424K pause ttyp0 0:00.06 csh
```

Colonne	*Définition*
UID	User ID, nom de l'utilisateur

PID	Process ID, numéro du processus
PPID	Parent Process ID, numéro du processus père
C	Facteur de priorité, plus la valeur est grande plus la priorité est élevée
STIME	Heure de lancement du processus
TTY	Nom du terminal depuis lequel le processu a été lancé.
TIME	Durée de traitement du processus
CMD	Commande exécutée
F	Drapeaux du processus (sort du cadre du cours)
S	Etat du processus S (sleeping) R (running) Z (zombie)
PRI	Priorité du processus
NI	Nice, incrément pour le scheduler

VII.5. Arrêt d'un processus / signaux

Lorsqu'un processus tourne en tâche de fond il ne peux pas être arrêté par une quelconque combinaison de touches. Pour cela il faut employer la commande **kill**. Contrairement à ce que son nom semble indiquer, le rôle de cette commande n'est pas forcément de détruire ou de terminer un processus (récalcitrant ou non), mais d'envoyer des signaux aux processus.

$kill [-l] -Num_signal PID [PID2...]

SUPPORT DE COURS

Le **signal** est l'un des moyens de communication entre les processus. Lorsqu'on envoie un signal à un processus, celui-doit doit l'intercepter et réagir en fonction de celui-ci. Certains signaux peuvent être ignorés, d'autres non. Suivant les Unix on dispose d'un nombre plus ou moins important de signaux. Les signaux sont numérotés et nommés, mais attention si les noms sont généralement communs d'un Unix à l'autre, les numéros ne le sont pas forcément. L'option « -l » permet d'obtenir la liste des signaux.

Voici ceux qui nous concernent.

Signal	***Rôle***
1 (SIGHUP)	Hang Up, est envoyé par le père à tous ses enfants lorsqu'il se termine
2 (SIGINT)	Interruption du processus demandé (touche suppr, Ctrl+C)
3 (SIGQUIT)	Idem SIGINT mais génération d'un Core Dump (fichier de débuggage)
9 (SIGKILL)	Signal ne pouvant être ignoré, force le processus à finir 'brutalement'.
15 (SIGTERM)	Signal envoyé par défaut par la commande kill. Demande au processus de se terminer normalement.

$ ps

PID TTY S TIME CMD

22608 ttyp0 S 0:00.04 -csh (csh)

22610 ttyp0 R 0:00.05 ls

$ kill 22610

22610 Terminated

...

$ ps

PID TTY S TIME CMD

22608 ttyp0 S 0:00.04 -csh (csh)

22615 ttyp0 R 0:00.05 ls

$ kill -9 22615

22610 (killed)

VII.6. nohup

Quand le Shell est quitté (exit, ctrl+D, ...) nous avons vu que le signal 1 SIGHUP est envoyé aux enfants pour qu'ils se terminent aussi. Lorsqu'un traitement long est lancé en tâche de fond et que l'utilisateur veut quitter le Shell, ce traitement sera alors arrêté et il faudra tout recommencer. Le moyen d'éviter cela est de lancer le traitement (processus) avec la commande **nohup**. Dans ce cas le processus lancé ne réagira plus au signal SIGHUP, et donc le Shell peut être quitté, la commande continuera sont exécution.

Par défaut avec nohup (sous sh bash et ksh) les canaux de sortie et d'erreur standards sont redirigés vers un fichier **nohup.out**, sauf si la redirection est explicitement précisée.

$ nohup ls -lR / &

10206

$ Sending output to nohup.out

$ ls

fic1 fic3 liste ls.txt nohup.out toto.tar.gz

fic2 fic4 liste2 mypass rep1 users

VII.7. nice et renice

La commande **nice** permet de lancer une commande avec une priorité pus faible, afin de permettre éventuellement à d'autres processus de tourner plus rapidement.

nice [-valeur] commande [arguments]

nice [-n valeur] commande [arguments]

Les deux syntaxes sont généralement admises, à vérifier par man. La valeur représenta la priorité du traitement. Pour la seconde syntaxe une valeur positive causera une baisse de priorité, une négative l'augmentation de la priorité (si autorisé). La première syntaxe n'accepte qu'un abaissement de la priorité. La valeur doit être comprise entre 1 et 20. Plus la valeur est élevée et plus le traitement est ralenti.

$ nice -10 ls -lR / >liste 2>/dev/null&

10884

$ ps -l

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

80808001 S 75 10153 10115 0.0 44 0 0 424K pause ttyp4 0:00.09 csh

80808001 S + 75 10822 10153 0.0 44 0 0 192K wait ttyp4 0:00.01 sh

```
80808001 U N+ 75 10884 10822 28.5 54 10 0 848K aa3b3a9c ttyp4 0:03.32
ls
```

La commande **renice** fonctionne un peu comme nice sauf qu'elle permet de modifier la priorité en fonction d'un utilisateur, d'un groupe ou d'un PID. La commande doit donc déjà tourner.

renice [-n prio] [-p] [-g] [-u] ID

La priorité doit être comprise entre -20 et 20. L'utilisateur standard ne peut utiliser que les valeurs entre 0 et 20 permettant de baisser la priorité. L'option -p précise un PID, -g un GID et -u un UID.

VII.8. time

La commande **time** mesure les durées d'exécution d'une commande, idéal pour connaître les temps de traitement, et retourne trois valeurs :

1. **real** : durée totale d'exécution de la commande
2. **User** : durée du temps CPU nécessaire pour exécuter le programme
3. **System** : durée du temps CPU nécessaire pour exécuter les commandes liées à l'OS (appels système au sein d'un programme).

Le résultat est sorti par le canal d'erreur standard 2. On peut avoir une indication de la charge de la machine par le calcul Real / (User+System). Si le résultat est inférieur à 10, la machine dispose de bonnes performances, au-delà de 20 la charge de la machine est trop lourdes (performances basses).

```
$ time ls -lR /home
...
real 4.8
user 0.2
sys 0.5
```

7.9 Consommation mémoire et CPU: top.

La commande ***top*** classe tous les processus par leur consommation de la CPU et de la mémoire. Elle affiche le "**top 20**" sous forme d'une table. Lancez top chaque fois que vous voulez voir le (ou les) processus qui monopolise(nt) votre machine. La commande ***top −q −d 2*** est utile. Ceci permet le rafraîchissement de l'affichage à l'écran sans décalage (important).

top −n l −b > top.txt affiche tous les processus et ***top −n 1 −b −p <PID>*** affiche de l'information à propos d'un processus en désignant ce dernier par son PID. ***top*** présente des réponses interactives très utiles lorsque certaines touches sont pressées:

- *f* affiche une liste des champs que vous pouvez modifier interactivement. Par défaut, les seuls champs montrés sont USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND, ce qui constitue l'essentiel (la signification des champs est donnée ci−après),
- *r* redéfinit la priorité (renice) d'un processus,
- *k* tue un processus.

La page de man de top décrit la signification des champs. Certains de ceux−ci sont un peu déroutant et supposent une connaissance des programmes en C qui les constitutent. La principale question qui intéresse un utilisateur se résume souvent ainsi: combien de mémoire un processus utilise−t−il? La réponse est donnée par la champ RSS, qui signifie Resident Set Size. RSS désigne la quantité de RAM qu'un processus consomme à lui seul.

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

CHAPITRE VIII: LE SHELL

Le Shell n'est pas qu'un simple interpréteur de commandes, mais dispose d'un véritable langage de programmation avec notamment une gestion des variables, des tests et des boucles, des opérations sur variables, des fonctions...

VIII.1. Structure et exécution d'un script

Toutes les instructions et commandes sont regroupées au sein d'un script. Lors de sont exécution, chaque ligne sera lue une à une et exécutée. Une ligne peut se composer de commandes internes ou externes, de commentaires ou être vide. Plusieurs instructions par lignes sont possibles, séparées par le « **;** » ou liées conditionnellement par « && » ou « || ». Le « ; » est l'équivalent d'un **saut de ligne**.

Par convention les Shell scripts se terminent généralement (pas obligatoirement) par « .sh » pour le Bourne Shell et le Bourne Again Shell, par « .ksh » pour le Korn Shell et par « .csh » pour le C Shell.

Pour rendre un script exécutable directement :

```
$ chmod u+x monscript
```

Pour l'exécuter :

```
$ ./monscript
```

Quand un script est lancé, un nouveau Shell « fils » est créé qui va exécuter chacune des commandes. Si c'est une commande interne, elle est

directement exécutée par le nouveau Shell. Si c'est une commande externe, dans le cas d'un binaire un nouveau fils sera créé pour l'exécuter, dans le cas d'un Shell script un nouveau Shell fils est lancé pour lire ce nouveau Shell ligne à ligne.

Une ligne de commentaire commence toujours par le caractère « # ». Un commentaire peut être placé en fin d'une ligne comportant déjà des commandes.

```
# La ligne suivante effectue un ls
ls # La ligne en question
```

La première ligne a une importance particulière car elle permet de préciser quel Shell va exécuter le script

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

Dans le premier cas c'est un script Bourne, dans l'autre un script Korn.

VIII.2. Les variables

On en distingue trois types : utilisateur, système et spéciales. Le principe est de pourvoir affecter un contenu à un nom de variable, généralement un chaîne de caractère, ou des valeurs numériques.

VIII.2.1. Nomenclature

Un nom de variable obéit à certaines règles :

1. Il peut être composé de lettres minuscules, majuscules, de chiffres, de caractères de soulignement

2. Le premier caractère ne peut pas être un chiffre
3. Le taille d'un nom est en principe illimité (il ne faut pas abuser non plus)
4. Les conventions veulent que les variables utilisateur soient en minuscules pour les différencier des variables système. Au choix de l'utilisateur.

VIII.2.2. Déclaration et affectation

Une variable est déclarée dès qu'une valeur lui est affectée. L'affectation est effectuée avec le signe « **=** », sans espace avant ou après le signe.

```
var=Bonjour
```

On peut aussi créer des variables vides. Celle-ci existera mais sans contenu.

```
var=
```

VIII.2.3. Accès et affichage

On accède au contenu d'une variable en plaçant le signe « $ » devant le nom de la variable. Quand le Shell rencontre le « $ », il tente d'interpréter le mot suivant comme étant une variable. Si elle existe, alors le $nom_variable est remplacé par son contenu, ou par un texte vide dans le cas contraire.

On parle aussi de référencement de variable.

```
$ chemin=/tmp/seb
$ pwd
/home/toto
$ ls $chemin
```

dump.log fic4 lien_fic1 liste_ls rep1 seb2
fic1 hardlink2_fic2 lien_fic2 ls.txt rep2 toto.tar.gz
fic2 hardlink3_fic2 liste mypass resultat.txt users
fic3 hardlink_fic2 liste2 nohup.out seb1
$ cd $chemin
$ pwd
/tmp/seb
$ cd $chemin/rep1
$ pwd
/tmp/seb/rep1

Pour afficher la liste des variables on utilise la commande **set**. Elle affiche les variables utilisateur et les variables système, nom et contenu. La commande **echo** permet d'afficher le contenu de variables spécifiques.

$ a=Jules
$ b=Cesar
$ set
EDITMODE=emacs
HOME=/home/seb<
LD_LIBRARY_PATH=/mor/app/oracle/product/8.1.7/lib
LOGNAME=seb
MAIL=/usr/spool/mail/seb
MAILCHECK=600
MANPATH=/usr/man:/sqlbt/datatools/obacktrack/man
PATH=/home/oracle/bin:/usr/bin:.:/mor/app/oracle/product/8.1.7/bin:/sqlbt/data tools/obacktrack/bin:
PS1=$
PS2=>
SHELL=/bin/csh

```
SHLVL=1
TERM=xterm
USER=seb
a=Jules
b=Cesar
$ echo $a $b a conquis la Gaule
Jules Cesar a conquis la Gaule
```

Une variable peut contenir des caractères spéciaux, le principal étant l'espace. Mais

```
$ c=Salut les copains
les: not found
$ echo $c
```

Ne marche pas. Pour cela il faut soit verrouiller les caractères spéciaux un par un, soit de les mettre entre guillemets ou apostrophes.

```
c=Salut\ les\ Copains # Solution lourde
c="Salut les copains" # Solution correcte
c='Salut les copains' # Solution correcte
```

La principale différence entre les guillemets et les apostrophes est l'interprétation des variables et des substitutions. " et ' se vérouillent mutuellement.

```
$ a=Jules
$ b=Cesar
$ c="$a $b a conquis la Gaule"
$ d='$a $b a conquis la Gaule'
$ echo $c
```

Jules Cesar a conquis la Gaule

$ echo $d

$a $b a conquis la Gaule

$ echo "Unix c'est top"

Unix c'est top

$ echo 'Unix "trop bien"'

Unix "trop bien"

VIII.2.4. Suppression et protection

On supprime une variable avec la commande **unset**. On peut protéger une variable en écriture et contre sa suppression avec la commande **readonly**. Une variable en lecture seule même vide est figée. Il n'existe aucun moyen de la replacer en écriture et de la supprimer, sauf à quitter le Shell.

$ a=Jules

$ b=Cesar

$ echo $a $b

Jules Cesar

$ unset b

$ echo $a $b

Jules

$ readonly a

$ a=Neron

a: is read only

$ unset a

a: is read only

VIII.2.5. Exportation

Par défaut une variable n'est accessible que depuis le Shell où elle a été définie.

```
$ a=Jules
$ echo 'echo "a=$a"' > voir_a.sh
$ chmod u+x voir_a.sh
$ ./voir_a.sh
a=
```

La commande **export** permet d'exporter une variable de manière à ce que son contenu soit visible par les scripts et autres sous-Shells. Les variables exportées peuvent être modifiées dans le script, mais ces modifications ne s'appliquent qu'au script ou au sous-Shell.

```
$ export a
./voir_a.sh
a=Jules
$ echo 'a=Neron ; echo "a=$a"' >> voir_a.sh
$ ./voir_a.sh
a=Jules
a=Neron
$ echo $a
Jules
```

VII.2.6. Accolades

Les accolades de base « {} » permettent d'identifier le nom d'une variable. Imaginons la variable fichier contenant le nom de fichier 'liste'. On veut copier liste1 en liste2.

```
$ fichier=liste
$ cp $fichier2 $fichier1
usage: cp [-fhip] [--] source_file destination_file
or: cp [-fhip] [--] source_file ... destination_directory
or: cp [-fhip] [-R | -r] [--]
[source_file | source_directory] ... destination_directory
```

Ça ne fonctionne pas car ce n'est pas $fichier qui est interprété mais $fichier1 et $fichier2 qui n'existent pas.

```
$ cp ${fichier}2 ${fichier}1
```

Dans ce cas, cette ligne équivaut à

```
$ cp liste2 liste1
```

Les accolades indiquent que fichier est un nom de variable.

VIII.2.7. Accolades et remplacement conditionnel

Les accolades disposent d'une syntaxe particulière.

{variable:Remplacement}

Selon la valeur ou la présence de la variable, il est possible de remplacer sa valeur par une autre.

Remplacement	***Signification***
{x:-texte}	si la variable x est vide ou inexistante, le texte prendra sa place. Sinon c'est le contenu de la variable qui prévaudra.

{x:=texte}	si la variable x est vide ou inexistante, le texte prendra sa place et deviendra la valeur de la variable.
{x:+texte}	si la variable x est définie et non vide, le texte prendra sa place. Dans le cas contraire une chaîne vide prend sa place.
{x:?texte}	si la variable x est vide ou inexistante, le script est interrompu et le message texte s'affiche. Si texte est absent un message d'erreur standard est affiché.

$ echo $nom

$ echo ${nom:-Ahmed}

$ echo $nom

$ echo ${nom:=Ahmed}

Ahmed

$ echo $nom

Ahmed

$ echo ${nom:+"Valeur définie"}

Valeur définie

$ unset nom

$ echo ${nom:?Variable absente ou non définie}

nom: Variable absente ou non définie

$ nom=Ahmed

$ echo ${nom:?Variable absente ou non définie}

Ahmed

VIII.3. variables système

En plus des variables que l'utilisateur peux définir lui-même, le Shell est lancé avec un certain nombre de variables prédéfinies utiles pour un certain nombre

de commandes et accessibles par l'utilisateur. Le contenu de ces variables système peut être modifié mais il faut alors faire attention car certaines ont une incidence directe sur le comportement du système.

Variable	***Contenu***
HOME	Chemin d'accès du répertoire utilisateur. Répertoire par défaut en cas de cd.
PATH	Liste de répertoires, séparés par des « : » où le Shell va rechercher les commandes externes et autres scripts et binaires. La recherche se fait dans l'ordre des répertoires saisis.
PS1	Prompt String 1, chaîne représentant le prompt standard affiché à l'écran par le Shell en attente de saisie de commande.
PS2	Prompt String 2, chaîne représentant un prompt secondaire au cas où la saisie doit être complétée.
IFS	Internal Field Separator, liste des caractères séparant les mots dans une ligne de commande. Par défaut il s'agit de l'espace de la tabulation et du saut de ligne.
MAIL	chemin et fichier contenant les messages de l'utilisateur
MAILCHECK	intervalle en secondes pour la vérification de présence d'un nouveau courrier. Si 0 alors la vérification est effectuée après chaque commande.

SHELL	Chemin complet du Shell en cours d'exécution
LANG	Définition de la langue à utiliser ainsi que du jeu de caractères
LC_COLLATE	Permet de définir l'ordre du tri à utiliser (si LANG est absent)
LC_NUMERIC	Format numérique à utiliser
LC_TIME	Format de date et d'heure à utiliser
LC_CTYPE	Détermination des caractères et signes devant ou ne devant pas être pris en compte dans un tri.
USER	Nom de l'utilisateur en cours.
LD_LIBRARY_PATH	Chemin d'accès aux bibliothèques statiques ou partagées du système.
MANPATH	Chemin d'accès aux pages du manuel.
LOGNAME	Nom du login utilisé lors de la connexion.

VIII.4. Variables spéciales

Il s'agit de variables accessibles uniquement en lecture et dont le contenu est généralement contextuel.

Variable	***Contenu***
$?	Code retour de la dernière commande exécutée
$$	PID du Shell actif

$!	PID du dernier processus lancé en arrière-plan
$-	Les options du Shell

$ echo $$

23496

$ grep memoire liste

$ echo $?

1

$ grep souris liste

souris optique 30 15

$ echo $?

0

$ ls -lR >toto.txt 2<&1 &

26675

$ echo $!

26675

VIII.5. Paramètres de position

Les paramètres de position sont aussi des variables spéciales utilisées lors d'un passage de paramètres à un script.

VIII.5.1. Description

Variable	***Contenu***
$0	Nom de la commande (du script)
$1-9	$1,$2,$3... Les neuf premiers paramètres passés au script
$#	Nombre total de paramètres passés au script

$*	Liste de tous les paramètres au format "$1 $2 $3 ..."
$@	Liste des paramètres sous forme d'éléments distincts "$1" "$2" "$3" ...

$ cat param.sh

```
#!/usr/bin/sh
echo "Nom : $0"
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3"
echo "Liste : $*"
echo "Elements : $@"
```

```
$ param.sh riri fifi loulou
Nom : ./param.sh
Nombre de parametres : 3
Parametres : 1=riri 2=fifi 3=loulou
Liste : riri fifi loulou
Elements : riri fifi loulou
```

La différence entre **$@** et **$*** ne saute pas aux yeux. Reprenons l'exemple précédent avec une petite modification :

$ param.sh riri "fifi loulou"

```
Nom : ./param.sh
Nombre de parametres : 2
Parametres : 1=riri 2=fifi loulou 3=
Liste : riri fifi loulou
Elements : riri fifi loulou
```

Cette fois-ci il n'y a que deux paramètres de passés. Pourtant les listes semblent visuellement identiques. En fait si la première contient bien

"riri fifi loulou"

La deuxième contient

"riri" "fifi loulou"

Soit bien deux éléments. Dans les premier exemple nous avions

"riri" "fifi" "loulou"

VIII.5.2. redéfinition des paramètres

Outre le fait de lister les variables, la commande **set** permet de redéfinir le contenu des variables de position. Avec

set valeur1 valeur2 valeur3 ...

$1 prendra comme contenu valeur1, $2 valeur2 et ainsi de suite.

```
$ cat param2.sh
#!/usr/bin/sh
echo "Avant :"
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3 4=$4"
echo "Liste : $*"
set alfred oscar romeo zoulou
echo "apres set alfred oscar romeo zoulou"
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3 4=$4"
```

```
echo "Liste : $*"

./param2.sh riri fifi loulou donald picsou
Avant :
Nombre de parametres : 5
Parametres : 1=riri 2=fifi 3=loulou 4=donald
Liste : riri fifi loulou donald picsou
apres set alfred oscar romeo zoulou
Nombre de parametres : 4
Parametres : 1=alfred 2=oscar 3=romeo 4=zoulou
Liste : alfred oscar romeo zoulou
```

VIII.5.3. Réorganisation des paramètres

La commande **shift** est la dernière commande permettant de modifier la structure des paramètres de position. Un appel simple décale tous les paramètres d'une position en supprimant le premier : $2 devient $1, $3 devient $2 et ainsi de suite. Le $1 originel disparaît. $#, $* et $@ sont redéfinis en conséquence.

La commande shift suivie d'une valeur n effectue un décalage de n éléments. Ainsi avec

shift 4

$5 devient $1, $6 devient $2, ...

```
$ cat param3.sh
#!/usr/bin/sh
set alfred oscar romeo zoulou
echo "set alfred oscar romeo zoulou"
```

```
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3 4=$4"
echo "Liste : $*"
shift
echo "Après un shift"
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3 4=$4"
echo "Liste : $*"
```

$ ./param3.sh

```
set alfred oscar romeo zoulou
Nombre de parametres : 4
Parametres : 1=alfred 2=oscar 3=romeo 4=zoulou
Liste : alfred oscar romeo zoulou
Après un shift
Nombre de parametres : 3
Parametres : 1=oscar 2=romeo 3=zoulou 4=
Liste : oscar romeo zoulou
```

VIII.6. Sortie de script

La commande **exit** permet de mettre fin à un script. Par défaut la valeur retournée est 0 (pas d'erreur) mais n'importe quelle autre valeur de 0 à 255 peut être précisée. On récupère la valeur de sortie par la variable $?.

$ exit 1

VIII.7. Environnement du processus

En principe seules les variables exportées sont accessibles par un processus fils. Si on souhaite visualiser l'environnement lié à un fils (dans un script par exemple) on utilise la commande **env**.

```
$ env
PATH=/home/oracle/bin:/usr/bin:.:/mor/app/oracle/product/8.1.7/bin
TERM=xterm
LOGNAME=oracle
USER=oracle
SHELL=/bin/csh
HOME=/home/oracle
SHLVL=1
MAIL=/usr/spool/mail/oracle
ORACLE_SID=ORAP
EPC_DISABLED=TRUE
ORACLE_BASE=/mor/app/oracle
ORACLE_HOME=/mor/app/oracle/product/8.1.7
EDITMODE=emacs
ORATAB=/etc/oratab
```

La commande **env** permet de redéfinir aussi l'environnement du processus à lancer. Cela peut être utile lorsque le script doit accéder à une variable qui n'est pas présente dans l'environnement du père, ou qu'on ne souhaite pas exporter. La syntaxe est

env var1=valeur var2=valeur ... commande

Si la première option est le signe « - » alors c'est tout l'environnement existant qui est supprimé pour être remplacé par les nouvelles variables et valeurs.

```
$ unset a
$ ./voir_a.sh
a=
$ env a=jojo ./voir_a.sh
a=jojo
$ echo a=$a
a=
```

VIII.8. Substitution de commande

Le mécanisme de substitution permet de placer le résultat de commandes simples ou complexes dans une variable. On place les commandes à exécuter entre des accents graves « ` ». (Alt Gr + 7).

```
$ mon_unix=`uname`
$ echo ${mon_unix}
OSF1
$ machine=`uname -a | cut -d" " -f5`
$ echo $machine
alpha
```

Attention, seul le canal de sortie standard est affecté à la variable. Le canal d'erreur standard sort toujours vers l'écran.

VIII.9. Tests de conditions

La commande **test** permet d'effectuer des tests de conditions. Le résultat est récupérable par la variable $? (code retour). Si ce résultat est 0 alors la condition est réalisée.

VIII.9.1. tests sur chaîne

- **test -z "variable"** : zero, retour OK si la variable est vide (ex test -z "$a")
- **test -n "variable"** : non zero, retour OK si la variable n'est pas vide (texte quelconque)
- **test "variable"** = chaîne : OK si les deux chaînes sont identiques
- **test "variable"** != chaîne : OK si les deux chaînes sont différentes

```
$ a=
$ test -z "$a" ; echo $?
0
$ test -n "$a" ; echo $?
1
$ a=Jules
$ test "$a" = Jules ; echo $?
0
```

VIII.9.2. tests sur valeurs numériques

Les chaînes à comparer sont converties en valeurs numériques. La syntaxe est

test valeur1 option valeur2

et les options sont les suivantes :

Option	*Rôle*

-eq	Equal : Egal
-ne	Not Equal : Différent
-lt	Less than : inférieur
-gt	Greater than : supérieur
-le	Less ou equal : inférieur ou égal
-ge	Greater or equal : supérieur ou égal

$ a=10

$ b=20

$ test "$a" -ne "$b" ; echo $?

0

$ test "$a" -ge "$b" ; echo $?

1

$ test "$a" -lt "$b" && echo "$a est inferieur a $b"

10 est inferieur a 20

VIII.9.3. tests sur les fichiers

La syntaxe est

test option nom_fichier

et les options sont les suivantes :

Option	***Rôle***
-f	Fichier normal

-d	Un répertoire
-c	Fichier en mode caractère
-b	Fichier en mode bloc
-p	Tube nommé (named pipe)
-r	Autorisation en lecture
-w	Autorisation en écriture
-x	Autorisation en exécution
-s	Fichier non vide (au moins un caractère)
-e	Le fichier existe
-L	Le fichier est un lien symbolique
-u	Le fichier existe, SUID-Bit positionné
-g	Le fichier existe SGID-Bit positionné

$ ls -l

-rw-r--r-- 1 oracle system 1392 Aug 14 15:55 dump.log

lrwxrwxrwx 1 oracle system 4 Aug 14 15:21 lien_fic1 -> fic1

lrwxrwxrwx 1 oracle system 4 Aug 14 15:21 lien_fic2 -> fic2

-rw-r--r-- 1 oracle system 234 Aug 16 12:20 liste1

-rw-r--r-- 1 oracle system 234 Aug 13 10:06 liste2

-rwxr--r-- 1 oracle system 288 Aug 19 09:05 param.sh

-rwxr--r-- 1 oracle system 430 Aug 19 09:09 param2.sh

-rwxr--r-- 1 oracle system 292 Aug 19 10:57 param3.sh

drwxr-xr-x 2 oracle system 8192 Aug 19 12:09 rep1

```
-rw-r--r-- 1 oracle system 1496 Aug 14 16:12 resultat.txt
-rw-r--r-- 1 oracle system 1715 Aug 16 14:55 toto.txt
-rwxr--r-- 1 oracle system 12 Aug 16 12:07 voir_a.sh

$ test -f lien_fic1 ; echo $?
1
$ test -x dump.log ; echo $?
1
$ test -d rep1 ; echo $?
0
```

VIII.9.4. tests combinés par critères ET OU NON

On peut effectuer plusieurs tests avec une seule instruction. Les options de combinaisons sont les mêmes que pour la commande **find**.

Critère	*Action*
-a	AND, ET logique
-o	OR, OU logique
!	NOT, NON logique
(...)	groupement des combinaisons. Les parenthèses doivent être vérouillées \(...\).

```
$ test -d "rep1" -a -w "rep1" && exho "rep1: repertoire, droit en ecriture"
rep1: repertoire, droit en ecriture
```

VIII.9.5. syntaxe allégée

Le mot test peut être remplacé par les crochets ouverts et fermés « **[...]** ». Il faut respecter un espace après et avant les crochets.

```
$ [ "$a" -lt "$b" ] && echo "$a est inferieur a $b"
10 est inferieur a 20
```

VIII.10 Les alias

Un **alias** est une substitution d'une commande par un raccourci. L'alias est prioritaire sur les fonctions, commandes internes et commandes externes. Il est possible d'y substituer du texte.

```
alias nom_alias=commande_ou_texte

$ alias deltree='rm -rf'
$ deltree rep1
```

La substitution ne s'effectue que si l'alias est la première commande ou si le texte de l'alias se termine par un espace.

```
$ alias list='ls -l '
$ alias home='/tmp/seb'
$ list home
total 22
-rwxr--r-- 1 oracle system 327 Aug 19 13:31 case1.sh
-rw-r--r-- 1 oracle system 1392 Aug 14 15:55 dump.log
-rwxr--r-- 1 oracle system 200 Aug 19 15:58 expr1.sh
-rw-r--r-- 1 oracle system 42 Aug 19 15:43 fonctionRichEditWindow
```

```
-rwxr--r-- 1 oracle system 66 Aug 19 14:09 for2.sh
-rwxr--r-- 1 oracle system 285 Aug 19 14:32 for3.sh
-rwxr--r-- 1 oracle system 133 Aug 19 14:37 for4.sh
lrwxrwxrwx 1 oracle system 4 Aug 14 15:21 lien_fic1 -> fic1
lrwxrwxrwx 1 oracle system 4 Aug 14 15:21 lien_fic2 -> fic2
...
```

Sans paramètre, c'est la liste des alias qui s'affiche.

```
$ alias
autoload='typeset -fu'
cat=/usr/bin/cat
command='command '
deltree='rm -rf'
functions='typeset -f'
grep=/usr/bin/grep
hash='alias -t -'
history='fc -l'
home=/tmp/seb
integer='typeset -i'
list='ls -l '
local=typeset
ls=/usr/bin/ls
nohup='nohup '
r='fc -e -'
rm=/usr/bin/rm
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'
```

L'option **-x** permet d'exporter l'alias. Enfin la commande **unalias** supprime l'alias.

VIII.11 *Options du shell*

La commande **set** permet d'autres options que vi et emacs. L'option -o active l'option, +o l'annule. La commande « set -o » sans rien d'autre affiche la liste des options et leur état.

- **allexport** : toutes les variables déclarées seront automatiquement exportées
- **bgnice** : les processus lancés en tâche de fond ont un facteur nice plus important et donc tournent avec une priorité moindre
- **ignoreeof** : La combinaison Ctrl+D n'est plus interprétée.
- **noclobber** : la redirection > n'écrase plus le fichier et produit un message d'erreur s'il existe. Pour l'écraser tout de même : >|

```
$ set -o noclobber
$ wc -l toto.txt
378264 toto.txt
$ ls > toto.txt
ksh: toto.txt: file already exists
$ ls >| toto.txt
$ wc -l toto.txt
18 toto.txt
$ set +o noclobber
```

RESSOURCES

Pour Aller plus loin

Quelques URLs pour Les Logiciels Libres:

– Association Pour la Promotion et la Recherche en Informatique Libre :

http ://www.april.org/

– Association Francophone des Utilisateurs de Linux et des Logiciels Libres :

http ://www.aful.org/

– *Tribune Libre*, un livre des éditions O'Reilly :

http ://www.editions-oreilly.fr/divers/tribune-libre/

– Da Linux French Page : un portail en français (ou presque) avec l'actualité autour de Linux et des «mouvements informatiques alternatifs»

http ://linuxfr.org/

– FreshMeat : annonces et base de données de logiciels autour de Linux (pas seulement des logiciels libres)—*en anglais*

http ://www.freshmeat.net/

– Slashdot (/.) : rumeurs et annonces informatiques autour de la vague des logiciels libres —*en anglais*

http ://www.slashdot.org/

Sites «logithèques»

– http ://www.freshmeat.net/

– http ://www.linux-center.org/fr/

– http ://sourceforge.net/

– http ://www.linuxapps.com/

Quelques URLs pour Linux

SUPPORT DE COURS

– Le Projet de documentation LINUX :

www.tldp.org* et *www.linuxdoc.org

– Le système Linux en général :

http ://www.linux.org/

– Informations sur le développement du noyau Linux (assez technique) :

http ://www.kernel.org/ et http ://www.kernelnotes.org/

Sites de quelques distributions

http ://www.debian.org/

http ://www.redhat.com/

http://www.fedora-project.org

http ://www.linux-mandrake.com/fr/

http ://www.suse.com/

Sites sur le noyau

http ://www.kernel.org/

http ://www.kernelnotes.org/

http ://kt.linuxcare.com/

Grandes applications

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

http ://www.gimp.org/

http ://www.apache.org/ et http ://www.php.net/

http ://www.postgresql.org/

Quelques Livres :

– Eric Dumas. *Le Guide du Rootard.* Disponible sur Internet à l'adresse

http ://www.freenix.org/linux/Guide/

– Dominique Bouillet. *Unix par la pratique.* Ellipses, 1997.

– Mais surtout, les pages de manuel (man).