Avrillon Serge, le 22 Avril 2004

L'essentiel d'UNIX

Objectifs : Comprendre les fonctionnalités du système et exploiter aux mieux les ressources de la machine.

Pré-requis : Bonnes connaissances en informatique et si possible notions sommaires sur le système UNIX

Durée : 5 jours

Programme :

- *Histoire et présentation du système UNIX*
- *Présentation de Mandrakelinux 10.0 et installation*
- *Processus de boot*
- *Système de fichiers*
- *Manipulation de fichiers*
- *Gestion des utilisateurs*
- *Gestion des droits d'accès*
- *Utilisation de Linux en réseau*
- *L'éditeur de texte vi*
- *Lancement automatique de processus*
- *Sauvegarde de données*
- *Gestion et surveillance des processus*
- *Observation et Analyse du système*
- *Récupération d'un système de fichier*
- *ProFTP – Serveur de fichiers*
- *Programmation Shell*

Première version le 22 Avril 2004, revu le 4 Mars 2005 et encore revu le 3 mai2006.
serge.avrillon@free.fr

Histoire et présentation du système UNIX

UNIX Past

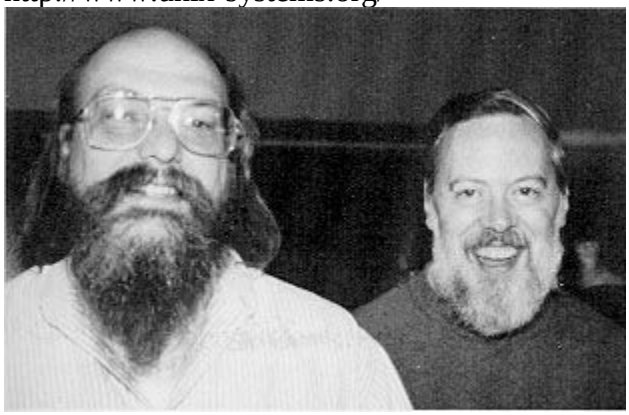
"...the number of UNIX installations has grown to 10, with more expected..."

- Dennis Ritchie and Ken Thompson, June 1972

"... When BTL withdrew from the project, they needed to rewrite an operating system (OS) in order to play space war on another smaller machine (a DEC PDP-7 [Programmed Data Processor] with 4K memory for user programs). The result was a system which a punning colleague called UNICS (UNiplexed Information and Computing Service)--an 'emasculated Multics'; no one recalls whose idea the change to UNIX was"

Source: A brief look at the early history

http://www.unix-systems.org/



Ken Thompson and Dennis Ritchie, 1970

- **En 1964 :**

Tout commence en 1964. General Electric, le MIT et les Bell Labs d'AT&T s'associent pour lancer le projet MULTICS (Multiplexed Information and Computing Service), combinant ordinateur et système d'exploitation, et qui doit répondre à deux besoins principaux :
- pouvoir être utilisé par plusieurs personnes à la fois : il doit donc s'agir d'un système d'exploitation "à temps partagé" (voir l'encadré) ;
- pouvoir en plus exécuter des calculs en tâche de fond.

- **En 1969 :**

Après 5 ans de développement non désicifs, les laboratoires Bell se retirent. Deux de leurs informaticiens, Ken Thompson et Dennis Ritchie, poursuivent sans l'appui de leur hiérarchie des travaux autour d'une sorte de MULTICS simplifié, baptisé UNICS (UNiplexed Information and

Computing Services). La légende veut que le nom UNICS ait également été choisi pour sa similitude de prononciation, en anglais, avec le mot "eunuchs" (eunuque), UNICS étant une version "castrée" de MULTICS !
Thompson avait récupéré un DEC PDP 7, une évolution du PDP-1, pour faire tourner Space Travel, un jeu qu'il avait développé. Ils l'utilisèrent donc pour débuter leur projet. UNICS reprend les principes introduits par son prédécesseur (les notions de processus, d'arborescence de fichiers, d'interpréteur de lignes de commandes en tant qu'application parmi d'autres, etc.) et y ajoute une approche résolument modulaire ainsi qu'une forme simple de communication entre applications (les données en sortie de l'une peuvent être récupérées en entrée par une autre). Il fut rebaptiser UNIX (par Brian Kernighan). Cette version est connue sous le nom "Unix Time-Sharing System Version 1".

- **En 1970 :**

Lancement de la ligne de mini-ordinateurs PDP-11 par Digital Equipment Corporation (DEC). Il s'agit d'une ligne de machines toutes compatibles entre elles basées sur un processeur 16 bits et qui rencontra un grand succès.

Mais il manque un langage de programmation "haut niveau" à UNIX (qui dispose pour l'instant d'un assembleur, langage très "bas niveau"). Ken Thompson travaille à combler cette lacune. Il pense d'abord à adapter le Fortran sur le PDP 7, puis crée finalement un langage de toutes pièces, appelée langage B, un langage interprété (en référence au BCPL dont il s'inspire).

- **En 1971 :**

Portage du nouveau système d'exploitation UNIX sur PDP 11/20. Ken Thompson et Dennis Ritchie ont obtenu cette machine en prétextant le développement d'un logiciel de traitement de textes, les responsables du Bell Labs ne voulant plus entendre parler de systèmes d'exploitation suite à l'abandon du projet MULTICS.

Le formateur de texte roff fût porté sur cette machine, ce qui permit à trois personnes du service des brevets d'utiliser effectivement la machine comme traitement de textes et ce en même temps que Thompson et Ritchie qui continuaient le développement d'applications.

Bell est convaincu, et le développement peut se poursuivre pour aboutir, la même année, au Unix Time-Sharing System Version 2.

Ken Thompson et Dennis Ritchie l'adaptent au DEC PDP-11/20. Il résultat de cette expérience le premier compilateur C. Le langage C a été conçu spécifiquement pour un OS portable

- **De 1971 à 1973 :**

Dennis Ritchie et Brian Kernighan du Bell Labs d'AT&T reprennent le langage B écrit par Ken Thompson pour PDP/7 en 1970 pour mieux l'adapter au PDP/11 sur lequel UNIX vient juste d'être porté. Ils font évoluer le langage et le dote d'un vrai compilateur générant du code machine PDP/11.

C'est le langage C qui est à la fois proche du matériel, permettant ainsi de réécrire le noyau UNIX en C et suffisamment généraliste, le rendant ainsi facilement portable. Les développements et les succès du langage C et d'UNIX sont intimement liés.

- **En 1973 :**

Le noyau du système UNIX est entièrement réécrit en C. Au vu de la qualité du résultat, tous les autres outils utilisés sous Unix vont être réécrits en C. Nous en sommes alors à la version 4 du Unix Time-Sharing System.

En raison d'une injonction du ministère de la justice qui n'a pas permis à AT&T de vendre le logiciel, le code source d'Unix est distribué librement aux universités. En conséquence, UNIX a gagné la faveur de la communauté scientifique et universitaire. Il a été ainsi à la base des systèmes d'exploitation des principales universités, dont l'Université de Californie à Berkeley.

- **En 1976 :**

Sortie de "Unix Time-Sharing System Version 6" aux Bell Labs d'AT&T, AT&T peut enfin commercialiser le logiciel.

En marge, Les laboratoires Bell d'AT&T développent UUCP (Unix to Unix Copy Program). Il s'agit du premier protocole d'échanges de données largement disponible et qui sera énormément utilisé avant l'avènement de TCP/IP et d'Internet.

- **En 1977 :**

Première expérience de portage d'UNIX sur un autre type d'ordinateur, l'Interdata 8/32, par Ken Thompson, Dennis Ritchie et Steve Johnson.

- **En 1978 :**

Mais à partir de 78 vont apparaître de nouveaux Unix, basés sur les sources d'Unix Time-Sharing System V6. Le premier d'entre eux étant réalisé au sein de l'université de Berkeley le 1BSD, une nouvelle distribution d'UNIX réalisée à l'université de Berkeley principalement par Bill Joy (l'actuel n°2 de Sun Microsystems).

- **En 1979 :**

Sortie de "Unix Time-Sharing System V7" aux Bell Labs d'AT&T. Cette version est la première à disposer, en standard d'UUCP. Elle tournait sur PDP/11 et VAX.

C’est aussi l’année de l’annonce de AT&T de commercialiser UNIX. Ceci a aussi incité l’université de Californie à Berkeley à créer sa propre variante : BSD UNIX

- **En 1980 :**

Microsoft commercialise Xenix OS, un UNIX portable pour machines à base de Intel 8086, Zilog Z8000 et Motorola M68000.

Apollo lance une ligne de stations de travail hautes performances, basées sur le processeur Motorola 68000 et optimisées pour le travail graphique. Ce type de stations aura un grand succès dans le domaine de la CAO et du calcul numérique.

Onyx présente son ordinateur Onyx C 8002 équipé d'un processeur Z8000, de 256 Ko de Ram, d'un disque dur, d'un lecteur de bandes, de 8 ports série et tournant sous Unix pouvant accueillir 8 utilisateurs pour 20000 $. Il s'agit du premier micro ordinateur tournant sous Unix.

- **En 1981 :**

Fondation de la société Silicon Graphics Incorporated (SGI) par James Clark. L'Unix System Group (USG) d'AT&T publie Unix System III, issu du développement de CB UNIX 3.0 (Columbus Unix de Bell labs) et UNIX 32V, deux saveurs d'UNIX d' AT&T.

- **En 1982 :**

Création de la société Sun Microsystems par Andy Bechtolsheim, Vinod Khosla et Scott Mc Neally, (L'actuel patron de Sun Microsystems) tous étudiants à Stanford, pour commercialiser une station de travail.

Le matériel a été initialement développée par Andy Bechtolsheim à Stanford et connecté sur le réseau de l'université : le Stanford University Network ou SUN (mais le nom SUN se voulait aussi un clin d'oeil au premier fabricant de stations de travail : Apollo)

Rapidement, Bill Joy, développeur de l'Unix de l'Université de Berkeley, va rejoindre la société pour s'occuper de la partie logicielle et participe à la création de SunOS 1.O, dérivé de 4.1BSD.

La société SUN commercialise la station de travail Sun 1 équipée d'un microprocesseur 68000, tournant sous Unix et munie d'origine d'une interface Ethernet et de TCP/IP.

- **En 1983 :**

AT&T met en vente la version commerciale du célèbre Unix Système V.
. L'Université de Berkeley distribue une nouvelle version de son Unix BSD 4.2 incluant d'origine le protocole TCP/IP.

- **En 1984 :**

Silicon Graphics commercialise sa première station de travail Unix avec moteur 3D intégré. Le MIT commence à développer le System X Window, un logiciel permettant de gérer l'affichage graphique des stations de travail Unix. Plus qu'une simple interface graphique, il s'agit d'un système client-serveur évolué capable par exemple de gérer plusieurs écrans sur une même machine ou d'afficher sur l'écran d'une machine distante.

- **En 1986 :**

Le MIT publie la première version de son environnement graphique pour station Unix : X v10.4, qui conduira à X11R6, à la base de XFree86 le serveur d'affichage sous linux.

- **En 1987 :**

Diffusion de X Window, une interface client/serveur graphique développée au MIT (Massachussetts Institute of Technology).

La version Unix système V release 3 d'AT&T est opérationnelle . C'est la version qui a forcé, les principaux constructeurs à développer un OS propriétaire , HP (HP-UX) se basant sur Unix système III, et IBM (AIX), se basant sur Unix système V release 2.

Sortie de la version 4.3 de BSD. Et finalement, c'est cette année là qu'AT&T et Sun ont choisi conjointement d'unifier le System V et BSD.

Sortie de Minix basé sur "Unix Time-Sharing System Version 7", développé par Andy Tanenbaum, qui inspira, les débuts de linux.

- **En 1990 :**

Sortie du System V release 4 d'AT&T, comportant de nouveaux standards d'unification d'UNIX. C'est le résultat de la coopération entre Sun et AT&T. Cependant, d'autres grands constructeurs (en particulier DEC, HP et IBM) se sentant menacés par cette collaboration entre deux des plus grands développeurs d'Unix ont décidé de créer l'OSF(Open Software Foundation).

Sortie de Solaris 1.0, également connu sous le nom de SunOS 4.1.1. Le Perl, langage de programmation écrit par Larry Wall spécifiquement pour les besoins de gestion d'Unix s'est grandement répandu. Tandis que le C est le langage de choix pour la programmation système d'Unix, le Perl est le langage pour la gestion de systèmes Unix.

- **En 1991 :**

Un étudiant finlandais, nommé Linus Torvalds, a acheté un micro-ordinateur de type PC, afin d'étudier, la programmation du microprocesseur i386 (ancêtre du pentium). Ne voulant pas être limité par MS/DOS, il a tout d'abord utilisé Minix un clone d'Unix pour x86. Conscient des limitations de Minix, il décide de lui apporter quelques améliorations. Chemin faisant, il propose à certains internautes de le rejoindre dans son projet.

Pour cela il lance le mail qui suit sur la Mailing-List de Minix (en Août 1991) :

Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-) Linus (PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

C'est le départ de L'aventure linux, qui vous mène aujourd'hui ici. De nombreux développeurs de part le monde se joignent à lui et Linux 0.01 est disponible en Septembre de la même année.

Encouragé par un ami et parce qu'il utilisait un compilateur GPL (gcc) pour son Linux, Linus Torvalds décide de mettre le code source de Linux sous licence GPL : tout le monde peut alors participer au développement de Linux.

- **En 1992 :**

Sun Lance Solaris 2.0 (également appelé SunOS 5.0) basé sur SunOS 4.1.1 (dérivé de 4.1 BSD) et Unix System V release 4, avec gestion des threads.

Sortie de SLS, la première distribution Linux.

Sortie de XFree86 1.0m, une implémentation libre de X11R5 (le serveur d'affichage mis au point par le MIT). Xfree86, aujourd'hui en version 4.2 est le serveur d'affichage de la plus part des unices libres.

- **En 1993 :**

Novell a acheté Unix System Laboratories (USL) d'AT&T propriétaire d'UNIX et il a donné des droits de la marque déposée "UNIX" à X/Open.

Sortie de Slackware la première distribution commerciale de Linux, par Patrick Volkerding qui en est toujours le principal (seul ?) développeur. Aujourd'hui en version 8.1. C'est avec elle qu'a débuté l'écrasante majorité des linuxiens de la première heure.

Ian Murdock, lance le projet Debian, qui aboutira à la distribution communautaire par excellence : Debian GNU/Linux, soutenue dans ses débuts par la Free Software Foundation. Elle est aujourd'hui en version 3.0 et supporte 11 types de processeurs différents.

- **En 1994 :**

Sortie de Linux 1.0, un système UNIX complet, capable d'exécuter X Window, TCP/IP, Emacs, UUCP, le courrier électronique et les news Usenet, entre autres . Pratiquement tous les programmes libres importants ont été portés sous Linux, et on commence à voir apparaître des applications commerciales.

Création de RedHat software par Bob Young, très active dans le monde du logiciel Libre et qui maintient la distribution Linux la plus titrée et connue. Elle est aujourd'hui en version 7.3.

Le projet XFree86 rejoint le Consortium X.

- **En 1995 :**

Création de l'apache Apache Group à qui ont doit le serveur web le plus utilisé, d'après les plus récentes études de netcraft apache serait utilisé par 60 % des serveurs web.

- **En 1996 :**

Sortie de MkLinux la version microkernel de linux, faisant tourner un serveur basé sur linux 1.3, au-dessus d'un micro-noyau Mach.

- **En Juin 1996 :**

Sortie de Linux 2.0. Attention, contrairement à tous les protagonistes ici (hormis UNICS ???), linux a été codé de A à Z, sans copié une ligne de code à un OS. Il est néanmoins compatible BSD et System V et respecte POSIX 1 et 2.

Lancement du projet KDE, se basant sur QT de Trolltech, destiné à doter les Unices d'une interface utilisateur conviviale.

- **En 1997 :**

Sun lance solaris 2.6 alias "SunOS 5.6". Linux fait son apparition dans les systèmes embarqués (téléphones cellulaires, navettes spatiales).

Lancement du proget Gnome par Miguel de Icaza, en réaction à KDE qui se basait sur QT. En effet, QT avait à l'époque une licence qui empêchait son utilisation dans un système libre.

- **En 1998 :**

Sun lance solaris 7 (après solaris 2.6) alias "SunOS 5.7" (vous n'avez pas suivi ? relisez le texte en entier ;-))

La société Netscape libère le code source de son navigateur, sous une licence spéciale. C'est ce projet qui conduira 4 ans plus tard à Mozilla 1.0 un navigateur très respectueux des standards sur le web.

Début de Google, qui deviendra le moteur de Recherche le plus utilisé et le plus performant du web, il utilise Linux.

Sortie de KDE 1.0 aujourd'hui en version 3.0.2.

Adoption massive de Linux dans le milieu professionnel, avec de grands noms comme : Oracle, Sun, Informix, HP. Selon IDC Linux représente 25 % des serveurs d'entreprises, après seulement 4 ans d'existence.

Lancement de la distribution Mandrake Linux (5.1) et création de Mandrakesoft par : Gaël Duval, Frédéric Bastok et Jacques Le Marois. C'est probablement l'une des distributions qui a fait le plus progresser Linux sur le bureau de "monsieur tout le monde".

- **En 1999 :**

Sortie de linux 2.2.0. La première expo de LinuxWorld se tient à San Jose, en Californie. Lancement du site Sourceforge par VA Linux, site accueillant tous les projets qui font du logiciels libres.

Sortie de Gnome 1.0

- **En 2000 :**

Sortie de Security Enhanced Linux 1.0 basée sur linux 2.2.13, distribution "sécurisée" par la NSA.

- **En 2001 :**

Sortie de Linux 2.4.0. IBM investi 1 Milliard de $ dans le développement de Linux. Sortie du kit playstation pour Linux. De nombreux jeux à succès sont portés sous Linux (Quake 3, Unreal Tournament ...)

- **En 2002 :**

Sortie de KDE 3.0 et Gnome 2.0, des bureaux complets et conviviaux. Les contrats OEM se multiplient, pour livrer des machines avec Linux préinstallé. C'est aussi l'année de WineX2 qui permet de faire fonctionner, la plus part des jeux DirectX sous Linux.

Sortie de Mozilla 1.0 après 4 ans de développement acharné. C'est également l'année d'Apache 2 et Xfree86 4.2 basé sur X11R6. De nombreuses Entreprises (IBM, HP, Sun ...), portent leurs technologies sous Linux et livrent des serveurs sous Linux en lieu et place de leurs Unices respectifs.

- **En 2003 :**

Sortie de Linux version 2.6, sortie en décembre 2003.

Aujourd'hui, près d'1 serveur sur 3 est sous Linux. La version stable la plus récente de Linux est la 2.6.16.13 au 2 mai 2006.

- **Conclusion :**

On peut dénombrer trois grandes familles de système Unix.

les BSD et ses dérivés (FreeBSD, OpenBSD...);

les Unix System V (dont l'éditeur américain SCO, qui les tenait de Novell, qui lui-même les tenait du propriétaire initial AT&T, est l'actuel possesseur des droits - et le fait d'ailleurs savoir par ses revendications) et ses dérivés (HP-UX, AIX) ;

les "hybrides" comme Solaris (d'abord basé sur BSD, maintenant plus proche de System V) et Linux (un cas particulier, plutôt proche de System V mais compatible BSD).

A noter qu'Unix est une marque déposée, dont seuls certains OS peuvent se prévaloir officiellement : AIX, HP-UX, Solaris entre autres.

http://solutions.journaldunet.com/0402/040205_unix_histoire.shtml
http://trustonme.net/didactels/?rub=145

Présentation de Mandrakelinux 10.1 et installation

La distribution utilisée:

Sortie officielle le 27 Octobre 2004

Mandrakelinux Official est la branche du système d'exploitation dédiée aux utilisateurs qui réclament un système éprouvé. La version 10.1 Official apporte un support matériel inégalé ainsi qu'une intégration avancée de technologies mobiles, telles que Bluetooth et WIFI. Les autres apports clés incluent GNOME 2.6, une gestion améliorée des ordinateurs portables, et de nouveaux compilateurs pour une performance étendue. Les passionnés de nouveautés et de fonctionnalités inédites n'ont pas besoin de chercher plus loin.

Mandrakelinux 10.1 utilise les composants logiciels majeurs suivants :

- Linux Kernel 2.6.8 (inclut différents fixs de la version 2.6.9rc)
- Xorg 6.7.0
- KDE 3.2.3
- GNOME 2.6
- Glibc 2.3.3, GCC 3.4.1
- Apache 2.0.50, PHP 4.3.8
- MySQL 4.0.18, Samba 3.0.6
- Mozilla 1.7.2, GIMP 2.0.4
- OpenOffice.org 1.1.3

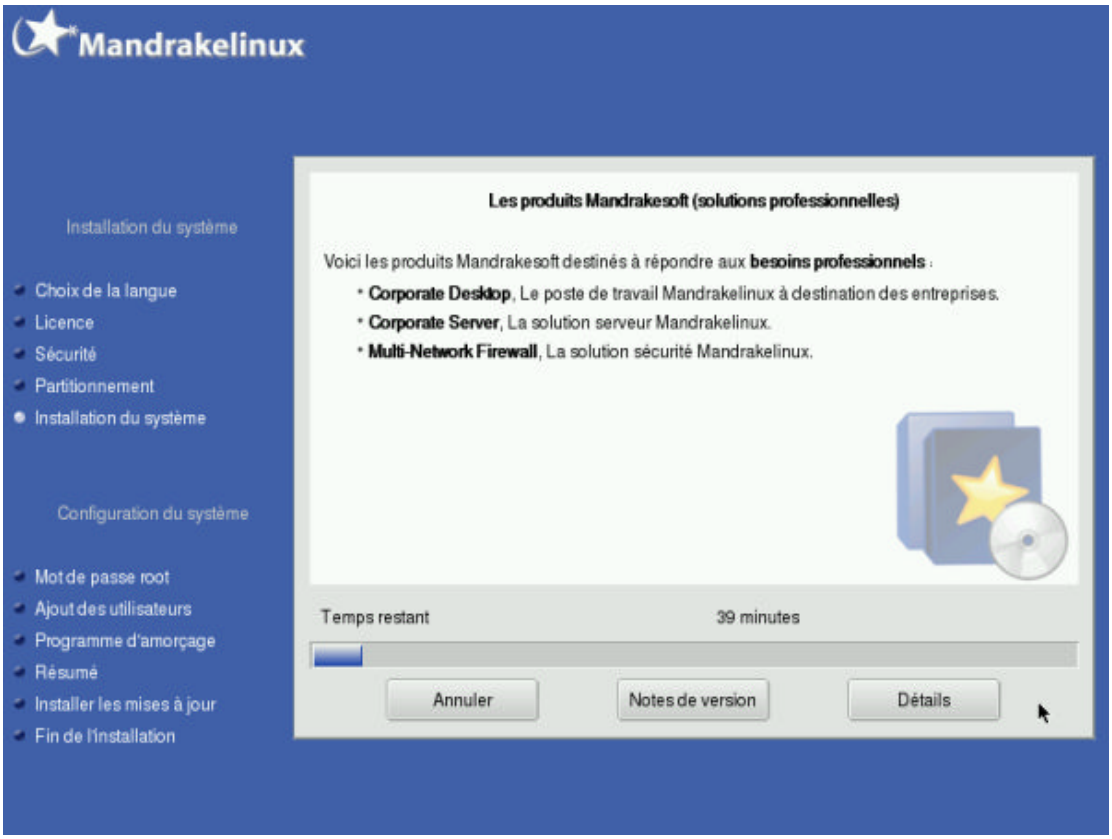
Mandrakelinux 10.1 est optimisé pour les processeurs Pentium ou plus récents (et compatibles), et ne fonctionne donc pas sur les architectures plus anciennes.

Une procédure d'installation en grande partie automatisée

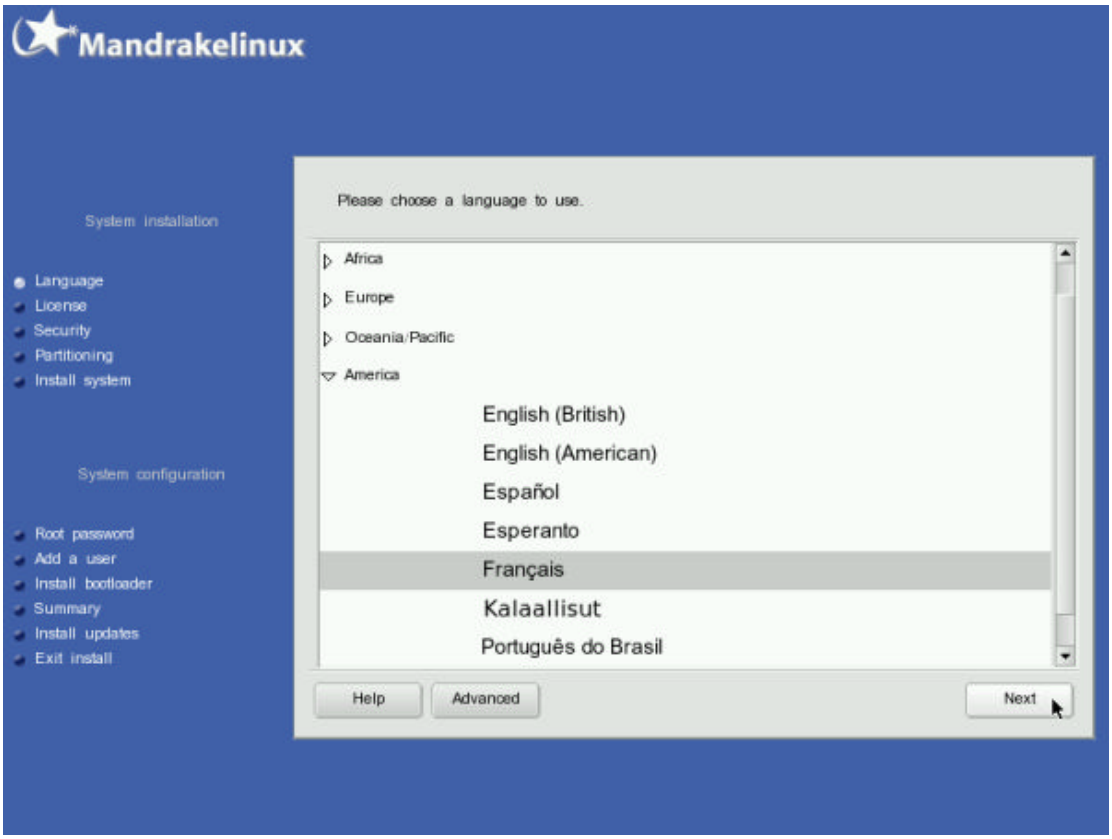
La procédure d'installation Mandrakelinux faisait déjà figure de référence dans le monde Linux, elle est aujourd'hui l'une des plus simples et des plus puissantes disponibles. Des fonctionnalités avancées comme le redimensionnement de partitions NTFS ou la configuration automatique des imprimantes sont intégrées à l'installation.

- L'installeur graphique dispose d'une interface moderne et cohérente, de nouvelles couleurs et des polices lissées.
- La procédure d'installation par défaut est très simplifiée. Les différentes phases sont très intuitives, et la plupart des matériels sont automatiquement reconnus et configurés. A chaque étape de l'installation, les utilisateurs avancés peuvent accéder à des options de réglage plus fines.
- Il est possible de procéder à une installation orientée poste de travail ou serveur en fonction des groupes de logiciels sélectionnés.
- L'installation peut s'effectuer par diverses méthodes, dont la connexion réseau, le CD ou le DVD.
- L'installeur Mandrakelinux est disponible en 60 langues.
- En plus de permettre le redimensionnement de partitions NTFS, la procédure d'installation offre de nombreuses fonctionnalités uniques comme l'existence de plusieurs systèmes de fichiers (dont les systèmes de fichiers journalisés Ext3, ReiserFS, XFS, et les systèmes de fichiers cryptés), la possibilité de configurer des partitions RAID et le redimensionnement de partitions MS-Windows FAT32.
- Plusieurs systèmes de fichiers réseau sont disponibles : NFS, SMB et WebDAV.
- L'outil d'auto-installation est très commode pour dupliquer des installations serveur et poste de travail.
- Un "mode secours" très simple à utiliser est disponible en cas d'imprévu.

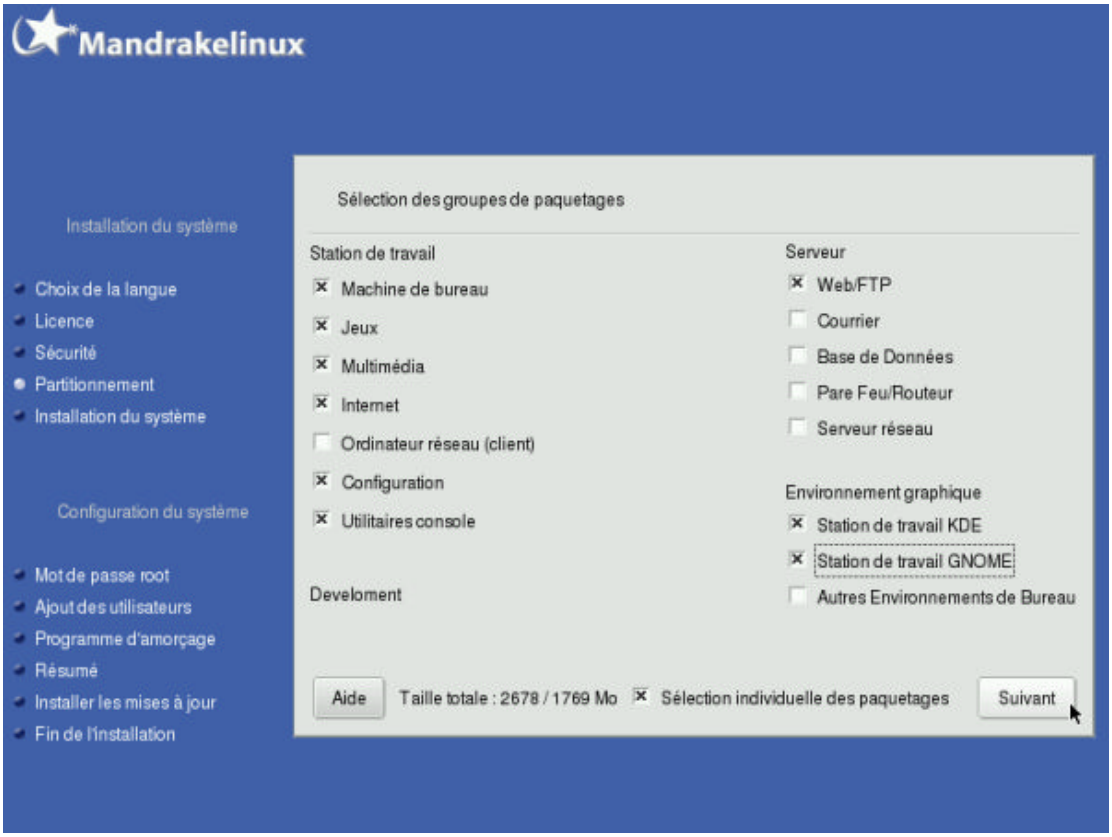
L'installation a été simplifiée et est largement automatisée.



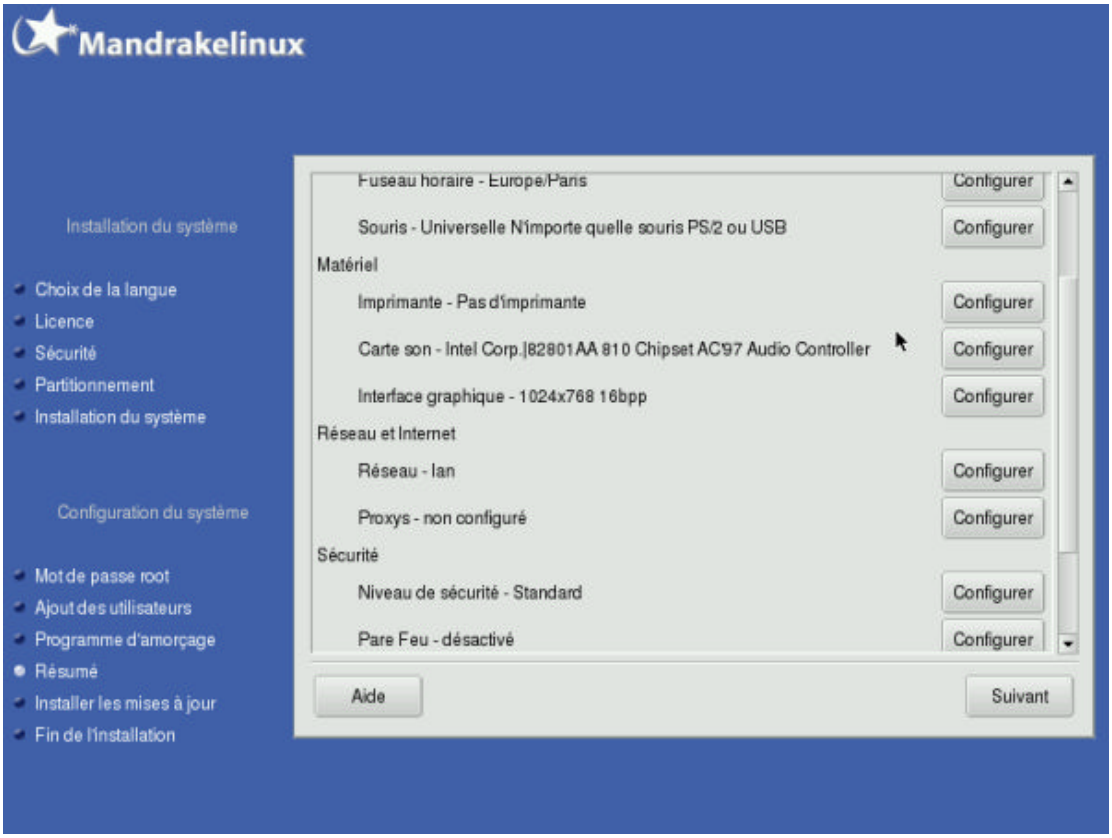
La procédure d'installation est disponible en un grand nombre de langues.



Faites votre choix : desktop, serveur... ou les deux ?



Toutes les options de configuration sont résumées et peuvent être modifiées à la fin de l'installation.



1 PRINCIPES DE BASE ET DEFINITIONS

1.1 LINUX/UNIX/POSIX

LINUX est un système 32 bits, respectant la norme **POSIX** (*Portable Operating System Interface uniX*). Cette norme est imposée pour tout système voulant appartenir au monde UNIX, c'est à dire capable de faire "tourner" des applications UNIX. Plus précisément, pour être POSIX, il faut être capable de compiler des applications développées pour UNIX.

1.2 Système d'exploitation

Un système d'exploitation est un ensemble de programmes et de sous-programmes qui ont pour rôle de gérer et de piloter le matériel. C'est l'intermédiaire entre les programmes d'application et le matériel.

Un OS digne de ce nom n'acceptera jamais qu'une application fasse appel directement au matériel.

UNIX/LINUX met ses fonctionnalités à la disposition des programmes d'application sous forme d'appels système (en langage C).

1.3 Rôle d'un système d'exploitation

Du fait de sa position d'intermédiaire entre les demandes et le matériel, voici les tâches qui sont du ressort de l'OS :

- la gestion de la mémoire ;
- intercepter tous les accès à des périphériques. Aucun programme ne peut accéder

directement à une interface série ou une unité de disquette. L'accès aux périphériques passe, sous UNIX/LINUX, par des fichiers de périphérique (/dev, et bientôt le système de fichier DevFS) ;
- gestion correcte de la mémoire disque. Dans ce domaine, UNIX/LINUX offre un système de gestion de fichiers hiérarchique ;
- gestion du fonctionnement des programmes (processus) ;
- gestion des droits d'accès sur les fichiers ;
- gestion des logs du système et des applications.

1.4 Structure du système d'exploitation LINUX

Il faut distinguer deux notions souvent mélangées : **noyau** (*kernel*) et **distribution**.

LINUX correspond à proprement parler au noyau d'un système d'exploitation de type UNIX (noyau MONOLITHIQUE). Ceci signifie qu'il s'agit des fondations du système définissant le type de système de fichiers ainsi que la gestion des périphériques.

A ce noyau se rattache certains modules.

Le noyau et ses modules constituent le **système d'exploitation** proprement dit.

Une distribution LINUX est un système d'exploitation comportant le couple noyau + modules sur lequel ont été greffés des utilitaires afin de constituer un système complet et convivial.

Il existe différents organismes (RedHat, Mandrake, SuSE, Debian, ...) qui proposent des distributions complètes : interface pour l'installation, gestionnaires de fichiers, éditeurs en tout genre, platine CD, outils de développement, utilitaires de retouche d'images...

1.5 LINUX en tant que système multi-utilisateurs, multi-tâches et multi-plateformes

- **multi-tâches** : plusieurs tâches peuvent être effectuées en même temps ;

- **multi-utilisateurs** : on distingue le super-utilisateur (= root = administrateur) qui a tous les droits sur tous les fichiers, et les utilisateurs ayant des droits plus limités ;

- **multi-plateformes** : Intel, Motorola (Apple MAC et Amiga), Sun, Sparc, DEC, Alpha, ...

1.6 LINUX et le monde du libre

LINUX est libre (ce qui n'est pas forcément synonyme de gratuit). Cela signifie que les codes sources sont accessibles et modifiables par qui veut. Vous pouvez par exemple récupérer le noyau de LINUX par FTP (ftp://ftp.kernel.org/pub/) et vous créer votre propre distribution.

1.7 Les licences des logiciels libres

GNU (*Gnu is Not Unix*), est un projet (http://www.fsf.org/home.fr.html) initialisé par **Richard Stallman** en 1984. Ce projet a pour but de définir le monde du logiciel libre (appelé aussi "**Open Source**"). En d'autres termes, tout logiciel libre appartient au projet **Gnu**.

Open Source : Il ne s'agit pas d'une licence mais seulement d'une définition des droits de l'utilisateur.
Cette définition sert de base à des licences : GPL, BSD, ...
Selon cette définition, l'utilisateur doit pouvoir copier le logiciel, le vendre ou donner ses copies sans avoir à reverser la moindre contribution. Il faut aussi que le code source soit inclus et LISIBLE.
La licence BSD autorise la privatisation d'une version modifiée.
La licence GPL ne le permet pas.

GPL : Elle répond à toutes les spécifications de l'**Open Source**.
Toutes modifications d'un programme doivent être distribuées sous licence GPL.
Par contre il est interdit d'utiliser du code **Open Source** pour la création d'un logiciel propriétaire.

LGPL : Elle permet l'inclusion d'un programme sous licence LGPL dans une application propriétaire.
Un programme sous LGPL peut migrer en GPL mais pas l'inverse !

BSD, X, Apache : Sont des licences radicalement différentes de la GPL et LGPL.
Ces licences permettent de conserver secrètes des modifications apportées à un logiciel.

Attention : appartenir au monde du libre ne veut pas dire GRATUIT !!!!
Il ne faut pas confondre avec **FreeWare**, **ShareWare** qui sont des licences commerciales !

1.8 Les distributions LINUX

Pour une description complète je vous renvoie à l'HOWTO suivant :
http://www.freenix.org/unix/linux/HOWTO/Distribution-HOWTO.html

Liste non exhaustive des distributions complètes :

- RedHat (EU)
- Mandrake (France)
- Debian (EU)
- Suse (Allemagne)
- Slackware
- Caldera
- Executive Linux (France)
- Corel Linux OS
- Storm Linux (basée sur la Debian) (Canada)
- BestLinux 2000 (basée sur la RedHat) (Finlande)
- WinLinux, PhatLinux (on les installe et on les lance depuis Windows)
- LinuxPPC pour iMac DV avec MacOS 8.6
- SuSe 6.3 Beta version PPC pour iMac

Liste des mini-distributions :
http://www.linux-center.org/fr/distributions/mini-distributions/

Tomsrtbt: Une distribution ``poids plume'' (1 disquette, 1722Mb). Elle sera présentée dans le chapitre « Réparation d'un système de fichiers ».
Trinux: Une micro-distribution qui s'amorce à partir de disquettes ou d'une partition Windows et qui dispose d'un ensemble complet d'outils de sécurité réseau.
Linux Router Project: Une mini-distribution de LINUX centrée sur les réseaux. Elle tient sur une seule disquette et permet de créer facilement des routeurs, des serveurs de terminaux et d'autres applications embarquées.
Yard: Un système qui permet de se fabriquer des disquettes de boot selon des spécifications personnelles.
Dragonlinux: Distribution ``poids plume'' (2 disquettes, 20 Mo de disque dur sous DOS).
Xdenu: Une véritable petite distribution très finalisée qui comprend un serveur X.
DosLinux: Une petite distribution qui peut être installée sur un système Dos/Win9x.
LOAF (Linux on a floppy): Une distribution qui tient sur une disquette incluant tous les utilitaires de base pour le réseau : ifconfig, route, ping, traceroute, telnet, ssh et lynx.
LEM Linux: LEM LINUX est une distribution légère de LINUX (< 11 Mo) pour les applications embarquées.
Alfalinux: Slackware Linux sur 2 disquettes.
ShareTheNet: Routeur sous LINUX avec un setup sous Windows.

1.9 Le Shell

Le Shell est un utilitaire qui sert d'intermédiaire entre la saisie des commandes et le noyau du système.
Au fil des années toute une série de Shell a été développée. Le plus connu et le plus répandu dans le monde UNIX est le Bourne-Shell (/bin/sh), d'après le nom de son inventeur, **Steve Bourne**. Il existe depuis le milieu des années 70.
Il fut suivi de toute une série d'autres Shell, destinés aux diverses variantes d'UNIX. Il s'agit entre autres, du C-Shell (/bin/csh) et du Korn-Shell (/bin/ksh). Ce dernier remplacera certainement à terme le Bourne-Shell sur les plates-formes UNIX standards.
Sous BSD, le Shell de base est CSH. Alors que LINUX propose le BASH.
Le CSH a une syntaxe proche du C. Il est considéré comme un Shell puissant mais plus difficile à utiliser que le BASH (petit fils du Bourne Shell). Les spécialistes recommandent d'utiliser soit TCSH (CSH avec complétion) soit le ZSH qui se rapproche de KSH.

2 PREMIERE INSTALLATION

Principes de l'installation :
- Les types d'installation :
Pour démarrer l'installation, il faut "booter" un système qui déclenche l'exécution du script d'installation.
Ceci se fait à partir d'une unité "bootable" : disquette, CD-ROM ou d'un serveur d'installation sur le réseau.
Le système chargé en mémoire crée un système de fichiers root en mémoire, dans un disque virtuel en RAM (RAM disk) et démarre l'exécution de la procédure d'installation.
A l'issue de cette phase, un nouveau noyau est généré et le système Linux est installé.
Il reste à réaliser la post-installation, dont la création des comptes utilisateurs et l'installation des progiciels spécifiques.

La localisation du chargeur LILO
- sur une disquette
- comme chargeur principal dans le MBR ("Master Boot Record").
- dans la partion Linux.

- Disquette d'installation :
Dans le cas d'un lecteur de CD-ROM non bootable, il faut fabriquer une disquette d'amorçage.
Cette disquette de démarrage contient LILO, un noyau Linux avec un nombre minimum de pilotes de périphériques et un système de fichiers root contenant l'essentiel pour réaliser l'installation.
Dans le cas d'une installation réseau, vous devez générer à partir du CD-ROM Linux, une disquette de démarrage spéciale, que l'on baptisera "bootnet".
Dans le cas o`u vous installez Linux à partir d'un CD-ROM PCMCIA, il vous faut en plus de la disquette de démarrage, générer une disquette supplémentaire que l'on baptisera "pcmcia".
- La création des disquettes:
Pour fabriquer les disquettes, on utilise la commande **rawrite** du répertoire **\dosutils** (dans le CD-ROM d'install) **si l'on opère à partir de Windows** ou la commande **dd à partir de Linux**.
- En pratique :
La création des disquettes d'installation sous Windows :
D:\dosutils\rawrite
Enter disk image source file name: \images\boot.img
Enter target diskette drive: a:
Idem pour l'image bootnet.img
La création des disquettes d'installation sous Linux :
Montez le CD-ROM : mount -t iso9660 /dev/cdrom /mnt/cdrom
Allez dans le répertoire de montage : cd /mnt/cdrom
Insérer une disquette vierge et exécutez la commande : dd if=images/boot.img of=/dev/fd0 bs=1440k
Idem pour la disquette d'install réseau : dd if=images/bootnet.img of=/dev/fd0 bs=1440k
Pour comprendre et réaliser une installation de LINUX via le réseau ,voir chapitre 3

Nous allons partir avec l'hypothèse suivante (car classique!) :
Vous avez un seul disque dur sur lequel vous avez installé Windows.

Pour installer LINUX, il faut lui faire de la place sur votre disque dur.
C'est la fameuse étape de partitionnement du disque.

Il va falloir créer au minimum 3 partitions (nous verrons plus loin qu'il faut en fait plus de 3 partitions pour des raisons de sécurité, mais pour le moment c'est suffisant. Attention tout de même, pour des raisons historiques un disque dur ne peut pas contenir plus de 4 partitions primaires. En fait si on veut plus de 4 partitions, on fait 3 partitions primaires + 1 partition étendue contenant autant de partitions logiques que l'on veut).
La première partition de votre disque doit absolument être celle de Windows.
Retenez que Windows est un système "qui n'en veut", il aime être le premier!
LINUX quant à lui n'est pas exigeant sur la place de la partition sur le disque.
Enfin ceci n'était pas vrai avant la RedHat 6.2 : car il fallait alors que la partition bootable ne se trouve pas au delà du 1024 ième cylindre.
Par contre il est nécessaire de créer 2 partitions pour LINUX :
- une partition "LINUX NATIVE" pour le système LINUX ;

- une partition "SWAP" qui servira de zone de mémoire temporaire (équivalent à la RAM) pour LINUX.
Attention la SWAP doit être comprise entre 16Mo et 127Mo (en tout cas pour les anciennes distributions).
En général on lui alloue le double de la RAM (ou mieux 2,5 fois la RAM...mais nous y reviendrons)

Pour une installation plus professionnelle :
Les partitions à créer :
/boot : de 10 à 20Mo : elle contient les fichiers de "boot LILO" et le noyau Linux et doit résider en deçà du cylindre 1023 de votre disque
/ : de 50 à 100Mo : elle contient le système de fichiers principal de Linux
/usr : de 300Mo à 1 voire 2 Go : elle contient les commandes et les services pour les utilisateurs. C'est de loin la plus volumineuse
/var : min 50Mo : elle est le pendant de "/usr". Elle contient les requetes adressées aux services des utilisateurs et le fichiers historiques (logs)
/tmp : sa taille dépend du nombre de services installés. On peut l'estimer à environ 50Mo. Elle contient des fichiers temporaires créés par des commandes et des fichiers. La création spécifique d'une partition pour /tmp évite la saturation de la partition root, o`u /tmp réside sinon, par des fichiers temporaires trop gros.
/home : sa taille dépend du nombre d'utilisateurs et de leur consommation d'espace disque.
/usr/src : sa taille est d'au moins 30Mo. Elle contient les sources du noyau et des paquetages. Elle est indispensable pour regénérer un noyau personnalisé.
swap : de 2 à 2,5 fois la RAM.

3 INSTALLATION DE LINUX VIA RESEAU

L'installation réseau peut être réalisée à partir d'un serveur **NFS** (UNIX, LINUX) ou d'un serveur **FTP/HTTP** (Site Internet ou privé).

Ce type d'installation nécessite que vous ayez une carte réseau d'un type très répandu, c'est le cas par exemple des cartes 3com, car le programme d'installation ne connait qu'un nombre restreint de cartes réseau.

3.1 Configuration d’un serveur NFS

Il suffit de monter le CD-ROM Linux et de partager ensuite le répertoire de montage du CD-ROM.

L'exemple qui suit est le partage du CD-ROM à partir d'un système Linux.

```
#mount -t iso9660 /dev/cdrom /mnt/cdrom

#echo "/mnt/cdrom (ro)" >> /etc/exports
```

Il faut ensuite arrêter les services "rpc" qui gèrent le partage et les redémarrer.

#cd /etc/rc.d/init.d/

#./rpc stop

#./rpc start

3.2 Installation à partir du serveur d'un réseau

Démarrer l'installation à partir de la disquette de **"bootnet"**, comme une installation locale, et sélectionnez "NFS image" comme type d'installation dans le menu qui propose les différents types d'installation réseau.

Que vous utilisiez un serveur NFS, un serveur FTP ou un serveur HTTP, la première partie de l'installation est commune aux 3 méthodes.

*** Partie commune aux 3 types d'installation (NFS, FTP, HTTP) :**

Démarrez l'installation à partir de la disquette d'installation réseau "bootnet", comme pour une installation locale.

On doit, par la suite, sélectionner la carte réseau parmi les cartes disponibles, puis choisir l'option [Autoprobe] pour que le programme d'installation détermine automatiquement les paramètres "IRQ" et "I/O Address" de la carte ou l'option [Specify] si vous souhaitez renseigner les paramètres vous meme.

Si la carte n'est pas détectée, vous ne pourrez pas continuer l'installation, sinon, l'écran suivant vous propose de configurer l'adresse IP de votre machine manuellement [Adresse IP statique], [BOOTP] via un serveur BOOTP, ou [DHCP] via un serveur DHCP.

3.3 Installation à partir d'un serveur NFS

Vous etes maintenant invité à entrer le nom ou l'adresse IP de votre serveur NFS, ainsi que le répertoire o`u se trouve la distribution RedHat.

Saisissez ces deux informations, et l'écran suivant vous indique que l'installation se poursuit normalement, ou vous affiche un message d'erreur en cas de problème.

A titre d'exemple, si sur le serveur NFS, le CD-ROM d'installation ou une image du CD sur le disque dur est monté dans le répertoire /install_reso, vous devez saisir "/install_reso" pour la rubrique "Chemin du répertoire RedHat".

3.4 Installation à partir d'un serveur FTP

Vous êtes maintenant invité à entrer le nom ou l'adresse IP du site FTP, ainsi que le répertoire o`u se trouve la distribution RedHat. Saisissez ces deux informations dans tous les cas.

A titre d'exemple, si sur le serveur FTP, la distribution RedHat se trouve dans le répertoire /public/linux_redhat, vous devez saisir "/public/linux_redhat" pour la rubrique "Chemin du répertoire RedHat".

Si votre serveur n'accepte pas de connexions anonymes, le cas d'un réseau privé par exemple, ou si vous etes dans un réseau local o`u les accès au reste du monde passent par une machine proxy et que le serveur FTP n'est pas dans votre réseau local, vous devez activer l'option [Utilisation d'un FTP non anonyme ou d'un service mandataire (proxy)]

Si vous avez activé l'option [Utilisation d'un FTP non anonyme ou d'un service mandataire (proxy)], vous devrez saisir :

- Le login/mot de passe pour le site FTP

- L'adresse IP de la machine mandataire (proxy), ainsi que le numéro de port du service proxy, normalement 80 ou 21.

3.5 Installation à partir d'un serveur HTTP

Idem FTP.

Processus de Boot

- Une fois l'ordinateur mis sous tension, le processeur lance l'exécution du **BIOS** (*Basic Input Output System*).

Le rôle du BIOS consiste à analyser la configuration matérielle de l'ordinateur et la déclarer apte pour le service.
<=> Recensement des périphériques + Test de certains périphériques :
Test de la mémoire,
Test de la présence de clavier, ...
Test de la présence des disques durs, lecteurs de CDROM IDE
Assignation d'une IRQ pour le contrôleur SCSI

- Localisation de l'OS par le BIOS : suivant la configuration du BIOS, il va rechercher l'OS sur : disquette, CDROM, Disque dur, réseau

Le BIOS lit les 512 premiers octets du périphérique d'amorçage :
- soit premier secteur (MBR = *Master Boot Record*) sur un disque dur ;
- soit le secteur d'amorçage sur une disquette.

Problème : le noyau LINUX a une taille > 1000 fois 512 octets (soit 500Ko).
=> le MBR ne contient bien sûr pas le noyau, mais seulement un chargeur qui va chercher le véritable système sur le disque et le charger en mémoire.

Ce secteur d'amorçage contient donc un petit programme et une table des partitions. Le programme ou "boot loader", se charge de désigner la partition active du média. L'OS possède son propre programme qui est un peu plus complexe que le précédent. Ce programme lié à l'OS va indiquer la partition active, mais également la partition d'amorçage et l'activer. Sous LINUX, ce programme "évolué" est en général LILO (*Linux Loader*).

Suivant votre installation LINUX, il y a 2 cas possibles : soit vous avez installé LILO(*Linux Loader*) dans le MBR, soit il est installé au début de votre partition LINUX.

Dans le premier cas, c'est donc LILO qui sera directement exécuté, sinon il s'agira d'un programme qui va charger la première partition active du disque dur (/root) et répéter le processus d'amorçage : lire les 512 premiers octets de cette partition et exécuter ce programme.

A ce stade (soit par le BIOS, soit par le MBR), LILO a été chargé en mémoire.
Tout LILO ?
NON!

En fait, le programme LILO fait 5Ko, ce qui est 5 fois trop pour le MBR. Pour pallier à ce problème, LILO est scindé en 2 parties : la première faisant moins de 512 octets étant chargée de lancer la deuxième partie qui chargera le noyau.

- Le prompt LILO s'affiche, on sélectionne l'OS

  En fait ces 4 lettres s'affichent chacune après une étape bien précise :
  - **L** : après le chargement de la première partie de LILO ;
  - **I** : après le chargement de la deuxième partie de LILO, mais avant son exécution ;
  - **L** : après le démarrage de la deuxième partie de LILO ;
  - **O** : lorsque la deuxième partie de LILO a trouvé sur le disque tous les éléments nécessaires à son bon fonctionnement.

- Le chargeur LILO (deuxième partie) va lire le contenu de **/boot/boot.b** afin de savoir quelle image doit être lancée (/boot/boot.b est un fichier créé à partir de lilo.conf quand on tape /sbin/lilo).

Note : Si au démarrage de votre ordinateur, celui-ci affiche juste « LI » et s'arrête, vous savez qu'il y a un problème pour l'exécution de la deuxième partie de LILO. Ce problème est en général dû, soit à un problème de configuration de la géométrie du disque dur, soit le fichier /boot/boot.log a été supprimé ou renommé.

- Une fois totalement et correctement chargé en mémoire, la première chose que fait LILO est de regarder si une des touches [Shift], [Ctrl] ou [Alt] est enfoncée. Si tel est le cas (ou bien si l'option « prompt » est spécifiée dans le fichier de configuration de LILO (**/etc/lilo.conf**)), celui-ci affichera en plus la chaîne « **boot:** » pour indiquer qu'il attend une action de la part de l'utilisateur.
A ce moment, il est possible d'obtenir la liste des différents noyaux LINUX ou autres systèmes d'exploitation amorçables par LILO en tapant sur [Tab]. Le choix correspondant doit être tapé au clavier; sinon au bout d'un délai d'expiration, c'est le choix par défaut qui sera sélectionné.
A la suite du nom du noyau, l'utilisateur peut également entrer des paramètres supplémentaires pour modifier le comportement du système.
Voici les principaux paramètres utilisés :

boot: Nom-Noyau root=/dev/hdaX : changer la partition racine par défaut.
boot: Nom-Noyau mem=128M : forcer l'utilisation dune certaine quantité de mémoire.
boot: Nom-Noyau ether=0,0,eth1 : forcer l'auto-détection d'une seconde carte réseau Ethernet.

Par défaut, LINUX arrête la recherche des cartes réseaux dès que la première est trouvée.
Il est également possible de spécifier directement les paramètres de la carte :
ether=11,0x400,eth1 : la deuxième carte réseau a pour IRQ « 11 » et utilise la zone d'entrée/sortie située à l'adresse 0x400.

boot: Nom-Noyau init=/bin/bash : spécifie un programme de démarrage autre que **/sbin/init**. On peut aussi utiliser un autre Shell en tapant **init=/bin/sh**. Enfin on peut spécifier directement le niveau d'exécution à transmettre au programme **init** :
single
ou **un chiffre de 1 à 3**.
Exemple :
boot: Nom-Noyau init 3
boot: Nom-Noyau single qui équivaut à **boot: Nom-Noyau init 1** ou encore **-s**.

- Une fois que LILO sait quelle image il doit charger, il va lire dans **/boot/map** pour avoir les adresses des blocs du disque dur qui contiennent le noyau à charger. Pendant ce temps s'affiche **"Loading Linux..."**

- Il y a décompression de l'image auto-extractible : **/boot/vmlinuz-XXX**. On voit le message "**Uncompressing Linux**"

- Le noyau LINUX démarre : **"OK, booting the kernel"**

Les différente étapes du processus de démarrage du noyau de LINUX sont les suivantes (décrites dans /arch/i386/kernel/head.S) :
Initialisation en fonction de l'architecture matérielle

Mise en place de la gestion des interruptions
Initialisation de la mémoire virtuelle
Initialisation des interruptions, du planificateur (le "scheduler") et de l'horloge
Traitement des arguments de la ligne de commande
Chargement des modules
Module : *partie du noyau mais qui n'est pas directement intégré à celui-ci. Vous pouvez aller faire un tour dans /**lib/modules/**. Vous trouverez des fichiers avec l'extension **.o**. Ce sont les modules compilés. L'intérêt des modules est qu'ils sont chargés en mémoire quand le besoin s'en fait sentir et déchargé quand on en a plus besoin. Cela permet d'économiser de la mémoire.*

En fait, on a recours à un ramdisk initial. Celui-ci comporte un mini-système de fichiers contenant quelques fichiers d'initialisation, mais surtout les modules indispensables, et compressés à l'intérieur d'un fichier (**/boot/initrd-XXXX.img**). C'est LILO qui va charger ce fichier à partir du disque dur en même temps que le noyau, de sorte que celui-ci dispose des pilotes nécessaires pour démarrer correctement.
Typiquement, lorsqu'il doit utiliser un ramdisk, LINUX le monte comme un système de fichiers racine et charge ainsi les modules indispensables. Puis, le ramdisk est abandonné et c'est alors le VRAI système de fichiers racine qui est monté à sa place (en mode « lecture uniquement » car on ne sait pas encore s'il contient ou non des erreurs). Le message qui s'affiche alors est :
VFS: Mounted root (ext2 filesystem) readonly.

- A ce stade, le noyau libère l'espace utilisé précédemment pour la détection et la configuration des périphériques : **Freeding unused kernel memory...**

- Ensuite, le programme **/sbin/init** est lancé par le noyau (cf. suite). A partir de ce moment là, le noyau en a fini avec l'initialisation du système. C'est **init** qui va se charger de la suite des opérations...

- Les 2 premiers processus démarrés se nomment **"idle"** et **"init"** et portent les numéros d'identification 0 et 1. La seule tâche effectuée par **idle** consiste à consommer le temps processeur qui n'est pas exploiter par les autres processus.
Idle se voit aidé dans sa tache par le **scheduler** (=**l'ordonnanceur**) qui répartie en permanence le temps processeur entre tous les processus. Et si d'aventure, aucun des autres processus ne demande de temps processeur, l'ordonnanceur s'en remet à **idle** pour occuper la machine.
Init se révèle beaucoup plus intéressant qu'idle. Ce processus s'avère le parent de tous les processus présents sur le système (sauf du singulier idle).
Rappel : les processus du système sont classés selon une arborescence dont chaque branche définit un processus possédant obligatoirement un parent et éventuellement un ou plusieurs fils.
Le processus **init** effectue un contrôle des partitions puis procède au montage du système de fichier principal sous /.

- Après, le processus **init** fera la distinction entre les différents niveaux d'exécution (**runlevels**)
/sbin/init = le premier processus système dont la tâche essentielle consiste à lancer les scripts de démarrage du système.
LINUX suit la norme System V, c'est à dire qu'il exécute tout d'abord **/etc/rc.d/rc.sysinit**.
rc.sysinit est un script qui va terminer la configuration de base du système : mettre l'horloge du système à l'heure du CMOS, charger la configuration clavier adéquate, activer la partition de swap, configurer le nom de la machine, configurer la variable d'environnement PATH,
Ce script vérifie notamment que le système de fichier racine ne comporte pas d'erreur et le remonte en lecture-écriture. Et enfin active les modules nécessaires en appelant **/etc/rc.d/rc.modules**.

- Puis **/sbin/init** continue avec les scripts de services du niveau d'exécution en cours (les **runlevels**).
A chaque mode d'utilisation de LINUX correspond un **runlevel**. Pour cela, **/sbin/init** va lire le fichier **/etc/inittab**. Ce script fait lui-même appel au script **/etc/rc.d/rc** flanqué d'un paramètre, pour définir le niveau d'exécution. Par exemple, dans le cas d'un niveau d'exécution 3, c'est la commande **/etc/rc.d/rc 3** qui sera exécutée.

Les 7 niveaux classiques :
0 : halte (Ne pas mettre initdefault égal à cette valeur)
1 : mode mono-utilisateur (principalement pour le dépannage)
2 : mode multi-utilisateurs avec gestion du réseau mais sans gestion NFS
3 : mode multi-utilisateurs complet
4 : pas utilisé sous LINUX
5 : mode graphique (X11) -> login graphique tel que **xdm** ou **kdm**
6 : redémarrage (ne pas mettre initdefault égal à cet valeur)

Les différents niveaux d'exécution peuvent être gérés via une interface : **tksysv** (accessible dans le control-panel).

A chaque **runlevel** correspond un répertoire **/etc/rc.d/rc.X/** contenant des liens symboliques sur des services à démarrer (présents dans **/etc/rc.d/init.d**/).
Dans chaque répertoire **/etc/rc.d/rc.X**/, les scripts commençants par **K** servent à arrêter les services du niveau d'exécution. Ceux qui commencent par **S** servent à démarrer les services. Les numéros qui suivent déterminent l'ordre d'exécution.

- Suivant la configuration du fichier **/etc/inittab**, le processus de boot peut aller jusqu'au niveau 5. C'est à ce runlevel qu'est démarré le serveur X Window.

Notes :

1)Démarrage en mode graphique :
Dans /etc/inittab,
changer la ligne : **id:3:initdefault:** en **id:5:initdefault:**
et vérifier la présence de la ligne :
x:5:respawn:/etc/X11/prefdm -nodaemon à la fin.
Les préférences utilisateur se trouvant dans **/etc/sysconfig/desktop** (KDE, GNOME,....).

2) Vous devez taper **/sbin/init** qu'après chaque changement du fichier /etc/inittab pour que ces changements soient pris en compte.

- Enfin... ouf ! c'est au tour du script **/etc/rc.d/rc.local** d'être lancé. C'est dans ce script que vous ajoutez des scripts ou toute autre commande que vous désirez lancer au démarrage de la machine.

Système de Fichiers

Table des matières

1 TYPES DE FICHIERS ET STRUCTURATION

UNIX/LINUX offre un système de fichiers hiérarchique

La base du système de fichiers est le fichier individuel.
En d'autres termes, dans le monde UNIX/LINUX, **TOUT est FICHIER**.

On distingue plusieurs types de fichiers :
- les fichiers normaux (*ordinary files*) ;
- les répertoires (*directories*) ;
- les fichiers spéciaux (*devices*) ;

Les fichiers normaux contiennent soit des textes en clair (courriers, textes sources des programmes, tableaux, ...), soit un programme exécutable. Dans ce dernier cas, on parle également de fichier binaires (*binary files*). Pour obtenir des informations sur le contenu d'un fichier nous utiliserons la commande `file`.
Les fichiers spéciaux (*devices*) représentent les interfaces avec les périphériques que gère l'OS.

Le point de départ de la hiérarchie de fichiers est le répertoire racine (noté /). Ce répertoire racine donne naissance à des sous-répertoires.

2 FICHIERS, CHEMINS ET ARBORESCENCE

2.1 Nomenclature des fichiers

Dans tous les cas, UNIX/LINUX fait la différence entre minuscules et majuscules.
On dit que UNIX/LINUX est sensible à la casse.

Si vous envisagez d'échanger vos fichiers avec d'autres systèmes d'exploitation non UNIX, ne dépassez pas les 8 + 3 caractères (8 pour le nom et 3 pour l'extension <=> format DOS).
Le caractère - ne doit surtout pas être utilisé au début d'un nom de fichier car de nombreuses commandes tenteraient d'interpréter ce nom comme une option!

2.2 Nomenclature des chemins d'accès

Chemin absolu : pris à partir de la racine.
Exemple : `/home/kmaster/DOCS/LINUX/cours.sdw`

Chemin relatif : pris à partir du répertoire courant.
Exemple : `~/DOCS/LINUX/cours.sdw`

3 GESTION DES REPERTOIRES

Chaque utilisateur reçoit de l'administrateur système un répertoire dans lequel il aboutira chaque fois qu'il se connectera (*home directory*).

3.1 Se déplacer dans les répertoires

cd [(Chemin absolu ou relatif) nom du répertoire] : change de répertoire
Aller directement dans le répertoire de l'utilisateur: **cd ~**

3.2 Savoir où on est dans l'arborescence

pwd : où suis-je dans l'arborescence ?

3.3 Lister les fichiers

ls [options] [noms_des_fichiers_ou_répertoires] : liste le contenu d'un répertoire
ls -l : Idem mais donne le maximum d'informations :

```
Type-Fichier Droits Compteur-Lien Nom_propriétaire Nom_groupe Taille_octet Mois Jour Heure
```

Dans l'ordre :
Le type de fichier : -, d ou l => d=directory, - = fichier, l = lien symbolique
Les droits:
il y a 9 caractères:
- les 3 premiers : droits du propriétaire du fichier ;
- les 3 suivants : droits du groupe ;
- les 3 derniers : droits des autres utilisateurs (= le reste du monde).

ls -d : affiche uniquement le répertoire.

D'autres options :

-a : Permet d'afficher tous les fichiers commençant par un point (=fichiers cachés).
-F : Ajoute un slash derrière les répertoires.
-s : Affiche la taille des fichiers.
-i : Afficher le numéro d'index (i-noeud = inode) de chaque fichier à gauche de son nom.
-r : Inverser le tri du contenu des répertoires.
-R : Afficher récursivement le contenu des sous-répertoires.
-t : Trier le contenu des répertoires en fonction de la date et non pas en ordre alphabétique.
Les fichiers les plus récents sont présentés en premier.

Exemples :

ls -lsa : donne l'apparence d'un DIR sous DOS.
D'autres variantes :
ls -rtl (comme la valise ;->)
ls -lisa
ls -arlt : liste les fichiers et répertoire selon la date de création

3.4 Créer un répertoire

mkdir [options] [nom répertoire] : crée un répertoire

Options :
-p : Crée les répertoires parents si nécessaire
-m : Donne des droits d'accès spécifiques

Exemples :
$ls /home/kmaster
Desktop IMAGES PROGS nsmail office52

$mkdir -p /home/kmaster/DOCS/COURS-LINUX/
=> Création du répertoire DOCS puis du répertoire COURS-LINUX

3.5 Supprimer un répertoire vide

rmdir [options] [nom répertoire] : détruit un répertoire (sans récursivité).

Options :
--ignore-fail-on-non-empty : Ignore les erreurs de répertoires non vides.
-p : Destruction des parents s'ils sont vides.

Exemples :
$ls /home/kmaster/DOCS/COURS-LINUX/
Cours.sdw StructureFHS.jpg TEST
$rmdir TEST/
rmdir: TEST: Le répertoire n'est pas vide.
$ rmdir --ignore-fail-on-non-empty TEST/
$ ls /home/kmaster/DOCS/COURS-LINUX/
Cours.sdw StructureFHS.jpg **TEST**

3.6 Supprimer un répertoire non vide

rm –rf [nom répertoire]

Attention : Soyez très prudent lorsque vous utilisez rm -rf ; en effet, cette commande supprime de façon irrémédiable un répertoire et son contenu en une opération.

4 GESTION DES FICHIERS

4.1 Copier un fichier

cp [options] [(Chemin absolu ou relatif) fichier origine] [(Chemin absolu ou relatif) fichier destination]
ou
cp [options] [(Chemin absolu ou relatif) fichier origine] [(Chemin absolu ou relatif) répertoire de destination]

Options :

-d : Préserve les liens vers d'autres fichiers.
-i : Questionne l'utilisateur.
-f : Supprime le fichier de destination s'il existe avant la copie.

Gardez à l'esprit que si le fichier cible existe déjà, il sera purement et simplement écrasé, sans aucun message d'avertissement. **Soyez donc très prudent en matière de copie de fichiers**.

4.2 Renommer ou déplacer un fichier

La commande **mv** (*move*) remplit ces deux fonctions.
mv [options] [(Chemin absolu ou relatif) fichier origine] [(Chemin absolu ou relatif) fichier destination]
ou
mv [options] [(Chemin absolu ou relatif) fichier origine] [(Chemin absolu ou relatif) répertoire de destination]

Options :

-b: Créer une copie de sauvegarde.
-i : Mode interactif.
-f : Force la suppression de la destination si elle existe avant le déplacement.

Exemple :

Déplacement d'un répertoire complet : c'est comme pour un fichier!

```
$pwd
        /home/kmaster/COURS-LINUX
$ls
        Cours.sdw  StructureFHS.jpg  TEST
$mv TEST/ ..
$cd ..
$ls
        COURS-LINUX TEST
```

Renommer un répertoire : c'est comme pour un fichier!

```
$ls
        Cours.sdw  StructureFHS.jpg  TEST
$mv TEST/  TEST-NEW/
$ls
        Cours.sdw  StructureFHS.jpg  TEST-NEW
```

4.3 Supprimer un fichier

rm [options] [fichier] : détruit un fichier.

Options :
-i : Mode interactif.
-f : Pas de message à l'écran.
-r : Suppression récursive.

Exemples :
rm -f [fichier] : détruit un fichier sans vérification du bien fondé (par une question posée à l'utilisateur.
rm -rf [répertoire] : détruit un répertoire avec récursivité.

La suppression de fichiers est définitive !
Il est IMPOSSIBLE de récupérer des fichiers effacés par erreur.
C'est pourquoi je conseille de prendre l'habitude d'utiliser l'option **-i** qui demande confirmation.

4.4 Créer un lien sur fichier

Dans certaines circonstances, vous voudrez ouvrir aux autres utilisateurs l'accès à vos propres fichiers sans pour autant leur fournir une copie de ces fichiers. Ils pourront accéder au même fichier que vous, mais sans indiquer systématiquement le chemin d'accès complet. Dans cette situation, vous utiliserez la commande **ln** (*link*), permettant d'attribuer à un fichier un deuxième nom.

ln [options] fichier1 fichier2

Options :
-b : Créer une copie de sauvegarde.
-f : Force la liaison.
-i : Mode interactif.
-s : Créer un lien symbolique.

Exemple complet :

```
$ cd TEST/
$ vi fichier-originel.txt
$ ls -li
total 2
            31873 drwxrwxr-x    2 kmaster  kmaster      1024 sep 10 17:25 COMMUN
            29883 -rw-rw-r--    1 kmaster  kmaster         5 sep 10 17:24 fichier-originel.txt

$ ln fichier-originel.txt COMMUN/fichier-commun.txt
$ ls -li
total 2
            31873 drwxrwxr-x    2 kmaster  kmaster      1024 sep 10 17:25 COMMUN
            29883 -rw-rw-r--    2 kmaster  kmaster         5 sep 10 17:24 fichier-originel.txt

$ ls -li COMMUN/
total 1
            29883 -rw-rw-r--    2 kmaster  kmaster         5 sep 10 17:24 fichier-commun.txt
```

On peut vérifier que les deux noms désignent le même fichier grâce au numéro (**29883**) qui s'inscrit devant la ligne correspondante. Ce numéro correspond au numéro d'index (i-noeud = inode) du fichier. La notion d'inode sera abordée un peu plus loin.

$ rm fichier-originel.txt
$ ls
COMMUN
$ ls -li COMMUN/
total 1
29883 -rw-rw-r-- 1 kmaster kmaster 5 sep 10 17:24 fichier-commun.txt

4.5 Liens symboliques

ln -s [(path)source-du-lien] [(path)destination = nom du lien] : crée un lien symbolique entre deux fichiers.

Exemple :

$ln -s /dev/cdrom /dev/dvd : crée un lien symbolique nommé 'dvd' dans /dev. Ce lien pointant sur le device /dev/cdrom

$ ls -l /dev/dvd

lrwxrwxrwx 1 root root 10 sep 3 19:07 /dev/dvd -> /dev/cdrom

Le type de fichier correspondant à un lien symbolique est indiqué par la lettre « **l** » dans la première colonne du résultat de la commande **ls -l**. La flèche suivant le nom indique en outre à quel fichier ce nom fait référence. La taille d'un tel lien correspond à la longueur du chemin auquel il fait référence.

4.6 Déterminer le type d'un fichier

Même si le résultat de la commande **ls -l** montre qu'un fichier normal est précédé du signe « **-** », cela ne signifie pas obligatoirement que vos pourrez afficher son contenu à l'écran. Il peut très bien s'agir d'un fichier binaire (= programme compilé). Pour connaître le type de fichier, utilisez la commande **file** qui est très pratique avant de faire un **cat** ou un **more** sur un fichier inconnu.

file [options] [fichier]

Options :

-L : Analyse des liens.
-z : Tenter de déterminer le type d'un fichier compressé.

4.7 Concaténation et affichage du contenu d'un fichier

cat :
* concaténer des fichiers.
* afficher en une seule fois le contenu d'un fichier sur la sortie standard (écran par défaut).

Options :

-b : Numéroter toutes les lignes (sauf les lignes vides).
-n : Numéroter toutes les lignes sans exception.
-s : Supprimer toutes les lignes vides successives en trop.

5 ORGANISATION DU SYSTEME DE FICHIERS

Les systèmes LINUX dans un souci de compatibilité et de portabilité, respectent une seule norme : la norme FHS (*File Hierarchy Standard*).

5.1 Hiérarchie de base d'un système conforme à la FHS

/ : Répertoire racine = base du système de fichiers.
/boot : Fichiers statiques du chargeur de démarrage. On y trouve entre autre l'image compressée du noyau (vmlinuz). + System.map + ... (à définir !)
/bin : Ce répertoire contient les commandes employées par tous les utilisateurs (root et autres). On y trouve par exemple les commandes sh-> bash, ls, rm, cp, ...
/sbin : Les commandes pour la gestion du système. On y trouve par exemple la commande adduser permettant d'inscrire de nouveaux utilisateurs. Ce sont en fait les binaires nécessaires à root.
/lib : Bibliothèques partagées essentielles au système lors du démarrage. Elles sont requises par /bin et /sbin. Les autres bibliothèques qui sont requises par /usr/bin ou /usr/sbin se trouvant dans /usr/lib.
/lib/modules : Les modules du noyau.
/etc : Ce répertoire contient les fichiers nécessaires à l'administration (passwd, group, inittab, syslog.conf, ...) ainsi que les scripts de démarrage (/etc/rc.d/).
/etc/X11 : Fichiers spécifiques à la configuration de X Window (XF86Config, ...).
/etc/opt : Contient les fichiers de configuration des applications installées dans /opt.
/etc/rc.d/init.d : Contient tous les démons lançables au démarrage du système.
/tmp : Certaines commandes génèrent des données temporaires au cours de leur travail. Ces données sont supprimées lorsque la commande est terminée.
/dev : Ce répertoire contient les fichiers spéciaux (périphériques = *devices*).
/var : Répertoire de données variables.
/var/cache : Données de cache des applications (httpd, man).
/var/lock : Fichiers de verrouillage.
/var/log : Fichiers de log.
/var/opt : Données variables du répertoire /opt.
/var/run : Fichiers relatifs aux processus en cours. (xinetd.pid, ...).
/var/spool : Données mises en attente pour les applications.
/var/mail -> /var/spool/mail/ : Courriers des utilisateurs.
/var/spool/lpd/ : On trouve ici les fichiers intermédiaires et de configuration pour la gestion de l'impression.
/var/tmp : Contient les fichiers temporaires.
/var/yp : Fichiers de base de données NIS (Network Information Service = Yellow Pages).
/usr : Arborescence secondaire. On y trouve quantité d'utilitaires, de bibliothèques, de manuels, ...
/usr/bin : Contient des fichiers binaires pour les utilisateurs (at, cc, cpp, dir, du, eject, find, ...)
/usr/sbin : Contient d'autres fichiers binaires pour le super-utilisateur (chroot, crond, httpd, kudzu, ntsysv, ...)
/usr/lib : Bibliothèques partagées.
/usr/libexec : Commandes exécutées par d'autres commandes (sftp-server, ...).
/usr/X11R6 : Réservé à X Window
/usr/games : Contient des fichiers binaires de jeux.
/usr/include : Réservé aux programmes C qui y incluent des fichiers headers (.h).
/usr/local : Hiérarchie locale, souvent utilisée pour des programmes compilés par l'administrateur sur la machine.
/usr/local/bin : Binaires des programmes locaux (mplayer, dvdinfo, ...).
/usr/local/games : Binaires des jeux locaux.

/usr/local/include : Fichiers d'en-tête C et C++ locaux (dvd_reader.h, ...).
/usr/local/lib : Bibliothèques partagées locales (libdvdread.so).
/usr/local/sbin : Binaires systèmes locaux.
/usr/local/share : Hiérarchie indépendante (videolan, ...).
/usr/local/src : Fichiers sources locaux.
/usr/share : Réservé aux données non dépendantes de l'architecture (aspell, emacs, vim, ...).
/usr/share/dict : Contient un fichier words -> linux.words proposant une liste de mots.
/usr/share/doc : Documentation sur le système et les programmes.
/usr/share/games : Fichiers statiques concernant /usr/games
/usr/share/info : Répertoire principal du système info de Gnu (sous forme de *info.gz).
/usr/share/locale : Contient les informations utiles au programme perl nommé locale.
/usr/share/man : Les pages de manuel.
/usr/share/misc : Contient des données diverses indépendantes de la machine.
/usr/share/terminfo : Contient les répertoires de base de données terminfo (bbd décrivant les terminaux, utilisée par tous les programmes orienté « affichage »).
/usr/share/zoneinfo : Contient les informations relatives à la configuration de la zone horaire.
/usr/src : Contient des fichiers de codes source du noyau.
/home : Hébergement de répertoires personnels.
/mnt : Point de montage de systèmes de fichiers temporaires (CDROM, disquette, Windows, ...).
/opt : Packages d'applications supplémentaires.
/root : Répertoire de l'administrateur root.
/proc : Ce répertoire et son contenu n'existent que si l'option /proc file system a été choisie lors de la compilation du noyau en cours d'utilisation. Ce système de fichier est dit « virtuel » car il est créé à chaque démarrage de la machine et, malgré la taille que vous pouvez lire pour chacun des fichiers, il n'occupe aucun espace sur le disque dur.
Tous les répertoires dont les noms sont des nombres correspondent aux processus en cours. Dans chacun de ces répertoires, vous trouverez entre autre la ligne de commande du processus, ses variables d'environnement ... Les autres répertoires donnent des infos sur la machine et le système d'exploitation. La plupart des infos qui s'y trouvent sont accessibles par l'intermédiaire de programmes ou de commandes telles que free, top ou mount.
Vous pouvez essayer d'utiliser la commande cat sur /proc/cpuinfo et vous verrez s'afficher comment LINUX a reconnu votre processeur (type, fréquence, capacité du cache, bugs connus, ...). La même commande sur le fichier /proc/interrupts affiche la liste des interruptions (IRQ) et à quel périphérique elles sont affectées. Jetez aussi un coup d'oeil sur /proc/pci pour voir tout ce qui est relié au bus PCI.

6 L'IMPLEMENTATION DU SYSTEME DE FICHIERS

6.1 Stockage intermédiaire des entrées/sorties

Le transfert des données depuis ou vers le disque dur est relativement lent en comparaison de la vitesse de traitement en mémoire.
C'est pourquoi toutes les entrées/sorties de données dans le système de fichiers (inodes, répertoires, fichiers normaux) sont stockées de façon intermédiaire. Une zone de la mémoire est prévue à cet effet, elle est réservée et accessible uniquement au système d'exploitation.

6.2 La procédure de stockage intermédiaire

Si des données sont écrites dans un fichier ou lues à partir d'un fichier, le transfert avec le disque dur n'est pas entrepris pour chaque caractère ou chaque bloc. Toutes les entrées/sorties atterrissent dans une mémoire tampon (*buffer cache*). Lors de la lecture d'un fichier, à chaque essai d'accès à un bloc de ce fichier, le système va vérifier si ce bloc n'existe pas déjà dans la mémoire tampon. Si c'est le cas, il n'y aura pas d'accès disque. Si par contre le bloc n'existe pas en mémoire tampon, la lecture du bloc est lancée et le noyau en profitera pour effectuer d'autres tâches.
Dès que le bloc est chargé dans la mémoire tampon, le gestionnaire de périphériques le signale sous forme d'interruption. Le noyau du système d'exploitation met ensuite ces données à la disposition du programme qui en a demandé la lecture. Le bloc de données est d'abord transféré du disque dur vers la mémoire tampon, puis copié dans le processus appelant. Même si cela peut sembler un double travail, cette mémoire tampon et ce stockage intermédiaire évitent dans bien des cas des accès disques.
Cette procédure est également appliquée dans l'autre sens, avec les sorties. Il arrive souvent qu'un processus qui vient de modifier un bloc ait à nouveau besoin de celui-ci. Le stockage intermédiaire évite alors un accès disque en écriture. Lorsque la mémoire tampon est pleine, ou après un laps de temps défini, elle est transférée sur le disque dur. Les données du disque dur sont ainsi actualisées; on parle dans ce cas de synchronisation. L'administrateur dispose de la commande **sync** pour transférer le tampon sur le disque sans délai.

6.3 Avantages et inconvénients du stockage intermédiaire

Les avantages de cette procédure sont essentiellement des gains de temps lors des accès aux données sur le disque dur ou autre support de stockage.

L'inconvénient est que pour que le gain soit optimum, il s'agit de n'écrire effectivement sur disque les données qu'à grands intervalles de temps. Cela implique que les données dans le tampon sont plus actuelles que celles du disque dur. Seule la mise en rapport de données du disque dur avec celles de la mémoire (via les commandes **sync** et **flush**) permet de créer un ensemble cohérent. Ainsi, si votre machine plante avant la synchronisation, vous pouvez perdre des données que vous avez pourtant sauvées au niveau applicatif, mais trop récemment pour qu'elles aient été effectivement écrites sur le disque.

7 PETIT RAPPEL SUR LA RELATION SWAP/RAM

Lorsqu'il n'y a pas assez de RAM pour tous les processus, les pages mémoires les moins utilisées sont stockées temporairement sur le disque, dans la zone d'échange (appelée SWAP). Ceci afin de laisser la place aux processus qui en ont besoin.
Cette technique permet de disposer d'une ''mémoire virtuelle'' qui peut être sensiblement plus grande que la ''mémoire physique'' (RAM).
Il est à noté qu'un accès disque est 1000 fois plus lent qu'un accès RAM. Ainsi quand les processus utilisent l'ensemble de la RAM, les performances du système se dégradent rapidement.
Pour déterminer l'usage réel de la mémoire, on peut utiliser les deux commandes suivantes : **free** ou **top**. Celles ci donnent la quantité de mémoire physique utilisée par rapport à la quantité disponible, la taille des zones de mémoire cache et mémoire virtuelle utilisées.

Pour la swap, il est courant de respecter la règle suivante :

Taille de swap = 2,5 * taille RAM

En effet : 1 * la taille de la RAM pour la copie de la mémoire
+ 1 * la RAM pour swaper
+ 0.5 pour les échanges entre les deux.

8 LA FRAGMENTATION

Deux types de fragmentation :

8.1 Fragmentation interne

La fragmentation interne consiste en une perte d'espace sur le dernier bloc alloué sur le disque. Par exemple sur un support de masse divisé en secteurs de 512 octets, un fichier de 513 octets occupera deux secteurs. Soit une perte de 511 octets.
Classiquement la taille d'un bloc est de 1Ko.
La commande **df -T -i** permet de déterminer la perte due à la fragmentation interne.

8.2 Fragmentation externe

Il s'agit là de la dispersion des blocs alloués à un fichier sur le disque dur.
La fragmentation externe cause 2 problèmes :
- un fichier mettra d'autant plus de temps à être parcouru qu'il est dispersé sur le disque ;
- il risque de manquer d'espace libre pour faire des allocations de blocs contiguës. Ceci n'est pas seulement théorique. En effet, certains fichiers nécessitent une allocation contiguë.

En principe : plus les fichiers à stocker seront grands, plus il faut employer une taille de blocs élevée lors du formatage des partitions. La taille est forcément un multiple de 512 octets. Avec une granularité (1 granule = bloc de 512 octets) plus grande, comme par exemple 64Ko, des fichiers de grande taille seront parcourus plus rapidement. Le revers de la médaille consistant en un accroissement important de la fragmentation interne pour les petits fichiers.
La parade classique : partitionner l'espace disque afin que les gros fichiers ne figurent que sur des partitions de haute granularité. Il faut cependant savoir qu'avec Ext2fs, il faut éviter de mettre autre chose que 1 Ko comme taille de bloc car cela peut engendrer des bugs.
En lecture, on constate que les transferts de données sont plus rapides avec des blocs contigus.
Le système Ext2fs est plus optimisé dans la gestion de l'allocation (/ à la perte de l'espace) que les autres systèmes (FAT de Microsoft et FFS d'Unix).

Manipulation de fichiers

Table des matières

1 AFFICHER

Sous Unix, il existe de nombreuses commandes spécialisées qui permettent d'afficher sur la sortie standard le contenu d'un fichier.

Par combinaison de ces commandes au sein de tubes, on peut former des commandes complexes très utiles.

1.1 Afficher le contenu d'un fichier (cat)

La commande `cat` permet d'afficher sur la sortie standard l'entier contenu d'un fichier. En voici les options :
- `v` (Verbose) permet de convertir les caractères spéciaux des fichiers binaires en caractères affichables
- `n` (Number) numérote les lignes
- `b` (number nonBlank) numérote que les lignes non vides
- `E` (show Ends) affiche le symbôle `$` (dollard) à la fin de chaque ligne
- `s` (Squeeze blank) n'affiche au plus un ligne vide
- `T` (show Tab) affiche les caractères tabulations comme `^I`
- `A` (show All) équivalent à -vET
- `e` équivalent à -vE
- `t` équivalent à -vT.

Syntaxe : `cat` *`fichier`*

Exemple : `cat` *`lettre.tex`*

L'exemple précédent affiche à l'écran le contenu du fichier *lettre.tex*. Si le fichier avait été binaire, l'affichage des caractères spéciaux aurait provoqué un grave disfonctionnement du terminal d'affichage.

Exemple : `cat -v` *`/bin/ls`*

L'ajout de l'option v permet de transformer les caractères de contrôle en caractères affichables à l'écran. Ceci est très utile pour rechercher des chaînes de caractères à l'intérieur d'un programme.

A l'origine, `cat` permet de concaténer des fichiers et d'en renvoyer le résultat sur la sortie standard.

Syntaxe : `cat` *`fichier_1 fichier_2 fichier_3`*

Exemple : `cat` *`lettre.txt rapport.txt conclusion.txt`*

L'exemple précédent affiche à l'écran le contenu des fichiers *lettre.txt*, *rapport.txt* et *conclusion.txt*.

Au lieu d'afficher sur la sortie standard le résultat de cette concaténation, on peut la rediriger vers un fichier.

Syntaxe : `cat` *`fichier_1 fichier_2 fichier_3 > fichier_4`*

Exemple : `cat lettre.txt rapport.txt conclusion.txt > publication.txt`

Ainsi le fichier *publication.txt* aura pour contenu la concaténation des fichiers *lettre.txt*, *rapport.txt* et *conclusion.txt*.

1.2 Affichage inverse (tac)

La commande `tac` est homologue à `cat` mais affiche le contenu d'un fichier en partant de la dernière ligne vers la première.

1.3 Afficher l'entête (head)

La commande `head` permet de n'afficher que les permières lignes d'un fichier (10 par défaut). En voici les options :
- `cN` affiche les N premiers octets
- `nN` affiche les N premières lignes
- `q` n'affiche pas le nom du fichier
- `v` affiche le nom du fichier avant d'en afficher l'entête.

Syntaxe : `head fichier`

Exemple : `head lettre.tex`

Dans cet exemple, il s'affiche les 10 premières lignes du fichier *lettre.tex*, si ce dernier en contient moins de 10, il sera affiché dans sa totalité.

Exemple : `head -n5 lettre.tex`

Affichage des 5 premières lignes.

Exemple : `head -vc20 lettre.tex`

Affichage des 20 permiers caractères après le nom du fichier.

1.4 Afficher la fin (tail)

La commande `tail` permet de n'afficher que les dernières lignes d'un fichier (10 par défaut). Elle est homologue à `head` et possède les mêmes attributs.

Syntaxe : `tail` *`fichier`*

Exemple : `tail` *`lettre.tex`*

Dans cet exemple, il s'affiche les 10 dernières lignes du fichier *lettre.tex*, si ce dernier en contient moins de 10, il sera affiché dans sa totalité.

Exemple : `tail -n5` *`lettre.tex`*

Affichage des 5 dernières lignes.

Exemple : `tail -vc20` *`lettre.tex`*

Affichage des 20 derniers caractères après le nom du fichier.

1.5 Afficher écran par écran (more et less)

Les commandes `more` et `less` permettent d'afficher page par page des fichiers volumineux sur la sortie stantard.

Pour passer à la page suivante : les touches fléchées. Pour effectuer un défilement vertical : touche ENTREE. Pour quitter : touche q.

1.6 Numérotation des lignes (nl)

La commande `nl` permet l'affichage du contenu d'un fichier et en numérote les lignes. En voici les options :
- `bt` numérote les lignes non-vides (par défaut)
- `ba` numérote toutes les lignes
- `bpXXX` numérote seulement les lignes qui contiennent la chaîne de caractères XXX
- `sX` supprime le décalage du à la numérotation et utilise le séparateur X
- `s'XXX'` supprime le décalage du à la numérotation et utilise la chaîne 'XXX'

Syntaxe : `nl` *`fichier`*

Exemple : `nl` *`lettre.tex`*

Affiche le contenu du fichier *lettre.tex* en insérant le numéro de chaque ligne en début de ligne avec un espace comme séparateur entre le numéro et le premier caractère de chaque ligne. Ne numérote que les lignes non vides. Les lignes vides sont affichées.

Exemple : `nl -ba` *`lettre.tex`*

Numérote et affiche toutes les lignes du fichier.
Synonyme de `cat -n` *`lettre.tex`*. Affichage sans doublon (**uniq**)

La commande `uniq` permet d'afficher le contenu d'un fichier ligne par ligne en ométant les doublons. En voici les options :
- `u` affichage sans doublon (par défaut)
- `d` affichage des doublons
- `c` comptage des doublons

Syntaxe : `uniq` *`fichier`*

Exemple : `uniq` *`lettre.tex`*

Affiche le contenu du fichier *lettre.tex* sur la sortie standard en ométant les duplications de lignes lorsqu'elles sont contigues.

1.7 Affichage par critère (grep)

⇨ recherche le(s) caractère(s) selon le modèle de critère dans le fichier mentionné.

Syntaxe : **grep [options] "modèle de critère" [(path) nom du fichier]**

Options :

-v : Affiche toutes les lignes ne comportant pas le modèle de caractères.
-c : (*count*) Retour uniquement le nombre de lignes trouvées, mais sans les afficher.
-i : (*ignore*) Ignore la casse (majuscules/minuscules)
-n : (*numbering*) Chaque ligne trouvée est présentée avec son numéro de ligne.
-l : (*list files only*) Ne sont affichés que les noms de fichiers contenant, dans une de leur ligne, le critère recherché. Les lignes elles-mêmes ne sont pas présentées.

Modèles de critères :

^c : Ligne commençant par la plage de caractère c.
. : Un caractère quelconque.
* : Signe de répétition.
$: Recherche en fin de ligne.

Exemples :

1) Recherche du mot « toto » dans le fichier nom.txt
grep "toto" ./Tests/nom.txt

2) Rechercher un processus
Note : les expressions suivantes sont possibles :
ps -x | grep xmms
ps -x | grep "xmms"
ps -x | grep 'xmms'

3) Vérifier les processus non mis en commentaires dans /etc/inetd.conf :
grep -v "^#" /etc/inetd.conf

4) Afficher la liste des fichiers contenant la chaîne de caratères "to" :

find /home/kmaster/DOCS/ -name "*" -exec grep -l "to*" {} \;

5) Rechercher les fichiers nommés mot?.txt (ou ? Remplace un caractère quelconque) et contenant la chaîne de caractère toto :
find . -name 'mot?.txt' -exec grep -l "toto" {} \;

6) Rechercher une chaîne de caractères se trouvant sur un fichier sur l'ensemble du disque dur:
find / -name '*' | xargs grep -l chaine_recherchee
ATTENTION ce sont des simples quottes !!

7) Vérifier si le package apache est installé et demande des informations sur le package xfree (A VERIFIER !!!)

rpm -qa | grep "apache"
rpm -qa | grep -i xfree

8) Chercher un mot clef dans les fichiers de logs :
grep mots-cles /var/log/messages

9) Vérifier qu'une librairie est installée :
ldconfig -v | grep lalib

1.8 Affichage avec tri (sort)

La commande `sort` permet de trier les lignes d'un fichier. En voici les options :
- `b` ignore les espaces en début de ligne
- `d` ordre alphabétique (A-Z, a-z, 0-9, *espace*) (par défaut)
- `f` ignore la casse
- `n` ordre numérique
- `r` inverse l'odre de tri.

Syntaxe : `sort` *`fichier`*

Exemple : `sort -bdf` *`essai.txt`*

Cet exemple permet de trier les lignes du fichier *essai.txt* dans l'ordre alphabétique (`d`) sans tenir compte des espaces de début de ligne (`b`) et sans différencier majuscules et minucules (`f`).

1.9 Découpage (split)

La commande `split` permet de découper un fichier en plusieurs plus petits. Ses options sont :
- `b` *n* (Bytes) découpage par blocs de *n* octets
ou
- `l` *n* (Lignes) découpage par blocs de *n* lignes

Syntaxe : `split` *`fichier`*

Exemple : `split -b 135000` *`vacances.mpeg`*

Ici on découpe le fichier *vacances.mpeg* qui est une vidéo volumineuse en fichiers de 1.35 Mo afin de la sauvegarder sur disquettes (de capacité maximum de 1.44 Mo).

Par défaut, les fichiers issus de la découpe auront un nom ayant pour préfixe *x* et pour suffixe une suite de lettre du type *aa*, *ab*,*ac*... créés dans l'ordre lexicographique naturel (descendant de 'a' vers 'z').
Pour changer le préfixe, il suffit de le spécifier en fin de commande.

Exemple : `wc -l 100 /var/log/httpd/access.log access.log.`

Ici on découpe le fichier de log du serveur HTTP en plus petits fichiers de 100 lignes chacun. Dont le nom sera du type *access.log.aaa*, *access.log.aab*...

Note : pour découper des fichiers texte brut, faire la découpe en nombre de lignes. Tandis que pour découper les fichiers binaires, utiliser la découpe en nombre d'octets.

Etude d'un cas d'école : on dispose du fichier *cours_de_c.pdf* qu'on souhaite sauvegarder sur disquette. Or la commande `ls` (voir Système de fichiers sous Unix > Lister les fichiers) nous apprend qu'il fait 5.7 Mo, ce qui est trop grand pour le copier directement sur disquette.
On va dont le compresser avec l'utilitaire `gzip` (voir Compression > Le compresseur gzip) afin d'en réduire la taille. Le fichier résultant *cours_de_c.pdf.gz* fait tout de même 2.8 Mo, ce qui reste encore trop.
On va donc utiliser notre recours ultime : le saucissonage de notre fichier en plusieurs plus petits de taille 1.3 Mo pour être sûr qu'ils rentreront dans les disquettes dont on dispose.

Commande : `split -b 130000 cours_de_c.pdf.gz cours_de_c.pdf.gz.`

Ce qui nous crée les fichiers suivants :
cours_de_c.pdf.gz.aa (1.3 Mo)
cours_de_c.pdf.gz.ab (1.3 Mo)
cours_de_c.pdf.gz.ac (200 Ko)
que l'on va copier chacun dans une disquette.

Pour retrouver le fichier de départ, on concatène et on décompresse.

Commandes :
```
cat cours_de_c.pdf.gz.* > cours_de_c.pdf.gz
gunzip cours_de_c.pdf.gz
acroread cours_de_c.pdf
```

1.10 Différences (diff)

La commande `diff` permet de comparer le contenu de deux fichiers pour en connaître les différences. Ceci est très pratique pour savoir si deux fichiers ont le même contenu. En voici quelques options :
- `b` ignore les différences du à des espaces blancs
- `B` ignore les différences du à des lignes blanches
- `i` ignore les différences minuscules/MAJUSCULES
- `q` indique seulement si les fichiers sont différents et ne pas afficher les différences elles-mêmes
- `s` indique lorsque deux fichiers sont identiques
- `r` comparaison récursive des fichiers d'un répertoire, sous répertoires...

Syntaxe : `diff [options] *fichier_1 fichier_2*`

Exemple :
```
$ split .signature .signature.old
3a4
> .signature.old
```

Ici on peut voir qu'il y a une différence entre les ligne 3 et 4 dans le fichier *.signature.old* où i ly a une ligne insérée. S'ils avaient été égaux, `diff` n'aurait rien affiché.

Exemple :
```
$ split -q .signature .signature.old
Files .signature and .signature.old differ
```

Exemple :
```
$ split -q .signature .signature
Files .signature and .signature are identical
```

1.11 Statistiques (wc)

La commande `wc` permet de compter le nombre de caractères, de mots et de lignes d'un fichier. Ses options sont :
- `l` (Lignes) compte le nombre de lignes
- `w` (Words) compte le nombre de mots
- `c` (Chars) compte le nombre de caractères
- `L` (Length max ligne) affiche la longueur de la ligne la plus longue

Syntaxe : `wc` *`fichier`*

Exemple : `wc` *`mail.txt`*

Sans options, `wc` renvoie atomatiquement le nombre de de lignes (`l`), de mots (`w`) et de caractères (`c`).

Exemple : `wc -lL` *`mail.txt`*

Renvoie le nombre de lignes (`l`) et la taille maximale d'une ligne (`L`).

Exemple :

```
$ wc mail.txt
12     108     671   mail.txt
```

Dans l'exemple précédent, le fichier *mail.txt* comporte 12 lignes, 108 mots et 671 caractères.

2 TYPE DE FICHIER (**FILE**)

La commande `file` permet de connaître le type d'un fichier.

Syntaxe : `file` *`fichier`*

Cette commande retourne le type d'un fichier passé en paramètre. Pour opérer, elle fait appel à un fichier qui contient les signatures binaires d'un grand nombre de fichier.

Elle est par exemple capable de fournir les caractéristiques du système qui a compilé un fichier exécutable parmis plusieurs centaines.

Exemple :

```
$ file ../*
../DATA:       directory
../cv.doc:     Microsoft Word document
../tp6.c:      program text C++
../mail.txt:   international ascii text
```

Dans l'exemple précédent, on souhaite connaître le type de tous les fichiers du répertoire parent.

La précision de la commande dépend du fichier *magic* qu'elle appelle.

---

3 CREATION D'UN FICHIER (TOUCH)

La commande `touch` permet de créer un nouveau fichier vide.

Syntaxe : `touch` *`fichier`*

Appliquée à un fichier déjà existant, elle modifie son heure de dernier accès et met cette dernière à l'heure courante par défaut.

---

4 CALCUL D'UN CONDENSAT (**MD5SUM**)

4.1 Présentation

Un condensat est une somme de vérification permettant de s'assurer de l'intégrité des données. Un condensat est calculé à partir d'une fonction de hachage. Ici la fonction utilisée implémente l'algorithme MD5.

4.2 A quoi ça sert ?

Lorsque l'on transmet des messages ou des fichiers par mail, ftp ou http ; il est utile de savoir si le fichier reçu à destination n'a pas subit d'altération pendant son transfert (erreur de transmission, piratage...). Pour cela on transmet en plus du fichier un condensat.
L'émeteur E calcule le condensat C du fichier F et envoie le tout au récepteur R.
Le récepteur R calcule le condensat K du fichier F et le compare à C. Si C=K alors tout va bien, sinon cela signifie que le fichier F a subit des modification durant son transfert. Et R va demander à E de le lui renvoyer.

4.3 Exemples

La fonction `md5sum` calcul le condensat d'un fichier selon l'algorithme MD5. En voici les options :
- `b` voit le fichier en binaire et pas en texte brut
- v mode verbeux
- c *`fichier`* compare le condensat avec celui enregistré dans le *fichier*

Syntaxe : `md5sum [options]` *`fichier`*

Exemple :
```
$ md5sum upload.zip
607cdbaeef8f20be5dcb428f007c9696 upload.zip
```

La commande `md5sum` affiche le condensat (`607cdbaeef8f20be5dcb428f007c9696`) et le nom du fichier (*upload.zip*).

Pour transmettre ce condensat, on fait une redirection de l'affichage vers un fichier.

Syntaxe : `md5sum` *`fichier > condensat`*

Exemple : `md5sum` *`upload.zip > upload.zip.md5`*

On va générer un nouveau condensat qu'on va comparer avec celui enregistré dans un fichier.

Syntaxe : `md5sum -c` *`condensat fichier`*

Exemple : `md5sum -c` *`upload.zip.md5 upload.zip`*

En cas d'erreur, il s'affiche : `md5sum: MD5 check failed for 'upload.zip'.`

Exemple :
```
$ md5sum -c upload.md5 -v
upload.zip     FAILED
md5sum: 1 of 1 file(s) failed MD5 check
```

---

5 RECHERCHE DE FICHIERS (**FIND**)

La commande `find` est ultra puissante, elle permet de faire une recherche sur le système de fichier et d'afficher la liste des fichiers satisfaisant à une combinaison de critères très variés.

5.1 Introduction

Syntaxe : `find` *`répertoire critères`* `[-print]`

Exemple : `find . -name "*.c" -print`

L'exemple précédent lance la recherche depuis le répertoire courant (`.`) et affiche le résultat de la recherche (`print`). Le critère de recherche porte sur le nom (`name`) et doit satisfaire le motif (expression régulière) suivant : "`*.c`" c'est-à-dire tous les fichiers d'extention `.c` (autrement dit les programmes sources écris en langage C).

Exemple : `find . -mtime +7 -print`

Cet exemple recherche et affiche (`print`) les fichiers dont la date de dernière modification (`mtime` : Modify Time) remonte à plus de 7 jours (`+7`).

On aurait pu spécifier `mtime -5` pour les fichiers dont la date de dernière modification date au plus de 5 jours. Ou encore `mtime 10` pour les fichiers modifiés exactement il y a 10 jours.

5.2 Liste des critères de recherche

Le tableau suivant récapitule les principales options de la commande `find`. Chacune de ces options (sauf la première `print`) sont des critères de recherche qui renvoient un bouléen *vrai* si le critère est satisfait, *faux* sinon. Si exactement tous les critères sont satisfaits, alors le fichier est considéré comme « *trouvé* » et est passé en paramètre à l'option `print` (si celle-ci est elle aussi spécifiée).

Option	Description
`print`	affiche le résultat
`name motif`	nom du fichier
`iname motif`	idem mais sans tenir compte de la casse
`mtime +n|-n|n`	nombre de jours depuis la date de dernière modification
`ctime +n|-n|n`	nombre de jours depuis la date de création
`atime +n|-n|n`	nombre de jours depuis la date de dernier accès
`mmin +n|-n|n`	nombre de minutes depuis la date de dernière modification
`cmim +n|-n|n`	nombre de minutes depuis la date de création
`amin +n|-n|n`	nombre de minutes depuis la date de dernier accès
`type type`	type du fichier b (fichier spécial en mode bloc), c (fichier special en mode caractère), d (répertoire), p (tube nommé), f (fichier normal), l (lien symbolique), s (socket)
`uid n`	le fichier porte le numéro de propriétaire spécifié
`gid n`	idem pour numéro de groupe
`size n`	le fichier est de taille spécifiée dont il faut préciser l'unité : b (bloc de 512 octets, par défaut), c (octet), k (Ko)
`used n`	nombre de jours entre la création et le dernier accès au fichier
`perm +n|-n|n`	valeur numérique des droits d'accès au fichier
`user nom`	le propriétaire du fichier a pour nom celui passé en paramètre
`group nom`	idem pour le groupe
`inum n`	numéro d'inode du fichier
`links n`	nombre de liens du fichier
`newer fichier`	le fichier est plus récent que celui passé en paramètre
`anewer fichier`	on a accédé au fichier plus récemment qu'on a modifié celui passé en paramètre
`cnewer fichier`	on a accédé au fichier plus récemment qu'on a créé celui passé en paramètre
`nouser`	aucun utilisateur dans le système ne correspond au UID du fichier
`nogroup`	aucun groupe ne correspond au GID du fichier
`empty`	le fichier est vide et il est soit un fichier normal soit un répertoire
`false`	renvoie toujours faux
`true`	renvoie toujours vrai

Les options doivent êtres précédées d'un trait d'union (`-`) et leur paramètre éventuel séparé par un espace.

Les paramètres numériques `n` peuvent être passé en valeur absolue (par exemple `7`) ou bien en valeur relative (`+2` ou `-20`).

5.3 Combinatoire des tests

Chacune des options du tableau précédent constitue un test. La commande `find` offre une algèbre booléenne permettant de combiner à l'aide d'opérateurs les tests souhaités afin de former une expression complexe.

Les opérateurs suivants sont présentés dans l'ordre de priorité décroissante.

1. (*expr*)
   force la priorité d'évaluation
2. `!` *expr*
   négation logique (devien vrai si était faux et inversement)
3. `-not` *expr*
   idem
4. *expr_1 expr_2*
   expr_2 est évaluée si *expr_1* est vrai
5. *expr_1* `-a` *expr_2*
   idem
6. *expr_1* `-and` *expr_2*
   idem
7. *expr_1* `-o` *expr_2*
   expr_2 n'est pas évaluée si *expr_1* est vrai
8. *expr_1* `-or` *expr_2*
   idem
9. *expr_1*, *expr_2*
   expr_1 et *expr_2* sont toujours évaluées mais la valeur booléenne de la liste est celle de *expr_2*. La valeur de *expr_1* est dont passée aux oubliettes.

Ainsi dans les exemples précédents, les critères étaient évalués de gauche à droite tant que que le précédent renvoyait vrai.

Exemple : `find . -not -name ".c" -print`

Cet exemple affiche sur la sortie standard la liste des fichiers qui ne portent pas l'extention "*.c*".

6 COMBINATOIRE DES COMMANDES PRECEDENTES

Exemple :

```
$ tac script.txt | nl | head -n5 | tail -n3 | tac
 5  then if [ -f "/bin/$1" ]
 4       then echo "Commande externe."
 3       else echo "N'est pas une commande externe."
```

Dans l'exemple précédent, on renvoie successivement le résultat des commandes aux suivantes par l'intermédiaire des tubes (|). On lit à l'envers le fichier *script.txt*, on numérote le résultat, on en prend que les 5 premières lignes. De ces 5 premières lignes, en ne prend que les 3 dernières (c'est-à-dire en fin de compte les lignes 3 à 5). On renverse une nouvelle fois le résultat!

Gestion des Utilisateurs

Table des Matières

1 INTRODUCTION

Le stockage des données propres à la gestion des utilisateurs se fait dans les deux fichiers **/etc/passwd et /etc/group**. Dans le temps, le fichier **/etc/passwd** contenait le mot de passe crypté de chaque utilisateur. Ce mot de passe crypté figure désormais dans le fichier **/etc/shadow**.

2 LE FICHIER /ETC/PASSWD

Contient des informations(nom, UID, GID, shell, ...) sur les différents comptes.
Les deux-points servent de caractères de séparation entre les divers champs.

Nom-user:passwd:num-user:num-groupe:champ spécial:répertoire courant:prog de démarrage (shell, ...)

Exemple :
root :x:0:0:root:/root:/bin/bash
kmaster:x:500:500:kmaster:/home/kmaster:/bin/bash

Nota le deuxième champ (**x**) spécifie q'il y a un mot de passe et que celui-ci est crypté (dans **/etc/shadow**).

L'UID est compris entre 0 (root) et la constante UID_MAX (définie dans le fichier **/etc/login.defs**).
ATTENTION : Si au sein de ce fichier, plusieurs lignes possèdent le meme UID pour plusieurs noms de connexion différents, UN SEUL utilisateur est en fait défini !
Exemple : on peut ainsi définir un utilisateur stop, dont l'UID est 0 et qui execute shutdown comme commande de connexion.

3 LE FICHIER /ETC/SHADOW

Il contient les mots de passe sous forme cryptée.

Nom-utilisateur:mot de passe crypté:numéro du jour:nombre de jours

Exemple :
kmaster:1zBvl.scX$hkqgDvBu40EqpAEwZfZZQ0:11493:0:99999:7:::

Numéro du jour et nombre de jours
11493:0:99999 signifie que le mot de passe en cours a été mis en place le 11493e jour après le 1/1/1970 (date de référence), qu'il pourra à nouveau être changé 0 jour après cette date et qu'il devra obligatoirement être modifié avant le 99999e jour après la date de création (autant dire qu'il n'expire jamais...).

4 LE FICHIER /ETC/GROUP

Ce fichier permet de faire le lien entre les numéros de groupe et leurs noms.

Structure de ce fichier :

Nom de groupe:Champ spécial:Numéro de groupe:Membre1,Membre2, ...

Exemple :
root:x:0:root
kmaster:x:500:

La commande **newgrp** permet de créer un nouveau groupe.

5 AJOUT D'UN UTILISATEUR

Commandes **adduser** ou **useradd**

Syntaxe : **adduser** ou **useradd** [nom utilisateur] : permet de créer un nouveau compte utilisateur
En fait ces deux commandes sont presque équivalentes.
La commande **useradd** manipule comme l'autre les fichiers **/etc/passwd** et **/etc/shadow**, mais elle ne travaille pas directement avec le répertoire **/home**. Pour forcer l'utilisation de **/home** comme référence, il faut utiliser la commande **$ useradd -m login** .
Certains administrateurs déconseillent l'utilisation de la commande **adduser**....

useradd :
-c comment : commentaire
-d home_dir : répertoire de connexion

-g initial_group : le groupe initial, toto pour l'utilisateur toto
-G group,... : les groupes supplémentaires
-m : création du répertoire de connexion de l'utilisateur
-k skeleton_dir : par défaut le contenu du répertoire /etc/skel est recopié dans le home directory de l'utilisateur
-s shell : shell
-u UID : l'UID
-r : la commande accepte de créer un compte avec un UID inférieur à UID_MIN, défini dans /etc/login.defs

 # useradd -c "Utilisateur TOTO pour les tests" -m -d /home/toto -G orbytes -k /etc/skel/ toto

6 SUPPRESSION D'UN UTILISATEUR

Commande **userdel**

Syntaxe : **userdel [nom utilisateur]** : détruit un compte utilisateur
Attention :
Comme définit dans le chapitre !!!!!, chaque utilisateur possède un UID unique. C'est par ce numéro et lui seul qu'un utilisateur est connu du système. Le login_name ne sert que comme identifiant lors de la connexion. A chaque fois que le système nous affiche un login_name (commande ls -l par exemple), le système effectue une conversion d'information pour nous faciliter la lecture; en fait c'est l'UID qui est stocké. Il en est de même pour le groupe_name avec le GID.
Conséquence :
Si un utilisateur (ou groupe) est supprimé du fichier **passwd** sans destruction ou affectation de ses fichiers, et que l'on lance la commande **ls -l** sur un de ces fichiers, l'affichage du login_name sera remplacé par le numéro d'UID qui n'est pourtant plus utilisé. Si l'on recrée un utilisateur avec ce même UID, c'est ce dernier qui deviendra le propriétaire de ces fichiers.

C'est pour cette raison, qu'il faut impérativement utiliser la commande **userdel** de la bonne manière :

 # userdel -r [login utilisateur]

Ceci afin de TOUT détruire.

Si vous voulez vérifier ce qui appartient à un utilisateur : **find / -user [login utilisateur]**

7 MODIFICATION D'UN COMPTE

Commande **usermod**

Syntaxe : **usermod [nom utilisateur]** : modifie les paramètres d'un compte utilisateur

Options :

-L : verrouille un compte

-U : déverrouille un compte

8 MODIFICATION PASSWORD

Commande passwd

Syntaxe : **passwd [options] [login utilisateur]**

Options :

-d : détruit le password
-l : lock le compte

9 MISE EN PLACE D'UN GROUPE D'UTILISATEURS :

Le répertoire **/etc/skel** contient des modèles de fichiers de configuraton des sessions des utilisateurs.
L'administrateur peut ajouter des fichiers ou les modifier pour définir les paramètres communs à tous les utilisateurs du site.

- Création d'un groupe nommé users afin de positionner des droits à tous les utilisateurs d'un seul coup : groupadd users
- On crée le répertoire /etc/skel_intranet qui contiendra la structure de base du répertoire des utilisateurs. Taper ensuite les commandes suivantes :
  * cp -r /etc/skel /etc/skel_intranet
  * cd /etc/skel_intranet
  * mkdir public_html
  * mkdir public_html/images
  * chmod -R 755 public_html
- Création d'un compte utilisateur orbytes :
  useradd orbytes -G users -m -u 500 -k /etc/skel_intranet -c "
Compte Orbytes "
  chmod 755 /home/orbytes

Ainsi, dans /home/orbytes, on se retrouve avec un répertoire public_html/images hérité de la structure /etc/skel_intranet.

Gestion des Droits d'Accès

Table des matières

1 PRINCIPES DE BASE DES DROITS D'ACCES

Chaque fichier de l'arborescence d'UNIX/LINUX est doté d'une identification spécifique. Les informations d'identification les plus importantes s'affichent grâce à la commande **ls -l**.
Cette commande montre de façon évidente que pour chaque fichier (fichier normal, répertoire ou fichier spécial) des autorisations d'accès individuelles sont mises en place. Lors de chaque accès à un fichier, le système vérifie ces autorisations.

Mais les autorisations d'accès individuelles ne suffisent pas à protéger les fichiers des accès interdits. Lorsqu'il se connecte, l'utilisateur reçoit un numéro d'identification (*User Identification*, **UID**).
Le fichier **/etc/passwd**, qui affecte un nom à ce numéro d'utilisateur, est l'élément central de l'ensemble de la gestion des utilisateurs. En complément chaque utilisateur reçoit un numéro de groupe (Group Identification, **GID**). Nous verrons un peu plus loin que ce n'est pas tout à fait vrai. En fait, ce sont l'UID effectif (**EUID**) et le GID effectif (**EGID**) qui déterminent l'accès, mais dans la majorité des cas :

EUID = UID et EGID = GID

Le fichier /etc/passwd :
Contient des informations(mot de passe ...) sur les différents comptes :

```
root:x:0:0:root:/root:/bin/bash
ftp:x:77:77:system user for proftpd:/var/ftp:/bin/false
avrillon_serge:x:507:507:Avrillon Serge:/home/avrillon_serge:/bin/bash
guelorget_i:x:508:508::/bin/false
```

Ici l'**UID** de avrillon_serge est 507. Son **GID** est 507.

Pour vérifier ces numéros, vous emploierez la commande **id**.

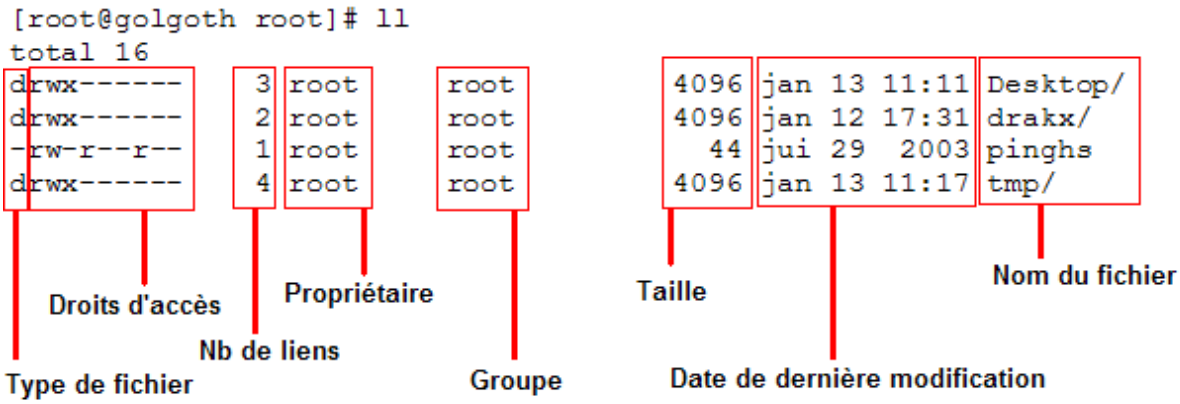
Exemple :

```
[root@golgoth etc]# id
uid=0(root) gid=0(root) groupes=0(root)
[root@golgoth etc]#

[avrillon_serge@golgoth etc]$ id
uid=507(avrillon_serge) gid=507(avrillon_serge) groupes=507(avrillon_serge)
[avrillon_serge@golgoth etc]$
```

Dans UNIX/LINUX, les noms des utilisateurs sont traduits en NUMEROS et pas représentés directement sous des noms. Ainsi, une fois la connexion établie, LINUX/UNIX oublie le nom d'utilisateur pour le remplacer par son UID. En cas de besoin du nom de l'utilisateur, le système ré inspectera le fichier /etc/passwd.
Il en va de même des fichiers. Là aussi, leur gestion au niveau du système se fait sur la base d'un numéro. Par ce numéro, vous accéderez aux données d'identification, en l'occurrence :
le nom de l'utilisateur propriétaire du fichier ;
le nom du groupe auquel ce fichier est affecté.
Bien sur, vous n'avez pas à mémoriser ces numéros, le système fait la correspondance directe à partir de son nom.

2 DROITS D'ACCES AUX FICHIERS



Chaque fichier possède des autorisations applicables à trois classes d'utilisateurs :

- **u : propriétaire du fichier**
- **g : groupe**
- **o : les autres**

Pour ces 3 catégories, il existe 3 types d'autorisation :

r : pour la lecture
w : pour l'écriture
x : pour l'exécution

D'où les droits d'accès :

Propriétaire	**Groupe**	**Autre**
R W X	R W X	R W X

2.1 Droits d'accès aux fichiers normaux

r : Le contenu du fichier peut être lu.
w : Le contenu est modifiable.
x : Le fichier peut être exécuté.

Si vous désirez réellement protéger votre fichier, préférez :

-rw------

2.2 Droits d'accès aux répertoires

r : Les éléments du répertoire sont accessibles en lecture. Cette autorisation est nécessaire pour la commande **ls** .

w : Les éléments du répertoire sont modifiables. L'utilisateur peut créer de nouveaux fichiers dans ce répertoire et supprimer les fichiers existants, cette dernière possibilité étant indépendante des autorisations d'accès aux fichiers proprement dits.

ATTENTION : Un fichier est protégé des modifications par ses autorisations, et des suppressions par les autorisations du répertoire qui le contient.

x : Le nom du répertoire peut apparaître dans un chemin d'accès. Par conséquent vous aurez accès à un répertoire par la commande **cd** à condition de disposer au minimum de l'autorisation d'exécution. On appelle ça aussi le droit de traversée.

La meilleure protection pour un répertoire serait :

drwx--x--x

2.3 Droits d'accès aux liens symboliques

Cela n'a aucun sens car il est uniquement possible d'accorder des droits au fichier désigné par le lien!

2.4 Notion de groupe

Un utilisateur peut créer un fichier et le faire appartenir à n'importe quel groupe du système. Quand il est créé, un fichier appartient au même groupe que le répertoire dans lequel il se trouve.
Par conséquent, il est possible de placer ce fichier dans le répertoire de n'importe quel groupe à condition de le doter du droit en écriture dans ce répertoire là.

2.5 Substitution d'identité

Il s'agit d'une autorisation particulière qui indique que le fichier est exécutable et que, pendant cette exécution, l'utilisateur prend tous les droits du propriétaire du fichier exécutable. Cette autorisation (**s**) remplace l'autorisation **x** du propriétaire ou du groupe.

Exemple :
Bit « s » au niveau du groupe :

```
#find /sbin/ /usr/ /bin -perm -2000 -exec ls -l {} \;
      Extrait du résultat :
      -rwxr-sr-x    1 root     games       28920 jui 18  2000 /usr/X11R6/bin/xbill
      -rwsr-sr-x    1 root     mail        63772 aoû 11  2000 /usr/bin/procmail
```

Bit « s » au niveau de l'utilisateur :

```
#find /sbin/ /usr/ /bin -perm -4000 -exec ls -l {} \;
      Extrait du résultat :
      -r-s--x--x    1 root     root        13536 jui 12  2000 /usr/bin/passwd
      -rwsr-sr-x    1 root     mail        63772 aoû 11  2000 /usr/bin/procmail
```

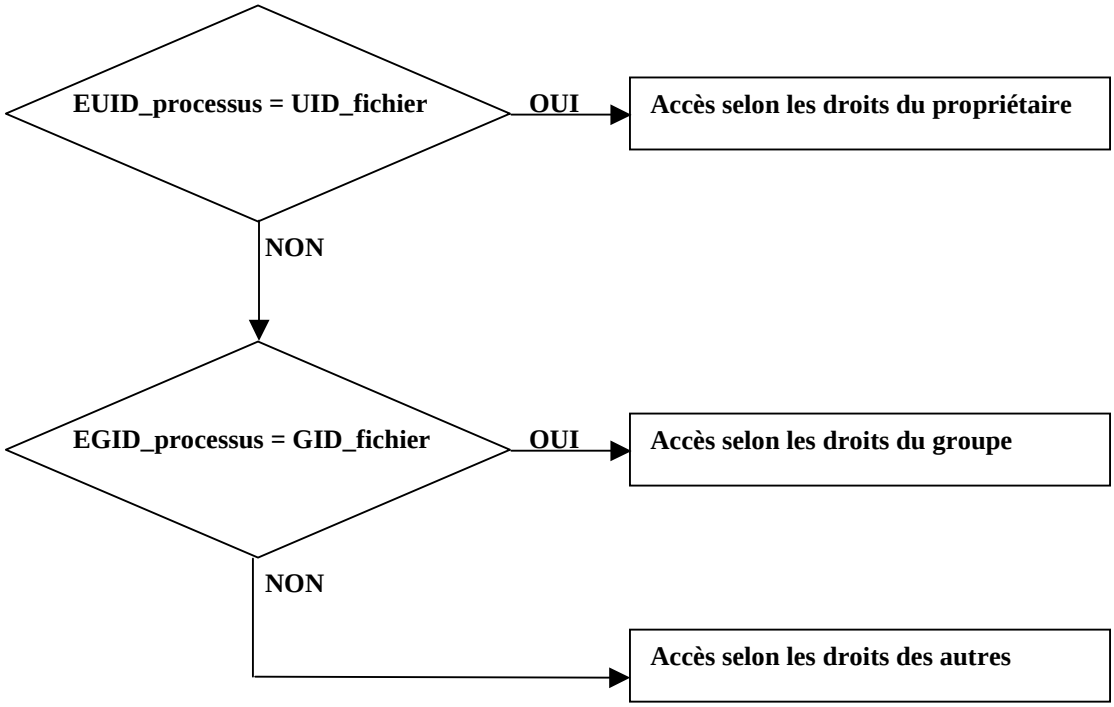
Bit « s » au niveau du groupe et de l'utilisateur :

```
#find /sbin/ /usr/ /bin -perm -6000 -exec ls -l {} \;
      -rwsr-sr-x    1 root     mail        63772 aoû 11  2000 /usr/bin/procmail
```

2.6 Le sticky bit

Ce bit (**t**) remplace l'autorisation **x** de other et s'applique uniquement aux fichiers exécutables et aux répertoires.
Dans le cas des fichiers exécutables, le sticky bit indique que le segment texte du fichier exécutable est conservé dans l'espace disque de swap une fois la commande exécutée. Cette autorisation permet un chargement en mémoire plus rapide lors d'une exécution ultérieure.
Dans le cas des répertoires, le bit t protège tous les fichiers de ce répertoire contre la suppression, quel que soit l'état des autres autorisations. Seul root pourra supprimer un fichier.

2.7 Algorithme des droits d'accès



3 MODIFICATION DES DROITS D'ACCES

Pour modifier les droits d'accès il faut utiliser la commande **chmod** (*Change mode*). Cet appel peut revêtir deux formes distinctes. La première est appelée la forme symbolique et la seconde, la forme octale. Cette dernière passe par des chiffres en base huit (et non en base 10).
La commande **chmod** ne peut être employée que par le propriétaire du fichier qui souhaite en modifier les droits d'accès. La seule exception à cette règle est l'administrateur système (root) qui peut modifier les droits d'accès de l'ensemble des fichiers.

3.1 Utilisation de la forme symbolique

Syntaxe : **chmod [options] [augo] [+-=] [rwxstugo] fichier(s)**

Options :

a : S'applique à tout le monde (user, group et other)
u : S'applique au propriétaire du fichier (user)
g : S'applique au groupe (group)
o : S'applique aux autres (other)

+ : Ajout de nouveaux droits
- : Suppression d'anciens droits
= : Redéfinition complète des droits sans tenir compte des anciens
R : récursif (**chmod -R + w .** => Rajoute le droits d'écriture pour tout le monde depuis le répertoire courant et sur tout ce qui se trouve hiérarchiquement en dessous)

Exemples :

1)

```
# ls -l
total 41
-rw-r--r--    1 root     root        40775 sep 19 14:57 rc.firewall

# chmod ug=rwx rc.firewall
# ls -l
total 41
-rwxrwxr--    1 root     root        40775 sep 19 14:57 rc.firewall
```

2)

```
# ls -l
total 41
-rwxrwxr--    1 root     root        40775 sep 19 14:57 rc.firewall

# chmod g-w rc.firewall
# ls -l
total 41
-rwxr-xr--    1 root     root        40775 sep 19 14:57 rc.firewall
```

3)

```
#ls -l
total 41
-rwxr-xr--    1 root     root        40775 sep 19 14:57 rc.firewall
# chmod g-rx,o-r rc.firewall
# ls -l
total 41
-rwx------    1 root     root        40775 sep 19 14:57 rc.firewall
```

3.2 Utilisation de la forme octale

Syntaxe: **chmod [options] valeur_permission_en_octal fichier(s)**

Options :
R : récursif (**chmod -R 777** . => Donne un accès complet pour tout le monde depuis le répertoire courant et sur tout ce qui se trouve hiérarchiquement en dessous)

Avec le chiffre en base 8, les autorisations d'accès sont redéfinies, sans que la commande prête attention aux anciens paramètres. Pour pouvoir former ces chiffres en base 8, il faut affecter des numéros aux différents droits d'accès.
En fait chaque droit correspond à un choix binaire: soit 1 (activé), soit 0 (inactif). Comme il existe 3 genre d'utilisateurs (user, group, other) possédant chacun une triplette de droits (rwx) binaires; il existe 2^3 possibilités :

Droits Symboliques	Triplette binaire	Nombre octal
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

Pour passer d'une triplette binaire à un nombre octal, il faut considérer chacun des trois binaires comme une puissance de 2.
Le binaire le plus à droite correspond à 2 à la puissance 0 => si ce bit est activé = $2^0 = 1$
Le binaire le plus au centre correspond à 2 à la puissance 1 => si ce bit est activé = $2^1 = 2$
Le binaire le plus à gauche correspond à 2 à la puissance 2 => si ce bit est activé =1 = $2^2 = 4$

Le nombre octal d'une triplette est obtenu en additionnant les puissances de 2 actives.

Exemple :
r-x <=> 101 en binaire <=> 4+1 = 5

Lorsque nous avons les 3 nombres octaux des 3 genres d'utilisateurs (user, group, other), la méthodologie à appliquer est la suivante. L'octal du propriétaire est l'octal de la centaine, celui du groupe est l'octal de la dizaine et celui du reste du monde (other) est l'octal de l'unité.

Propriétaire	Groupe	Reste du monde
rwx	rwx	rwx
400 200 100	40 20 10	4 2 1
-> 7	-> 7	-> 7

Exemples (reprises des mêmes exemples que pour la forme symbolique) :

```
1)
# ls -l
total 41
-rw-r--r--    1 root     root        40775 sep 19 16:35 rc.firewall
# chmod 774 rc.firewall
# ls -l
total 41
-rwxrwxr--    1 root     root        40775 sep 19 16:35 rc.firewall

2)
# ls -l
total 41
-rwxrwxr--    1 root     root        40775 sep 19 16:35 rc.firewall
# chmod 751 rc.firewall
# ls -l
total 41
-rwxr-x--x    1 root     root        40775 sep 19 16:35 rc.firewall

3)
# ls -l
total 41
-rwxr-x--x    1 root     root        40775 sep 19 16:35 rc.firewall
# chmod 700 rc.firewall
# ls -l
total 41
-rwx------    1 root     root        40775 sep 19 16:35 rc.firewall
```

4 VALEUR PAR DEFAUT DES DROITS D'ACCES

Nature de l'objet	Valeur symbolique	Valeur en octal
Répertoire	rwx rwx r-x	775
Fichier normal	rw- rw- r--	664

5 UMASK

Chaque fichier ou répertoire nouvellement créé est doté dès le départ de certaines autorisations d'accès.
Elles sont fonction des désirs de l'utilisateur. Celui-ci se servira de la commande **umask**, suivie, comme la commande **chmod**, d'un nombre en base 8. C'est ce nombre qui fixe les autorisations d'accès pour le nouveau fichier ou le répertoire. La logique négative de **umask** en rend la compréhension moins évidente.
Dans le masque créé par la commande **umask**, tous les chiffres en base 8 sont soustraits des droits d'accès maximaux.

Droits d'accès maximaux :

Nature de l'objet	Valeur symbolique	Valeur en octal
Répertoire	rwx rwx rwx	777
Fichier normal	rw- rw- rw-	666

Imaginons maintenant que sur le fichier qui va nouvellement être créé (de base : 664 <=> rw-rw-r--) vous souhaitiez un accès en lecture/écriture pour le propriétaire et en lecture seule pour le groupe et le reste du monde. Pour effectuer une telle opération :

Masque Max	rw-rw-rw-	666
A retirer	--- -w- -w-	022
Résultat	rw- r-- r--	644

Utilisation de la commande umask dans ce cas :

umask 022

ATTENTION:
- Si vous lancez cette commande, TOUS les nouveaux fichiers créés auront de base les droits rw-r-r.
- De plus, TOUT nouveau répertoire créé aura :

Masque Max	Rwxrwxrwx	777
A retirer	--- -w- -w-	022
Résultat	rwx r-x r-x	755

Pour revenir aux droits par défaut du système :

Nature de l'objet	Valeur symbolique	Valeur en octal
Répertoire	rwx rwx r-x	775
Fichier normal	rw- rw- r--	664

Il faut lancer la commande : **umask 002**

Masque Max	rw-rw-rw-	666
A retirer	--- --- -w-	002
Résultat	rw- rw- r--	664

RAPPEL FINAL (pour enfoncer le clou) :
La commande **umask** agit sur tous les éléments qui ne sont pas encore créés (fichiers et répertoires).
La commande **chmod** agit uniquement sur les fichiers et répertoires existants.

6 CHANGEMENT DE PROPRIETAIRE ET DE GROUPE

La commande **chown** permet d'effectuer un changement de propriétaire.
La commande **chgrp** ne peut être utilisée que pour rendre des fichiers et des répertoires accessibles à d'autres groupes dont on est également membre.

6.1 Chown

Permet de changer le propriétaire et/ou le groupe d'affiliation.

Syntaxe: **chown [options] utilisateur[.[groupe]] fichier(s)**

Options :
-R : Agit récursivement sur tous les fichiers d'un répertoire.

Exemples :
1)

```
# ls -l
total 1
-rw-rw-r--    1 root     root            5 sep 19 17:23 test.txt

# chown kmaster test.txt
# ls -l
total 1
-rw-rw-r--    1 kmaster  root            5 sep 19 17:23 test.txt
```

2)

```
# ls -l
total 1
-rw-rw-r--    1 root     root            5 sep 19 17:23 test.txt
# chown kmaster.kmaster test.txt
# ls -l
total 1
-rw-rw-r--    1 kmaster  kmaster         5 sep 19 17:23 test.txt
```

3)

```
#chown -R kmaster.kmaster /backup
```

6.2 Chgrp

Permet de rendre accessible des fichiers à d'autres groupes.

Syntaxe : **chgrp [options] groupe fichier(s)**

Options :

-R : Agit récursivement sur tous les fichiers d'un répertoire

Exemple :

```
# ls -l
total 1
-rw-------    1 kmaster  kmaster         5 sep 19 17:23 test.txt
# chgrp root test.txt
# ls -l
total 1
-rw-------    1 kmaster  root            5 sep 19 17:23 test.txt
```

Utilisation de LINUX en Réseau

Table des Matières

1 RECONFIGURATION DE L'ADRESSE IP D'UNE INTERFACE RESEAU :

Modification permanente :
Soit utiliser linuxconf (pas terrible....)
Soit le faire à la main (marche à tous les coups) :
vi /etc/sysconfig/network-scripts/ifcfg-ethX
avec X = numéro de l'interface.

Exemple de fichier :

Sans DHCP :

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=10.0.0.255
IPADDR=10.0.0.206
NETMASK=255.255.255.0
NETWORK=10.0.0.0
ONBOOT=yes
```

Avec DHCP :
BOOTPROTO=dhcp

Modification temporaire :
Soit en ligne de commandes :
/sbin/ifconfig eth0 192.168.1.254 netmask 255.255.255.0

2)Redémarrer le démon réseau et vérifier la configuration :
Redémarrage de l'interface : **/etc/rc.d/init.d/network restart**
Vérification du paramétrage d'une interface : **ifconfig**

2 POUR METTRE LES DNS A INTERROGER :

Il faut renseigner le fichier **/etc/resolv.conf**.
Exemple :
vi /etc/resolv.conf : ce fichier contient par exemple :

```
search orbytes.fr
nameserver 194.51.3.49
nameserver 194.51.3.65
```

3 POUR FIXER LE NOM DE LA MACHINE ET SON DOMAINE :

Vérifier dans **/etc/sysconfig/network**
Rajouter **HOSTNAME=**Venus.labo.fr et **DOMAINNAME=**labo.fr

4 LISTER LES ROUTES :

/bin/netstat -nr
ou
/sbin/route -n

5 AJOUTER UNE ROUTE SPECIFIQUE :

De manière temporaire :
/sbin/route add -net 172.20.2.0 netmask 255.255.255.0 gw 10.0.0.1 eth0 (on peut rajouter :1 à la fin s'il s'agit d'une interface virtuelle)
De manière permanente :
A)Ancienne méthode :
Modifier l'un des 3 fichiers suivants : **/etc/rc.d/rc3.d/S10network**, ou **/etc/sysconfig/network-scripts/ifup-routes**, ou alors le fichier **/etc/rc.d/rc.local**
En rajoutant dans tous les cas une ligne équivalente à celle entrée en ligne de commande.
B) Nouvelle méthode :
Il faut modifier le fichier **/etc/sysconfig/static-routes** :
eth0 net 172.20.2.0 netmask 255.255.255.0 gw 10.0.0.1

6 METTRE UNE GATEWAY PAR DEFAUT :

Soit de manière permanente :
vi /etc/sysconfig/network :
NETWORKING=yes
HOSTNAME=kmaster.orbytes.fr
GATEWAY=10.0.0.1
GATEWAYDEV=
Soit de manière temporaire, en ligne de commande :
/sbin/route add default gw 10.0.0.1

7 DETRUIRE UN GATEWAY PAR DEFAUT :

/sbin/route del default gw 10.0.0.1
ou tout simplement :
/sbin/route del default

8 INFOS SUR LA CONVERSION ADRESSE MAC/ @IP :

arp -n

9 FAIRE TOMBER (= ARRETER) LA PREMIERE INTERFACE ETHERNET

/sbin/ifconfig eth0 down

ou **/sbin/ifdown eth0**

10 MONTER (= DEMARRER) LA PREMIERE INTERFACE ETHERNET

/sbin/ifconfig eth0 up
ou **/sbin/ifup eth0**

11 FAIRE DU MULTI HOMING = PLUSIEURS ADRESSES IP POUR UNE SEULE INTERFACE :

Exemple :
on a une interface **eth0** avec comme adresse **172.17.6.5** (netmask 255.255.255.0)
On veut rajouter une autre adresse en plus sur cette interface :
/sbin/ifconfig eth0:1 172.17.6.6 netmask 255.255.248.0

12 FAIRE UN PING EN EVITANT LA RESOLUTION DE NOM

ping 10.0.0.1 –n

13 ANNEXES :

MANDRAKE

La plupart des adaptateurs ethernet sont reconnus, dès l'installation, par les distributions récentes de Linux.
Il faut cependant parfois reprendre la configuration (adaptateur trop récent, changement d'adaptateur, mise à jour de driver, ...).

Fichiers, services et utilitaires

Sous Mandrake, cette configuration est mise en oeuvre via (Cf. les "man" pour descriptions détaillées) :

- Les services "network", "portmap"
- Les fichiers "/etc/sysconfig/network", "/etc/sysconfig/network-scripts/ifcfg-eth0", "/etc/conf.modules"
- Les répertoires "/etc/sysconfig/network-scripts", "/lib/modules/noyau/net/", /etc/sysconfig/network-scripts/
- Les commandes ifconfig, ping,...
- Le fichier "/var/log/messages"
- La doc dans "/usr/doc/kernel-pcmcia..."

Configuration

- **linuxconf**/ Configuration/Réseau/Config. de base/Adaptateur 1
  - Nom : pc.domaine.fr (Cf. "/etc/sysconfig/network")
    Adresse IP, masque : Cf. administrateur réseau (Cf. "/etc/sysconfig/network-scripts/ifcfg-eth0")
  - Interface réseau : eth0 (Cf. "/etc/sysconfig/network-scripts/ifcfg-eth0" , "/etc/conf.modules")
  - Module noyau : celui de la carte (ou compatible !) (Cf. "/etc/conf.modules")
- Lancer ou relancer le service **"network"** Cf. commande : "ifconfig eth0 down/up"
- Si la carte n'est pas reconnue une visite du répertoire " **/lib/modules/_noyau_/net/**" s'impose pour savoir si elle est supportée par le paquetage que vous utilisez ; Sinon, il vous faudra rechercher et installer un nouveau driver ; La visite d'un How To sera probablement utile...
- Vérification de la configuration et validation de la connexion :
  - Par la commande "ifconfig"
  - Voir le fichier "/var/log/messages" pour analyser les messages au démarrage
  - Commande "**ping**"

Configuration du réseau

- Outil interactif : netcfg
- Modification manuelle des fichiers
  - /etc/sysconfig/network : nom de la machine, paramètres généraux
  - /etc/sysconfig/network-scripts/ifcfg-xxxx : configuration de chaque interface réseau
  - /etc/resolv.conf : nom du domaine et serveurs DNS
  - /etc/nsswitch.conf : définition de l'ordre de recherche des noms de machines
  - /etc/hosts : liste des noms de machines locaux
  - /etc/modules.conf : paramètres matériels des cartes réseau (entre autres)
- Commandes manuelles, de diagnostic et de dépannage
  - ifup, ifdown : démarre/arrête une interface réseau selon les paramètres spécifiés dans /etc/sysconfig/network-scripts
  - ifconfig : configure manuellement une interface réseau, ou affiche sa configuration
  - route : définit ou affiche les règles de routage
  - hostname : affiche ou change le nom de l'ordinateur
  - modprobe, lsmod, rmmod : démarre/arrête les gestionnaires matériels des cartes réseaux
  - commandes usuelles : ping, traceroute, nslookup, arp...

Les scripts de démarrage et le lancement des services

- Démarrage : le programme init exécute /etc/rc.d/rc.sysinit, puis les scripts déterminés par le runlevel.
- Les niveaux d'exécution (runlevels) : à chaque niveau correspond un état de la machine. Le programme init effectue les changements de runlevels (exemple : init 1 passe en mode mono-utilisateur).
  - 0 = arrêt
  - 1 = mode mono-utilisateur avec services réduits (pour dépannage)
  - 2 = mode multi-utilisateurs avec services réduits
  - 3 = mode multi-utilisateurs normal, sans X Window
  - 5 = mode multi-utilisateurs normal avec X Window (environnement graphique)
  - 6 = redémarrage de l'ordinateur
- Les scripts de démarrage à la norme System V : pour chaque runlevel, un répertoire contient la liste de ce qu'il faut démarrer ou arrêter (/etc/rc?.d ou /etc/init.d/rc?.d).
- Le fichier /etc/inittab définit le runlevel par défaut

Les services - aspects de sécurité réseau

- Un service = un programme qui tourne en tâche de fond (daemon). Les services fournissent des fonctionnalités à l'ordinateur local ou aux autres ordinateurs du réseau.
- Les services sont lancés par les scripts System V ou le méta-service inetd (ou xinetd).
- quelques services réseau usuels :
  - httpd : serveur web
  - ftpd, wu-ftpd : serveur FTP
  - samba (smbd et nmbd) : serveur de partage de fichiers compatible Windows
  - telnetd : serveur telnet
- quelques services locaux usuels :
  - gpm : gestionnaire de la souris en mode texte
  - syslog : gestionnaire des journaux d'événements (logs)
  - cron, anacron : gestionnaire des tâches programmées
  - kudzu : détecteur d'ajout/suppression de matériel
  - network : arrêt/démarrage du réseau
- Services pouvant être locaux ou réseau selon leur configuration :
  - lpd : gestionnaire des imprimantes
  - sendmail : envoi de courrier électronique
- Pour limiter les risques de piratage, désactiver tous les services réseau non utilisés.
- arrêt et démarrage manuel d'un service : /etc/rc.d/init.d/ *nomduservice* start|stop
- Activation et désactivation d'un service indépendant : par ntsysv, chkconfig ou modification manuelle des liens dans rc?.d
- Activation et désactivation d'un service lancé par xinetd : édition d'un fichier dans /etc/xinetd.d; redémarrage de xinetd pour valider. (anciennement : service inetd, et fichier unique /etc/inetd.conf)
- identification des services réseau en fonctionnement : ps -ax (ou -ef), netstat -a, lsof -i

Ignorer les requetes ICMP

Pour déactiver:

echo 1 > /proc/sys/net/ipv4/**icmp_echo_ignore_all**

Pour activer:

echo 0 > /proc/sys/net/ipv4/**icmp_echo_ignore_all**

Cette commande intervient pour toutes les interfaces. localhost, eth0, eth1, etc

Autre méthode:

Ajouter les lignes suivantes dans /etc/sysctl.conf

net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_echo_ignore_all = 1

Et lancer la commande suivante pour appliquer les changements immédiatement:

sysctl -p

Cette configuration restera après le reboot.

L'Editeur de Texte Vi

Table des Matières

Vi

vi est une extension de ses deux prédécesseurs **ed** et **ex**.

Actuellement une version « couleur » existe : **vim**.

vi [options] fichier [fichier...]

Options :

-r : Permet de reprendre le fichier qui était édité au moment d'un plantage.
+n : Permet d'afficher le fichier directement au niveau de ligne n.

L'éditeur **vi** possède deux modes :
- le mode Commande ;
- le mode Insertion.

Pour entrer en mode Insertion, on utilise les commandes i ou a

- tapez [i] ou [Insert] pour insérer du texte à l'endroit où se trouve le curseur,
- tapez [A] pour ajouter du texte à la fin d'une ligne.

Pour sortir du mode Insertion, on appuie sur la touche « echap »

1 Commandes de saisie de texte

Ajout :	**a** : fin curseur	**A** : fin de ligne
Insertion :	**i** : début curseur	**I** : début ligne
Nouvelle ligne :	**o** : au-dessous	**O** : au-dessus

2 Déplacement par écran

ctrl + F : une page écran vers le bas
ctrl + B : une page écran vers le haut

3 Déplacement par caractère

h : Déplacement vers la gauche
l : Déplacement vers la droite
k : Déplacement vers le haut
j : Déplacement vers le bas

4 Déplacement par ligne

0 : Retour au début de la ligne courante
$: Déplacement en fin de ligne
G : Déplacement en fin de fichier
:n : Déplacement du curseur sur la ligne n
:$: Déplacement sur la dernière ligne du fichier
Crtl + G : Permet de connaître le numéro d'une ligne

5 Marquer une ligne

mc : Marque la position courante du curseur par la lettre x

6 Numéroter/Dé-numéroter les lignes

: se nu : Pour afficher les numéros des lignes
: se nonu : Pour enlever le numéro des lignes

7 Modification et correction des erreurs

r*c* : remplace le caractère sous le curseur par le caractère c
dd : supprime la ligne active
df*c* : Supprime tous les caractères à partir du curseur et jusqu'au caractère *c*
D : Supprime tous les caractères à partir du curseur jusqu'à la fin de la ligne
yy : Copie la ligne dans un buffer
yw : Copie le mot (quand curseur placé au début du mot)
yf*c* : Copie les caractères à partir du curseur jusqu'à *c*
y$: Copie les caractères à partir du curseur jusqu'à la fin de la ligne
p : Coller le contenu du buffer
:[x,y]d : Effacer de la ligne x à la ligne y
:r Nom_Fichier : Insère un fichier en dessous du curseur

Voici la séquence qu'il faut faire pour copier/coller plusieurs lignes à la fois :
"a4yy : Copie les 4 lignes à partir de la ligne courante dans le buffer nommé a
"ap : Colle la sélection précédente

Voici la séquence qu'il faut faire pour couper/coller plusieurs lignes à la fois :
"a4dd : Coupe les 4 lignes à partir de la ligne courante dans le buffer nommé a
"ap : Colle la sélection précédente

8 Rechercher et remplacer

La syntaxe générale est :
:[x,y]s /Modèle/Remplacement/[gc]

Sans l'indication de la lettre **g** (*globa*l) après le dernier slash, comme élément de fin du texte de remplacement, seule la première occurrence de recherche pour chaque ligne sera remplacée. Si la lettre **g** suit le slash, toutes les occurrences seront remplacées.

Si le **g** est remplacé par un **c** (*confirm*), vous serez obligé de valider ou de refuser chaque remplacement individuellement.

Exemples :

Remplace dans la ligne courante la première occurrence du mot GASP par GlobalServiceProvider
:s/GASP/GlobalServiceProvider/

Remplace dans la ligne courante toutes les occurrences du mot GASP par GlobalServiceProvider
:s/GASP/GlobalServiceProvider/g

Remplace la première occurrence du mot GASP par GlobalServiceProvider dans l'ensemble du texte (de la première ligne à la dernière)
:1,$s/GASP/GlobalServiceProvider/

Remplace toutes les occurrences du mot GASP par GlobalServiceProvider dans le texte, de la ligne courante à la dernière
:.,$s/GASP/GlobalServiceProvider/g

9 Recherche de texte

/word : cherche la chaîne contenant *word*

10 Récupération sur erreur

u : Annule la dernière modification
. : Répète la dernière modification

11 Sauvegarder et/ou Quitter vi

Quitter et enregistrer : **ZZ** ou **Echap :wq**
Quitter sans enregistrer : **:q !**
:w Nom_Fichier : Sauvegarder le fichier en spécifiant le nom

12 Editer plusieurs fichiers

vi fic1.txt fic2.txt
Puis :
:n : Permet de passer directement au fichier suivant (fic2.txt)
:e # : Permet de revenir au fichier précédent (fic1.txt)
:rew : Permet de revenir au premier fichier

:e Nom-Fichier : Editer un autre fichier sans quitter **vi**

Home page de VI : http://www.vim.org/

Lancement Automatique de Processus

TABLE DES MATIERES

1 INTRODUCTION

Le lancement automatique de processus, ou même de commandes, est effectué grâce au démon **crond**.
Celui ci va explorer le fichier /etc/crontab dans lequel sont référencées toutes les actions à engager.

Il est possible de lancer des processus de deux manières différentes :
- Lancement différé d'un processus grâce à la commande **at** ;
- Lancement cyclique d'un processus grâce à la commande **crontab**.

2 LANCEMENT DIFFERE

Cela signifie que nous voulons lancer un processus à un moment donné, et par conséquent UNE SEULE FOIS. Ce « moment donné » peut être défini aussi bien comme une date précise ou sous la forme « dans un temps défini à compter de maintenant ». Pour cela nous utiliserons la commande **at**.

Syntaxe : **at Heure [date] nom_du_processus**

Le démon **cron** contrôle entièrement la commande **at**, en lançant de manière cyclique la commande **atrun** : celle-ci va explorer le répertoire **/var/spool/at/spool** qui contient les lancements **at** en cours ou en attente. Ces lancements sont créés sous forme de fichiers de nomenclature ***yy.mm.hhhh*** (pour année, jour et heure d'exécution du processus).

La commande **atq** permet d'afficher la queue des processus en attente de lancement.

La commande **atrm** permet de supprimer un lancement automatique.

Supposons que nous voulions lancer, à 17h05, la commande **df -h**.
La commande à entrer est alors :

 # at 17:05
warning: commands will be executed using /bin/sh

at étant prêt à recevoir une liste de commandes, comme l'indique son invite « **at>** ».
Entrons la commande à exécuter, puis terminons par **CTRL D** :

at> df -h
at> ^D
at> <EOT>
job 2 at 2001-09-21 17:05

A l'heure spécifiée, cette commande sera exécutée par le Shell **/bin/sh**. Une fois l'exécution de cette commande terminée, **at** enverra à son instigateur un courrier électronique reprenant les sorties et les messages d'erreur de la commande exécutée.

En réalité les choses n'ont pas eu lieu tout à fait comme cela. En clôturant la saisie des commandes, **at** a affiché un message du type :

job 2 at 2001-09-21 17:05

Ce message indique que les commandes ont été enregistrées en tant que numéro 2 et seront exécutées à la date prévue. Si vous voulez savoir exactement ce qui aura lieu, c'est très simple : connaissant votre numéro de commande, il vous suffit de lancer :

 # at -c numéro

Ce qui donne dans notre cas :

```
#!/bin/sh
# atrun uid=0 gid=0
# mail  kmaster 0
umask 22
[...]
OLDPWD=/var/spool/at; export OLDPWD
cd /var/spool/at/spool || {
 echo 'Execution directory inaccessible' >&2
 exit 1
}
df -h
```

Il s'agit d'un script Shell qui initialise un certain nombre de variables d'environnement, tente de se déplacer dans le répertoire de l'utilisateur, avant de lancer la commande souhaitée.

Supposons maintenant que vous ayez demandé l'exécution d'une commande, mais que vous ne sachiez plus à quelle heure ou quelle était la commande ou son numéro... La commande **atq** (ou **at -l**) vient alors à votre secours et vous indique numéro et date des commandes en attente. Utiliser en combinaison avec « **at -c** » si vous ne souvenez plus des commandes entrées, cela vous permet d'inspecter toute la file des commandes en attente.

 # atq
2 2001-09-22 17:05 a root

La lettre **a** minuscule, à droite de la date de déclenchement, est un indicateur de file d'attente, allant de a à z. Plus la lettre est loin dans l'alphabet, plus la priorité de la file d'attente est faible. Ceci permet de hiérarchiser l'exécution des commandes dans le cas d'un déclenchement de plusieurs commandes à la même heure.

 # at -q d 17:45
warning: commands will be executed using /bin/sh
at> echo toto
at> <EOT>
job 3 at 2001-09-21 17:45
atq
2 2001-09-22 17:05 a root
3 2001-09-21 17:45 **d** root

La commande **atrm** permet de supprimer des commandes de la file d'attente avant leur exécution :

atq
2 2001-09-22 17:05 a root
3 2001-09-21 17:45 d root
 # atrm 3
atq
2 2001-09-22 17:05 a root

Pour finir, outre les heures sous la forme **hh:mm**, vous pouvez mentionner un jour précis sous la forme **mm/jj/aa** (le format anglo-saxon) à la suite de l'heure :

1)Déclenchement de la commande le 21 septembre 2001 à 17h55 :
 # at 17:55 09/21/01

2)Déclenchement de la commande dans 1 heure à partir de maintenant :
 # at now + 1 hours

Attention : il est possible de restreindre les droits d'utilisation de la commande **at** grâce aux fichiers **/etc/at.allow** et **/etc/at.deny**. Si **/etc/at.allow** existe, seuls les utilisateurs dont le NOM figure dans ce fichier seront autorisés à utiliser at. Au contraire, si **/etc/at.deny** existe, aucun des utilisateurs dont le NOM est présent dans ce fichier ne pourra utiliser **at**.

3 LANCEMENT CYCLIQUE

Si **at** donne pleine satisfaction pour l'exécution d'une commande, il n'est pas adapté au lancement de tâches périodiques.
C'est là que **cron** entre en scène.

3.1 Présentation de cron

Derrière le terme CRON se cache 3 éléments :

3.1.1 Le démon crond :

Ce démon est un programme lancé automatiquement au démarrage de votre système. Il fonctionne continuellement en tâche de fond et vérifie toutes les minutes dans les « cron tables » si une commande doit être exécutée.

3.1.2 Les fichiers « cron tables » :

Ce sont des fichiers contenant les commandes planifiées par les utilisateurs et qui doivent être lancées à intervalles réguliers. Il s'agit simplement de fichiers texte.

Les « cron tables » sont chargées en mémoire par le démon **crond** et se trouvent dans le répertoire **/var/spool/cron/**. Il en existe une « cron table » par utilisateur. Par exemple, pour l'utilisateur root, vous trouverez le fichier **/var/spool/cron/root**.

Attention: il ne faut jamais éditer directement les fichiers de « cron tables ». Il faut passer par l'utilitaire **crontab** afin que le système prenne en compte les modifications.

3.1.3 La commande crontab :

Placée dans le répertoire **/usr/bin/**, elle permet plusieurs actions en fonction des paramètres qui suivent cette commande.

Voici les principales utilisations :

- crontab -e (e comme *edit*)
Cette commande ouvre l'éditeur par défaut (vi) et votre « cron table », si elle existe, sinon elle sera créée lorsque vous enregistrerez le fichier.
- crontab -l (l comme *list*)
Cette commande affiche simplement votre table.
- crontab -r (r comme *remove*)
Supprime votre « cron table ».
- crontab -u Nom-Utilisateur (u comme *user*)
Pour éditer la table d'un utilisateur.

Voici un extrait d'un fichier de configuration :

```
# crontab -l
        SHELL=/bin/bash
        PATH=/sbin:/bin:/usr/sbin:/usr/bin
        MAILTO=root
        HOME=/

        # run-parts
        01 * * * * root run-parts /etc/cron.hourly
        02 4 * * * root run-parts /etc/cron.daily
        22 4 * * 0 root run-parts /etc/cron.weekly
        42 4 1 * * root run-parts /etc/cron.monthly
        # Execution de supercleaner pour le nettoyage des logs :
        15 10 * * 1 /sbin/supercleaner
```

Ce fichier de « cron table » définit quelques variables d'environnement, avant d 'énumérer les tâches périodiques.

3.2 Droits d'utilisation

Vous pouvez autoriser ou refuser l'édition d'une « cron table » aux utilisateurs de votre machine.
Si un fichier **/etc/cron.allow** existe alors seuls les utilisateurs listés dans ce fichier pourront utiliser la commande **crontab**.
En revanche, si le fichier **/etc/cron.deny** existe et que **/etc/cron.allow** n'existe pas, les utilisateurs inscrits dans **/etc/cron.deny** ne pourront pas utiliser la commande **crontab**.
Si aucun des deux fichiers n'existe, tous les utilisateurs de la machine pourront lancer la commande **crontab**.

3.3 Syntaxe de cron

Les commandes que le démon crond doit exécuter s 'écrivent sur une seule ligne est doivent obligatoirement respecter la syntaxe suivante :

mm hh jj MM JJ Commande-à-lancer Options

mm : minutes (de 0 à 59)
hh : heures (de 0 à 23)
jj : jour du mois (de 1 à 31)
MM : mois (de 1 à 12)
JJ : jour de la semaine (de 0 à 6)
0 = dimanche
1 = lundi
2 = mardi
3 = mercredi
4 = jeudi
5 = vendredi
6 = samedi

Commande-à-lancer :
Exemple : si vous voulez copier le fichier /home/kmaster/test.txt dans /home/kmaster/archives :
cp /home/kmaster/test.txt /home/kmaster/archives

Remarquez que, contrairement à **at**, il n'est possible de préciser qu'une seule commande à la fois. Si vous souhaitez exécuter un enchaînent de commandes, il vous faudra écrire un script Shell et faire exécuter ledit script par **cron**.

Options : Pour écrire des informations périodiquement dans un fichier, tapez :
> /mon_répertoire/nom_du_fichier.
Ceci effacera le fichier « nom_du_fichier » puis inscrira les informations. Pour ne pas vider le fichier à chaque fois mais écrire à la suite des données déjà présentent, remplacez > par >>. Un mail de confirmation vous sera envoyé après l'exécution d'une commande. Pour ne pas recevoir ce mail, tapez > /dev/null.

mm, hh, jj, MM, JJ peuvent être également substitués par :

, : La virgule représente le « et ». Pour lancer une action le 12 et le 15 du mois, taper « 12,15 » à la place de jj.
- : Le tiré signifie « jusqu 'à ». Ainsi « 12-15 » signifie du 12 au 15.

***** : L'étoile représente toutes les valeurs d'un paramètre. Si vous mettez une étoile à la place
de hh, l'action sera effectuée toutes les heures.

/ : Le slash vous permet de spécifier une répétition. Pour tous les 3 mois, remplacez MM par
*/3

3.4 Edition de votre cron table

Lancez la commande **crontab -e** .
L'éditeur **vi** s'exécute.

Exemples :

1)Lister le contenu de /usr dans le fichier /home/kmaster/usr.txt tous les jours à midi :
0 12 * * * ls /usr > /home/kmaster/usr.txt

Quand vous quittez l'éditeur en enregistrant, vous voyez apparaître le message : « crontab: installing new crontab ». L'utilitaire crontab vous informe que les modifications du fichier ont été prises en compte.

2)Exécuter un script de nettoyage des logs tous les lundi à 10 h 15 :
15 10 * * 1 /sbin/supercleaner

3.5 Exemples

Exemples simples :

Une petite série d'exemples ne fera pas de mal ;).

- Lancer une tâche le 1er et 15 du mois à 1h du matin :
  0 0 1,15 * * tâche à lancer
- Rebooter la machine une fois par mois, le 7, à 3h du matin :
  0 3 7 * * /sbin/shutdown -r
- Lancer un script tous les Mardi à 3 h 30 du matin :
  30 3 * * 2 /usr/monscript
- Lancer un script tous les matins, du Lundi au Vendredi à 6h15 :
  15 6 * * 1-5 /home/monscript
- Plus dur ;) ... Lancer une tâche tous les 1/4 d'heure de 15h à 18h, du Lundi au Vendredi et cela, seulement la 1ère quinzaine du 3ème trimestre :
  0,15,30,45 15-18 1-15 7-9 1-5 /usr/ma tâche
- Une commande directe, trouver puis supprimer du répertoire **/tmp** les fichiers non-modifiés et datant de plus de 31 jours.La tâche se fera tous les 1er de chaque mois à 2h du matin :
- **0 2 1 * * find /tmp -atime 31 -exec rm -f {} ;**

Exemples pratiques :

- Sauvegarde des répertoires personnels dans **/home** le 1er du mois à 18h :
  0 18 1 * * /usr/monscript

Détail du fichier " **monscript** ":

```
#!/bin/sh
#SAUVEGARDE DES REPERTOIRES PERSONNELS
date="$(date)
set -- $date
tar -czvf /var/sauvegarde/home/home.$3$2$6.tgz /home/*
```

- Pratique, ce script vous permettra de vérifier si le service "SAMBA" est en cours de fonctionnement ou non. Si ce n'est pas le cas, le script relance le service et vous prévient par E-mail.Nous prévyons de vérifiez cela tous les 15 minutes :
  15 * * * * /usr/verifsamba

Détail du fichier " **verifsamba** ":

```
#!/bin/sh #VERIFICATION DU SERVICE SAMBA /sbin/pidof smbd > /dev/null if [ $? = 1 ] then /etc/rc.d/init.d/smb stop /etc/rc.d/init.d/smb start echo "Redemarrage de Samba" | mail -s "[SAMBA VERIFICATION]" monemail@qqch.fr fi
```

ATTENTION : les 2dernières lignes avant "fi" n'est qu'une seule et même ligne !

- Vérifions maintenant si l'ADSL est toujours connecté sinon on reconnecte : comme le script précédent, toutes les 15 minutes :
  15 * * * * /usr/verifadsl

Détail du fichier " **verifadsl** ":

```
#!/bin/sh #VERIFICATION DE LA CONNEXION if ! ping -c 1 195.165.15.65 > /dev/null 2>&1 then /usr/bin/killall pppd ppoe sleep 40 /usr/sbin/ppoe/ 10.0.0.20 fi
```

195.165.15.65 représente une adresse ip qui peut être "pinguée" sur internet.Ex : prenez www.google.fr, faite "ping www.google.fr" et relever son adresse IP puis remplacer celle de l'exemple par celle-ci.

l'adresse 10.0.0.20 représente l'adresse IP de votre modem, bien évidemment à modifier également par la votre.

Voilà, ce tutorial prend fin ici en éspérant vous avoir bien aidé. Si vous souhaitez récuperer quelques scripts sympa, dirigez-vous dans la rubrique "Téléchargements" section "Scripts".

3.6 Et anacron ?

Cron part du principe que votre système fonctionne en permanence. Or sur un ordinateur personnel, c'est rarement le cas.
Ceci pourrait poser un problème si votre système est sensé régénérer une base locate, ou tout autre tache de maintenance, toutes les nuits vers 3 à 4 heure du matin. Si votre machine n'est jamais allumée à ces heures-là, ces tâches ne seront jamais effectuées.

Pour solutionner ce problème, un utilitaire du nom d'**anacron** a été développé en plus du **cron** standard.
Les commandes s'exécutant avec des périodicités d'une journée et plus sont alors déplacées de la **crontab** classique vers la **crontab** spécifique à **anacron**.

Anacron mémorise la date de dernière exécution de chaque commande, afin de détecter si une ou plusieurs exécutions ont été sautées. Dans ce cas, au démarrage du système, la commande sautée est exécutée le plus rapidement possible.

Le format des « cron tables » d'**anacron** suit celui des **crontab** standard, en utilisant que deux colonnes pour les informations de périodicité. La première colonne indique la période d'exécution en jours, et la seconde, le délai d'attente (en minutes) avant l'exécution forcée de la commande au démarrage du démon **anacron**. La dernière colonne indique la commande à exécuter.

/etc/anacrontab: configuration file for anacron

See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# These entries are useful for a Red Hat Linux system.
1       5       cron.daily              run-parts /etc/cron.daily
7       10      cron.weekly             run-parts /etc/cron.weekly
30      15      cron.monthly    run-parts /etc/cron.monthly
```

Anacron est donc un outil extrêmement pratique pour les machines ne fonctionnant pas en permanence.

Sauvegarde de Données

1 TABLE DES MATIERES

Les données peuvent être détruites, supprimées ou rendues inutilisables pour tous les systèmes et pour des causes assez diverses :

-erreur matérielle ;
-erreur logicielle ;
-manipulation hasardeuse ou sabotage.

Différentes sortes de supports de sauvegarde :

-disquettes
-CD-Rom
-Disques zip
-cartouches DAT, ...

2 COMMANDES DISPONIBLES

Les systèmes UNIX/LINUX disposent au minimum des 3 commandes pour la sauvegarde des données :
-**tar** (en anglais *tape archiver*)
-**cpio** (en anglais *copy in/out*)
-**dd** (en anglais copy and convert, cc est le compilateur C, d'où dd !)

De ces 3 commandes, seules les 2 premières sont d'utilisation courante. La commande **dd** procède plutôt à une copie physique.

3 LA COMMANDE TAR

Cette commande permet de sauvegarder des fichiers ou des hiérarchies complètes de répertoire.

Syntaxe : **tar [options(s)] [destination] [fichier(s)_ou_répertoire(s)]**

Les principales options :

-A, --catenate ou --concatenate : ajoute des fichiers tar à une archive
-c ou --create : crée une nouvelle archive
-d, --diff ou --compare : cherche une différence entre une archive et les fichiers du disque
--delete : supprime un fichier de l'archive

-f ou --file Nom_Fichier : indique le nom du fichier archive
-r ou --append : ajoute les fichiers à la fin d'une archive
-t ou --list : donne le contenu de l'archive
-u ou --update : ne met dans l'archive que les fichiers nouveaux ou plus récents que dans
l'archive
-v ou --verbose : affiche le nom des fichiers traités
-x, --extract ou --get : extrait les fichiers d'une archive

Exemples :

1)connaître le contenu d'une archive tar
tar tvf [archives].tar

2)idem pour extraire une archive tar compressée
tar tvfz [archives].tgz ou **tar tvfz [archives].tar.gz**

3)crée une archive du répertoire mesdoc (c : créer, f : fichier)
tar -cf mesdoc.tar mesdoc/

4)crée une archive qui lors de son extraction sera placée dans le répertoire d'origine. Très pratique pour les sauvegardes !!!
tar -cf mesdoc.tar /home/toto/mesdoc

5)l'option v (verbose) permet de lister les fichiers archives
tar -cvf ...

6)crée l'archive mon_archive.tar dans le répertoire courant, et y met tous les fichiers, répertoires, sous-répertoires et leurs fichiers de /home/masse/
tar cvf mon_archive.tar /home/masse/*

7)va extraire tous les fichiers de l'archive dans le répertoire courant
tar xvf mon_archive.tar

8)va lister le nom, les droits, etc... des fichiers contenus dans l'archive
tar tvf mon_archive.tar

9)utilisation combinée de tar et gzip
tar cvf mon_archive.tar /home/masse* puis **gzip mon_archive.tar** : tar va créer l'archive mon_archive.tar, puis gzip va la compresser en mon _archive.tar.gz
L'opération inverse se faisant par : **gunzip mon_archive.tar.gz** puis **tar xvf mon_archive.tar**
Et si vous souhaitez passer en une seule ligne :
gunzip -c mon_archive.tar.gz | tar xvf -
Le "-c" de gunzip signifie « vers la sortie standard de gunzip» et le "-" de tar récupère l'archive par l'entrée standard de tar

10) crée une archive compressée avec gzip(option z)
tar -czf mesdoc.tar.gz mesdoc/ ou **tar -czf mesdoc.tgz mesdoc/**

11) crée un archive compressée en listant les fichiers. Cette archive lors de son extraction sera placée dans le répertoire d'origine
tar -czvf mesdoc.tar.gz /home/toto/mesdoc

12) crée une archive compressée
tar cvfz [nom archives].tgz [fichiers à archiver]

13) décompression de l'archive
tar xfvz [nom archives].tar.gz

14) liste une archive(option t). ceci permet de savoir le répertoire parent sera crée lors de son extraction.
tar -tf mesdoc.tar

15) idem avec une archive compressée
tar -tzf mesdoc.tar.gz ou **tar -tzf mesdoc.tgz**

16) extraction de l'archive
tar -xvf mesdoc.tar

17) extraction pour une archive compressée
tar -xzvf mesdoc.tar.gz
tar -xzvf mesdoc.tgz : idem
Attention avant d'extraire une archive présente sur un lecteur (CDROM, disquette...)
il faut la copier sur le disque dur dans un répertoire temporaire!!!

18)lister le contenu d'une archive
tar tvf X.tar | less

idem en tgz ou tar.gz :
tar tvfz X.tar.gz | less

4 LA COMMANDE CPIO

Pour une sauvegarde régulière par l'administrateur du système, utilisez la commande **cpio**. Elle utilise un format d'enregistrement interne lisible sur pratiquement tous les systèmes UNIX/LINUX.
La commande **cpio** lit les noms des fichiers à sauvegarder par le canal d'entrée standard. Elle envoie ensuite le contenu de ces fichiers par le canal de sortie standard. Vous travaillerez avec une redirection des entrées/sorties, pour éviter d'avoir à saisir manuellement les noms de fichiers et faire en sorte que les données à sauvegarder ne s'affichent pas sur l'écran mais sur le média de sauvegarde. Les noms des fichiers peuvent être :
-saisis au clavier ;
-lus dans un fichier par une redirection d'entrée ;
-produits par la commande find.

Lors de la lecture des données du support de sauvegarde, la commande **cpio** accède à ce support par le canal d'entrée standard. Là encore, il faudra une redirection d'entrée. Lors de la lecture des données, vous pourrez utiliser un critère de sélection, pour ne restaurer que certains fichiers.

Pour les commandes **tar** et **cpio**, vous ferez attention à indiquer des chemins d'accès absolus ou relatifs. Si vous sauvegardez des fichiers ou des répertoires avec des chemins absolus, vous ne pourrez les restaurer qu'à leur emplacement d'origine dans l'arborescence. La restauration des données avec chemin relatif est un peu plus souple. Attention dans ce

cas, au répertoire, dans lequel vous vous trouverez lors de l'appel à la commande de restauration.

Syntaxe : **cpio [Options] [répertoire] [fichier(s)]**

Elle peut être accompagnée d'une seule des 3 options suivantes :

-o (en anglais *output*) : La commande **cpio** lit les noms de fichiers à partir de l'entrée standard et les envoie vers la sortie standard.

-i (en anglais *input*) : Cette option demande à la commande **cpio** de lire à partir de l'entrée standard.

-p (en anglais *porting*) : D'après cette option, la commande **cpio** doit lire des noms de fichiers par l'entrée standard et placer ces fichiers dans un répertoire indiqué comme paramètre complémentaire (copie d'arborescence).

Les autres options :

-B : La commande **cpio** travaille avec des blocs dix fois plus grands, en l'occurrence des blocs de 5120 octets.

-c : La commande **cpio** utilise pour chaque fichier un en-tête de gestion composé de caractères ASCII. C'est ce qui autorise par la suite le portage des sauvegardes.

-t : En collaboration avec l'option -i, cette option crée une table des matières pour le support de données concerné.

-v : verbose.

-d : Lors de la lecture, de nouveaux répertoires sont créés si nécessaire.

Exemples :

1)On fait une sauvegarde des fichiers (ayant été créés il y a moins d'un jour) sur une diquette :

$ find /backup/COURS-LINUX/ -ctime -1 -print | cpio -ocvB > /dev/fd0

2)La restauration corresponante :

$ cpio -ivcB < /dev/fd0

3)On ne souhaite que la restauration des fichiers commençant par une minuscule (entre a et k) :

$ cpio -ivcB "[a-k]*" < /dev/fd0

4)Avec l'option -t, seule la table des matières de la bande de sauvegarde s'affiche. Aucune donnée n'est restaurée.

$ cpio -ivtBc < /dev/fd0

```
drwxrwxr-x   2 kmaster  kmaster        0 Oct  1 10:53 /backup/COURS-LINUX/
-rw-rw-r--   1 kmaster  kmaster   627200 Sep 23 12:03 /backup/COURS-
LINUX/CoursTest.sdw
123 blocks
```

5 LA COMMANDE DD (DISK DUMP)

La commande **dd**, contrairement aux commandes **cpio** ou **tar**, sert à créer une copie physique. Elle lit les données octets par octets, bloc par bloc, les convertit et les enregistre ensuite octet par octet et bloc par bloc. Les données peuvent être lues à partir soit d'un fichier normal soit d'un fichier de périphérique. Ces deux possibilités existent aussi pour l'écriture. La commande **dd** se paramètre différemment des deux commandes précédentes. Elle est suivie de différents mot-clés :

if=FICHIER
of=FICHIER

if signifie *input file* (fichier d'entrée) et **of** *output file* (fichier de sortie).

Exemple :

1)Créer une disquette de boot pour Linux :

dd if=nom_de_l'image.img of=/dev/fd0 bs=1440k

Explications:
La disquette contiendra de quoi charger le système à partir de la disquette, même si le LILO n'est pas installé, ou a été abîmé par une mauvaise manipulation ou un mauvais système d'exploitation (comme Windows par exemple).

bs=1440k précise qu'il s'agit d'un lecteur pour disquettes 1.44M.

6 LES COMMANDES DUMP, RESTORE ET MT

La commande **dump** permet de sauvegarder des systèmes de fichiers entiers.
Afin de pouvoir sauvegarder un système de fichier, il est nécessaire que ce dernier soit démonté.
Pour restaurer les dumps, utilisez la commande **restore**.
La commande **mt** permet quant à elle de contrôler les bandes et les cartouches magnétiques.

7 SAUVEGARDE DES META-DONNEES

Pour les distributions basées suer le gestionnaire de paquages de RedHat (RPM), il faut penser à sauvegarder les méta-données RPM au cours de la sauvegarde normale.
Il faut donc que vos scripts de sauvegarde contiennent une ligne du type :

 # rpm -Va > /home/save

-a : questionne tous les packages installés
-V : vérifier les packages

8 LES PROGRAMMES DE SAUVEGARDE

4 grands programmes :

- **PERFECTBACKUP+** (Merlin Software) : 1 version (free) :
doit encore beaucoup progresser (surtout dans la restauration)
- **BRU** (Enhanced Software Corp) : 2 versions (1 free et une payante) :
équivaut à un super tar. Installation super facile et bonnes performances.
- **ARKEIA** (Knox) : 2 versions (1 free et une payante) :
le plus performant dans le cas d'un parc contenant des PC windows et Linux.
Cependant : Très difficile à installer et le GUI est brouillon.
- **QUICK RESTORE** (Workstation Solutions) : 1 version payante (ou 1 CD d'éval.)
le haut de gamme !
L'installation se fait toujours avec assistance téléphonique d'un technicien, installation facile, GUI très parlante, ...
Cependant : le plus cher des 4!

Gestion et surveillance des processus

1 GENERALITES

- **Un processus est un programme en cours d'exécution.**
  Il peut donc y avoir simultanément plusieurs processus du même programme en exécution en mémoire. Le noyau Linux lance, gére les processus et controle leur échanges avec les périphériques.
- On sait déjà que le premier processus, ancêtre de tous les autres, est **`init`**
  Tous les autres sont créés par un processus parent et appartiennent à un utilisateur.
  Ainsi en règle générale, à chaque nouvelle commande, le système lance un nouveau processus.
- Chaque processus est identifié par un numéro unique, son **`PID`** qu'il peut être important de connaître
- Pour lancer un processus en tâche de fond (en arrière-plan), faire suivre la commande du symbole **&** Cela ordonne au processus parent de "reprendre la main", sans attendre la fin du processus "fils".
- Ainsi, dans une console où on a lancé **`startx &`**, on dispose toujours du processus shell pour lancer d'autres commmandes.
- Autre exemple utile
  tail -f /etc/httpd/logs/access.log > /dev/tty11 &

2 LES DIFFERENTES SORTES DE PROCESSUS

En plus des processus exécutés dans une session de travail en avant-plan (foreground), mode habituel de fonctionnement et ceux qui sont lancés en tâche de fond, il existe des processus dits détachés qui ne sont exécutés dans aucune console (ps affiche alors un ? au lieu du nom de console tty-).
Les processus détachés qui sont les exécutions de services du système (en particulier services réseaux) sont appelés **daemons**, francisés en "démons".
L'administrateur n'ayant pas directement la main sur un démon (où taper Crtl-C pour stopper le programme ?), il doit lui envoyer un signal pour le supprimer ou agir sur sa configuration. Cela s'effectue par la commande **kill**.

3 POUR CONNAITRE TOUS LES PROCESSUS EN COURS DE FONCTIONNEMENT

- **`ps`** : liste les processus actifs lancés dans la console courante.
- **`ps aux`** : affiche la liste de tous les processus, avec leur numéro PID, le terminal tty où ils ont été lancés (sinon ?). Voici la liste des colonnes du tableau obtenu .
  - "USER" à quel utilisateur appartient le processus.

    - "PID" est le numéro qui identifie le processus
    - "%CPU" en % les ressources du microprocesseur utilisées par le processus.
    - "%MEM" en % les ressources en mémoire vive utilisées par le processus.
    - "RSS" mémoire réellement utilisée en ko par le processus.
    - "START" l'heure à laquelle le processus a été lancé.
- ps aux | less : pour contrôler le défilement
- **ps aux | grep httpd** : pour n'afficher que les lignes concernant le processus cherché.
- **`pstree | less`** permet de visualiser la filiation des processus sous forme arborescente.
  Sous X-KDE, on peut utiliser `TaskManager`, qui montre cette arborescence graphiquement.
- **`pidof httpd`**, pour connaitre la liste des PID des processus d'un programme
- Le numéro PID d'un service est souvent stocké dans un fichier qui porte son nom, dans le répertoire **/var/run**

4 POUR MODIFIER L'ETAT D'UN PROCESSUS

On peut gérer les processus en leur envoyant des signaux par l'intermédiaire des commandes **kill** et **killall**, suivant que l'on connaisse le numéro PID du processus, ou bien son nom.
Voici les principales actions que l'administrateur peut etre amené à utiliser

- `kill -15 PID` : demande normale d'arret au processus, il peut refuser (-15 peut etre remplacé par SIGTERM)
- `killall -9 httpd` : suppression plus radicale, en cas de processus récalcitrant ! (Le signal -9 par exemple s'appelle SIGKILL
- `kill $(pidof ypserv)` : supprime le processus serveur NIS, dont le pid est obtenu le résultat de la commande pidof
- `killall -HUP httpd` : ordonne au processus de relire son fichier de configuration, ce qui évite de le relancer.
- `kill -l` : pour connaitre la liste des signaux qu'on peut passer à kill.
- Exemple : déconnecter radicalement l'utilisateur toto
- `  ps aux | grep toto`
- `  ---> toto 858 ..... -bash`

```
kill -9 858
```

Connaitre l'état de la mémoire

La commande **`free`** affiche la mémoire disponible, utilisée, libre ...

Connaitre les ressources utilisées par les processus

- La commande **`top`** affiche une page d'information, périodiquement mise à jour (taper q pour quitter), pour gérer les processus et être informé de la charge de travail du CPU et de l'utilisation mémoire.
- L'équivalent graphique existe sous X-KDE, lancer `K/système/gestionnaire de taches` **`Ktop`** `ou kpm ou ksysguard`

Observation et Analyse du Système

1 TABLE DES MATIERES

2 LISTER LES CONNECTIONS

La commande **who** permet d'afficher la liste des utilisateurs connectés. Pour cela, elle explore le fichier **/var/run/utmp** dans lequel sont enregistrées les fameuses connections.

Sans paramètre cette commande affiche pour tous les utilisateurs connectés les informations suivantes :

- nom de l'utilisateur ;
- terminal sur lequel il est connecté ;
- heure de connexion.

```
# who
kmaster  :0      Oct  1 10:04
kmaster  pts/0   Oct  1 10:04
kmaster  pts/2   Oct  1 10:11
kmaster  pts/3   Oct  1 10:38
```

Par les options, d'autres informations sont tirées du fichier **/var/run/utmp**. Il s'agira par exemple du numéro de processus et du nom de shell.

Syntaxe : **who [options] [am i]**

Les options :

-H : Les colonnes sont surmontées d'un en-tête.

-u ou -i : Indique le temps (HH:MM) de connexion

-q : n'affiche que les noms d'utilisateur.

-T : Affiche l'état du terminal. Si vous avez un signe +, tous les autres utilisateurs peuvent envoyer des messages à ce terminal. Avec un signe -, c'est impossible. Une liaison de terminal invalide est représentée par ?

Enfin, la commande who peut prendre la forme suivante :

who am i

ou

who am I

Elle indique alors les informations sur l'utilisateur lui-même.

La commande **finger** offre à peu près les mêmes fonctionnalités. Elle possède cependant une fonctionnalité réseau réputée dangereuse puisqu'elle permet d'afficher les noms de login utilisés sur d'autres machines.

3 AVOIR LES ADRESSES IP DES GENS QUI SE CONNECTENT SUR UNE MACHINE LINUX

Première méthode : commande **lastlog** (login, date...) ou **lastlog -t days** qui donne les logs depuis days.
Deuxième : commande **last** qui exploite le binaire **/var/log/wtmp**
Exemple : **last -10a** : donnant les 10 dernières entrées

4 STATISTIQUES D'UTILISATION DES RESSOURCES

La commande **vmstat** permet d'afficher les statistiques sur l'utilisation de la mémoire virtuelle.
Elle offre entre autres des informations sur les processus, la pagination, la mémoire, les entrées/sorties disques, les interruptions et l'activité du processeur.

```
# vmstat
   procs                  memory    swap        io     system        cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy  id
 0  0  0      0   2932  26564  80948   0   0    24     7  330   747   4   1  95
```

Procs :
- r : nombre de processus en attente de lancement.
- b : nombre de processus en sommeil
- w : nombre de processus « swappés »

Memory :
- swpd : mémoire vituelle utilisée (Ko)
- free : mémoire libre (Ko)
- buff : mémoire utilisée en tant que buffer (Ko)

Swap :
- si : mémoire swappée à l'entrée du disque (Ko/s).
- so : mémoire swappée à la sortie du disque (Ko/s)

IO :
- bi : blocs envoyés d'un bloc device (blocs/s)
- bo : blocs recus d'un blocc device (blocs/s).

System :
- in : nombre d'interruptions par seconde
- cs : nombre de switchs par seconde

CPU :
- us : temps utilisateur (pourcentage du temps CPU)
- sy : temps système (pourcentage du temps CPU)
- id : temps inutilisé (pourcentage du temps CPU)

Autre commande utile : **free** :

```
$ free
                     total       used       free     shared    buffers     cached
Mem:             196128     193352       2776          0      16856      89688
-/+ buffers/cache:      86808     109320
Swap:            136040          0     136040
```

5 UTILISATION DES DISQUES

5.1 Affichage de la taille des répertoires

La commande **du**.

Exemples :

1)Connaitre l'occupation respective de chaque sous répertoire du répertoire courant :
du * -sx 2>/dev/null

2)Affiche la taille totale du répertoire courant avec un affichage compréhensible :
du -sch .

5.2 Taux d'occupation du système

La commande **df**.

Exemples :
df : donne l'espace disque disponible sur tous les systèmes de fichiers montés (par défaut la taille est indiquée en nombre de blocs de 1024 octets), ainsi que les points de montage
df -h : affiche la taille en multiples du Ko (Mo, Go)
df -T : rajoute dans l'affichage le type de système de fichier (ext2, vfat, ...)
df -t ext2 : affiche les informations concernant uniquement les partitions du type précisé (ici ext2)
df -x ext2 : exclut les partitions du type précisé
df . : pour connaître l'espace restant dans la partition contenant le répertoire courant

Récupération d'un Système de Fichiers

TABLE DES MATIERES

1 INTRODUCTION

D'autres systèmes nous l'auraient presque fait oublier ;-) , mais l'intégrité du système de fichiers est une des principales fonctions que doit opérer un OS.
Normalement UNIX/LINUX assure l'intégrité du système de fichiers tout seul comme un grand.
Mais des erreurs (le plus souvent : arrêt brutal) peuvent entraîner des incohérences. Les ¾ de ces erreurs peuvent être traitées à l'aide de la commande **/sbin/fsck** (*file system check*). Cette commande possède des soeurs : **fsck.ext2** aussi nommée **e2fsck** qui est dédiée à la manipulation des partitions de type EXT2 utilisées par LINUX.

Si cette commande trouve une erreur dans la structure du système de fichiers, une opération de réparation débute. C'est le premier moyen pour résoudre les erreurs systèmes les plus courantes.

2 LE CONCEPT D'INTEGRITE DU SYSTEME DE FICHIERS

Sous UNIX/LINUX, le système de fichiers essaie toujours d'accélérer les opérations en utilisant des algorithmes simples et en stockant sur le disque dur les entrées/sorties. Les données nécessaires se rechargent ensuite depuis le disque dur vers la mémoire, elles y sont modifiées puis re transférées vers le disque dur. Il peut donc exister des différences entre les données en mémoire centrale et celles stockées sur le disque dur, plus anciennes. Si des modifications sont apportées à ce fichier, les données d'identification de l'inode en mémoire seront différentes de celles de l'inode du disque dur.
Si dans ces circonstances, une erreur a lieu (coupure de courant, ...) et empêche la retranscription de l'inode de la mémoire sur le disque dur, les informations en mémoire sont perdues et le système de fichiers devient inconsistant.

3 LES COMMANDES /SBIN/FSCK ET /SBIN/E2FSCK

Pour mettre à jour ces incohérences et les réparer, on va utiliser en premier secours la commande **/sbin/e2fsck sous LINUX (fsck sous UNIX)**.
Tout d'abord quelques mises en garde : ce programme va devoir manipuler votre système de fichiers à un niveau assez bas, vous devez donc être root pour effectuer toute réparation ou opération de récupération. Et en tant que **root**, l'erreur dramatique est hélas très vite arrivée. Ainsi je vous conseille de prendre garde à bien lire les questions posées par **e2fsck/fsck** avant de répondre trop rapidement, et évitez aussi de jouer les killers avec les options de la commande.

e2fsck

Syntaxe : **e2fsck [options] Fichier-de-périphérique**

Fichier-de-périphérique = Device = /dev/hda1 (pour par exemple la première partition du 1er disque ide)

Options :

-b : Permet d'indiquer à fsck quel superblock utiliser à la place du superblock normal. Ceci est très utile si le premier superblock par défaut est endommagé. La plupart des systèmes de fichiers ont le premier superblock position en 8193, 16385, ... Si un superblock « de substitution » est utilisé et que le système de fichier n'est pas ouvert en lecture seule, e2fsck s'assurera d'abord que le superblock « de substitution » est en bon état.

-B : Habituellement, fsck cherche des superblocks de tailles différentes afin de choisir le plus approprié. Grâce à cette option, vous pouvez le forcer à choisir un superblock de taille précise. Si ce dernier n'est pas trouvé, e2fsck s'arrêtera avec une erreur fatale.

-c : demande à e2fsck d'analyser la surface du disque dur afin de marquer les blocks défectueux.

-C : Permet d'afficher une barre de défilement représentant le % d'analyse de la partition.

-d : Affiche des informations de debbugage.

-f : Force la vérification du système de fichier, même si celui-ci semble impeccable. C'est cette option qui est utilisée tous les X reboot du système. En effet, à chaque reboot le processus /sbin/init lance fsck via le fichier de script /etc/rc.sysinit. Là, un compteur vérifie le nombre de reboot effectué. Si ce nombre = X, alors il y a un scan forcé.

-l filename : Ajoute à la liste des blocs défectueux ceux inscrits dans le fichier donné en paramètre.

-L filename : Met les blocks défectueux dans la liste des blocks spécifiée par le fichier mis en paramètre. Cette option est la même que -l, sauf qu'ici la liste de blocks est vidée avant que les blocks listés dans le fichier ne soient ajoutés à la liste des blocks défectueux.

-n : N'effectue aucune modification sur le système de fichier, et ne l'ouvre qu'en lecture seule.

-p : Répare sans poser de question...

-s : Si l'ordre des octets, dans le fichier n'est pas en little endian, alors e2fsck le modifie. Dans le cas contraire, rien n'est fait.

-S : Inverse l'ordre des octets quelque soit l'ordre courant (little ou big endian)

-a : Répare automatiquement le système de fichier sans poser de questions. Notons que e2fsck supporte cette option uniquement pour assurer la compatibilité ascende vis -à-vis de fsck, mais qu'il est recommandé d'utiliser l'option -p.

-t : Affiche le temps passé à effectuer les opérations.

-v : Verbose.

-V : indique la version du programme et sort sans pratiquer la moindre opération.

-y : Fonctionne en mode non-intéractif en répondant à toutes les questions par yes.

Notes :

-e2fsck fonctionne par défaut en mode intéractif. Pour le rendre non-intéractif, il faut utiliser soit l'option -p ou -a si vous voulez que les erreurs soient corrigées automatiquement, ou -n sinon)

- dans le cas des options -c, -l et -L, les corrections sont réellement apportées. Le système est donc monté en lecture et écriture. DONC PRUDENCE!

- Dans le cas d'une réparation, on voit apparaître le terme de Pass (passe en français!) dans la phase de d'analyse. Les numéros de passe sont les suivants :
1 : Vérification des inodes, blocs,
2 : Vérification de la structure des répertoires
3 : ?
4 : Vérification des références

fsck

Syntaxe : **fsck [options] Fichier-de-périphérique**

Options :

-s : Exécute en série les opérations de scan. Cette option est très intéressante dans le cas d'une analyse de plusieurs systèmes de fichiers. Il faut aussi que la vérification soit en mode interactif.

-t fstype : Spécifie le type de système de fichier à scanner.

-A : Pour limiter le check aux partitions mentionnées dans /etc/fstab.

-C : Pour afficher la barre de progression du scan.

-N : Pour juste vérifier (en lecture seule) ce qui pourrait être fait en cas de scan « réparateur ». Cette option est SUPER utile. Je vous conseille de l'utiliser en priorité pour bine vérifier si vous avez compris la réelle nature du problème en court.

-P : Cette option combinée avec -A, permet de faire un scan simultané de la partition root et des autres partitions. Cette option n'est pas trop conseillée...

-R : Pour ne pas checker la partition root (qui dans ce cas est déjà montée en lecture/écriture.

-a : Répare automatiquement le système de fichiers sans poser de questions. Cette option est à manipuler avec grande précaution. Notons que e2fsck support aussi l'option -a uniquement pour une compatibilité ascendante. Pour e2fsck il est conseillé d'utiliser l'option -p.

-r : Mode interactif.

Petite explication supplémentaire
Les options -s et -S de e2fsck mentionnent le terme d' « endian ».
Quel rapport avec les pingouins ?
Lorsque l'ordinateur doit enregistrer une information tel qu'un entier (par exemple) sur plusieurs octets, il doit définir dans quel ordre il place ces octets.

Conseil final
NE MODIFIER UNE PARTITION QUE SI VOUS L'AVEZ DEMONTEE PREALABLEMENT!!!

Exemple :
Vous en conviendrez, c'est pas facile de donner un exemple de crash du système.
Mais en voici quand même un vrai...

Descriptif :
Sur un Pentium 166Mhz avec 64 Mo de RAM, j'ai lancé la lecture de 1 DIVX + 1 MPEG + compilation de mplayer.
Puis...un saut de tension EDF m'a tout fait planter.

Au reboot :

...
Partition check :
hda: hda1 hda2 hda3
...
VFS: Mounted root (ext2 filesystem) readonly
Freeing unused kernel memory: 236k freed
INIT
...
Checking root filesystem
/ contains a file system with errors, check forced.
/:
Duplicate or bad block in use!
/: Duplicate block found...invoking duplicate block passes.
Pass 1B: Rescan for dupicate/bad blocks
/: Duplicate/bad block(s) in inode 49667:/ 120924/:
/: Duplicate/bad block(s) in inode 49806:/ 120924/:
/: Pass 1C: Scan directories for inodes with dup blocks.
/: Pass 1D: Reconciling duplicate blocks
/: (there are 2 inodes containing duplicate/bad blocks.)

/: File /home/amethys/Desktop/.directory (inode #49806, ...)
has 1 duplicate block(s), shared with 1 file(s):
/: /home/amethys/.xsession-errors (inode #49667 ...)

RUN fsck MANUALLY
(i.e, without -a or -p options)

A ce stade vous vous retrouvez avec un shell en tant que root.

La suite est simple :

J'ai lancé un **e2fsck -y /dev/hda3**

Tout est rentré dans l'ordre.

4 DANS LES CAS DURS

Dans les cas difficiles, ou quand votre soucis n'est pas d'ordre block/inode/Indiens, ..., il vous faudra passer par l'incontournable disquette de récupération **TOMSRTBT** (http://www.toms.net/~toehser/rb/).
Il s'agit en fait d'une distribution de poche (1 disquette) qui va vous permettre d'avoir une console et pleins de petits utilitaires très pratiques.
Attention, pour utiliser cette mini-distribution, il faut ABSOLUMENT que vous ayez sur votre système un noyau compilé avec le support **ELF**.
Pour le vérifier :
cd /usr/src/linux
make xconfig

Section « General Setup » :
L'option « Kernel support for ELF binaries ».

Rappel :
ELF (Executable and Linkable Format) est un format pour les bibliothèques et exécutables utilisés sur différentes architectures et systèmes d'exploitations.

Une fois que **tomsrtbt** a démarré et que vous avez une console :
Vous pouvez par exemple :
- Monter un lecteur zip (en read-only) :
 # mount /dev/sda1 /mnt/zip -o ro
- Effacer les deux premiers secteurs de votre disque dur :
 # dd if=/dev/zero of=/dev/hda bs=512 count=2
Ceci remplit de zéros les deux secteurs de boot (MBR), cela détruit donc toutes les informations de partitions et tout code de démarrage, comme le code LILO par exemple.
Ainsi vous pourrez repartitionner votre disque si nécessaire, monter les partitions existantes, relancer lilo, ...
Rappel utile :
* pour créer une partition EXT2 : **mke2fs /dev/hdaX** ;
* pour la partition swap, IL FAUT UTILISER UNIQUEMENT : **mkswap /dev/hdaY**.

5 BOOTER EN MODE SINGLE (INIT 1)

Ce mode est utile pour surtout dans deux cas de figure :

Le système plante lors du passage au runlevel 2 ou supérieur :

Relancez la machine et stoppez sur le prompt **Lilo boot:** puis tapez **linux init=/bin/bash** (ou **sh**)

Ce mode dit Single permet d'atteindre le shell sans avoir à se logger (sauf si la configuration du lilo spécifie l'utiliation d'un mot de passe pour le mode single...).

On a alors un seul process qui tourne : **/bin/bash**. Il faut donc se faire tous les **mount** à la main.

Il est nécessaire aussi de lance **/etc/rc.d/init.d/keytable start** pour avoir le clavier en azerty!!!

Oublis du mot de passe root :

Relancez la machine et stoppez sur le prompt **Lilo boot:** puis tapez linux **init=/bin/bash** (ou **sh**)

Ce mode dit Single permet d'atteindre le shell sans avoir à se logger (sauf si la configuration du lilo spécifie l'utiliation d'un mot de passe pour le mode single...).

Editer le fichier **/etc/shadow**

Supprimer les caractères bizarres qui se trouvent entre **root:** et les **:** suivants.

Ces caractères bizarres correspondent au mot de passe crypté.

Et voila, vous pouvez vous logger à nouveau normalement sans avoir à taper de mot de passe.

Puis créer un nouveau mot de passe pour le root en tapant la commande : **passwd root**

Note : on peut aussi utiliser la commande suivante pour démarrer en mode mono utilisateur avec un shell en root : **Lilo boot : linux –s**

PROFTPD

ProFTPD

Highly configurable GPL-licensed FTP server software

Site officiel :
http://www.proftpd.org/
Télécharger:
ftp://ftp.easynet.be/proftpd/distrib/packages/RPMS/

Proftpd est un logiciel conçu pour créer un serveur ftp.

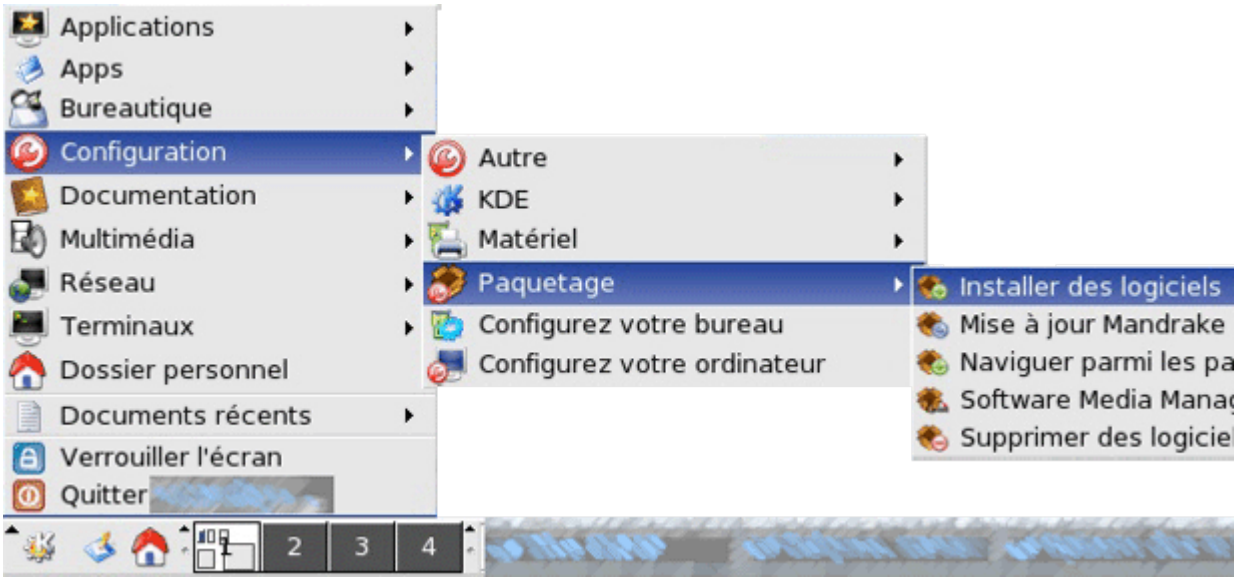
Table des matières

Sur les distributions Mandrake, le paquetage est peut être déjà installé.

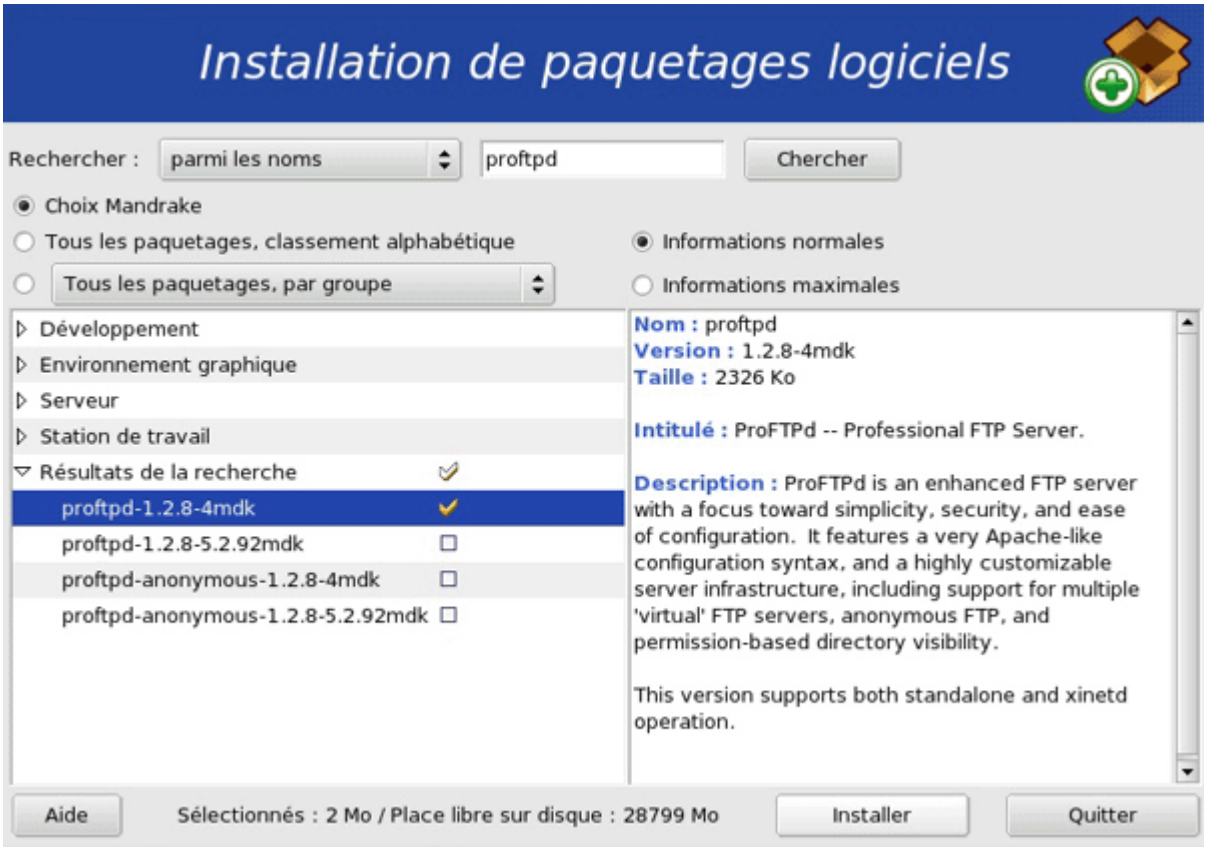
Pour vérifier : `rpm -q proftpd` --> *proftpd-1.2.8-4mdk*

1 INSTALLATION DE PROFTPD - PAR RPM ET MODE GRAPHIQUE

Il y a différente manière de l'installer.
Commençons par la démarche d'installation la plus simple, c'est-à-dire par fichier dit RPM et par mode graphique.
Donc, je vous invite à à lancer votre interface graphique (Par défaut KDE). Ensuite allez dans le menu chercher "Installer des logiciels" comme montre la photo ci-dessous :



Une fois cliqué sur "Installer des logiciels", une nouvelle fenêtre s'ouvre dans laquelle vous taperez "profptd" dans le champ rechercher puis cochez sa case correspondante et terminer par cliquez sur "Installer" comme l'indique l'image ci-dessous:



Proftpd est installé.
Une fois installé, le fichier de configuration principal est "/etc/proftpd.conf".

Vous pouvez paramétrer ce fichier par console grâce à la commande :

"**vi /etc/proftpd.conf**"

2 INSTALLATION DE PROFTPD - PAR RPM ET MODE CONSOLE

Par la console, ce n'est pas plus compliqué.

Télécharger le fichier RPM de Proftpd (ou bien prendre celui du CD de la distribution)

Une fois téléchargé, allez dans le répertoire ou se trouve le fichier RPM (En général, il se trouve dans /home/*nom-utilisateur-actuel*.

Tapez la commande :
rpm -ivh proftpd-1.2.8-5.2.92mdk.i586.rpm puis entrer.

S'il y a un conflit détecté avec le serveur *wu-ftpd*, le désinstaller au préalable :

```
rpm -e wu-ftpd
```

Le résultat lors de l'installation devrait ressembler à quelque chose comme ça :

```
Preparing : ##################### [100%]

Proftpd : ##################### [100%]
```

Si l'installation est refusée, connectez-vous en utilisateur "**root**" (Pour cela, tapez su puis entrez le mot de passe root)
ATTENTION : en ROOT vous avez tous les droits, même de casser le PC !

Pour Info :

- L'option "**i**" affiche des information concerant le RPM.
- L'option "**v**" permet d'embellir l'affichage.
- L'option **"h"** permet de faire des marque de hachage.

Une fois installé, differents fichiers se sont inscrits, ainsi vous trouverez :

- Le fichier executable dans **/sbin/proftpd**
- Le fichier de log dans **/var/log/proftpd/***
- Le fichier de configuration dans **/etc/proftpd.conf**

Rebooter le PC et normalement , proftpd devrait se lancer automatiquement.

3 INSTALLATION DE PROFTPD - PAR COMPILATION ET MODE CONSOLE

Précisons tout d'abord que la compilation d'un logiciel ne requiert la connaisssance d'aucun langage de programmation.
Compiler un logiciel et l'installer requiert, dans la très grande majorité des cas, six commandes :

- Le décompresser grâce à la commande : "**tar -zxvf proftpd-1.2.8-5.2-92mdk.i586.rpm"**
- Entrer dans le répertoire ou il a été décompressé : "**cd proftpd-1.2.8-5.2-.....**"
- Tapez ensuite **/configure**
- Puis tapez "**make**"
- Tapez cette fois "**su**" (puis passe root)
- Et enfin tapez "**make install**".

La commande tar zxvf nom_du_logiciel.tar.gz permet de décompresser une archive (remplacer z par I si l'archive est au format bz2). Ensuite, il faut se rendre dans le répertoire créé par cette opération de désarchivage / décompression. Les sources comportent toujours un fichier README expliquant la procédure d'installation.

Il est rare d'avoir à taper d'autres commandes que ./configure, make et make install. N'oubliez pas d'installer les petits utilitaires automake, autoconf, yacc, flex et bison, qui facilitent le travail des scripts. Tout cela est en principe présent dans votre distribution, sur le CD-Rom, dans la rubrique développement.
Souvent, il vous sera indiqué que tel fichier .h est introuvable, cela signifie tout simplement que la librairie de développement n'est pas installée. Procédez à cette installation avec le gestionnaire de packages de votre distribution, qui inclut normalement ces librairies.

Les avantages de la compilation manuelle sont principalement au nombre de deux, si vous n'êtes pas programmeur :

- La possibilité de tester un logiciel avant son installation. En effet, le logiciel compilé est présent sous forme binaire dans le répertoire de compilation (ou dans le sous répertoire src). On peut ainsi le lancer (commande ./son-nom) et le tester avant de

l'installer, sauf s'il requiert des librairies qui ont été compilées et doivent être installées en même temps que lui.

- La possibilité d'installer le logiciel ailleurs que dans son répertoire par défaut. C'est le paramètre prefix du Makefile qui permet cela. Par exemple, la commande ./configure --prefix="/home/mon_repertoire/gwget" permet d'installer gwget dans votre repertoire personnel, dans le sous répertoire gwget, ce qui peut être parfois utile. Il est donc possible de vérifier en quelle manière le système sera modifié avant d'installer quoique ce soit. La commande ./configure --help permet d'avoir la liste des options de la compilation.

Si toutefois, vous avez des messages d'erreurs vous signalant un manque de fichiers installer, n'oubliez pas : "**gcc**", "**autoconf**" et "**automake**" à installer.

Il ne reste plus qu'a configurer le fichier proftpd.conf

4 CONFIGURATION DE PROFTPD - PAR MODE CONSOLE

Proftpd possède un seul fichier de configuration ce qui est pratique .Il est possible de configurer plusieurs serveurs FTP virtuels et également de les établir en anonymes, bref pas mal de possibilités pour un serveur FTP mais nous allons voir tous ca.

Nous allons pour commencer voir la disposition du fichier de configuration.

Pour cela, tapez "**vi /etc/proftpd.conf**" dans une console .Autant s'habituer à la console de suite !

Voici le fichier tel qu'il est par défaut :

 # This is a basic ProFTPD configuration file (rename it to
 # 'proftpd.conf' for actual use. It establishes a single server
 # and a single anonymous login. It assumes that you have a user/group
 # "nobody" and "ftp" for normal operation and anon.

ServerName "ProFTPD Default Installation"
ServerType standalone
DefaultServer on

 # Allow FTP resuming.
 # Remember to set to off if you have an incoming ftp for upload.
AllowStoreRestart on

 # Port 21 is the standard FTP port.
Port 21

 # Umask 022 is a good standard umask to prevent new dirs and files

 # from being group and world writable.
Umask 022

 # To prevent DoS attacks, set the maximum number of child processes
 # to 30. If you need to allow more than 30 concurrent connections
 # at once, simply increase this value. Note that this ONLY works
 # in standalone mode, in inetd mode you should use an inetd server
 # that allows you to limit maximum number of processes per service
 # (such as xinetd).
MaxInstances 30

 # Set the user and group under which the server will run.
User nobody
Group nogroup

 # To cause every FTP user to be "jailed" (chrooted) into their home
 # directory, uncomment this line.
#DefaultRoot ~

 # Normally, we want files to be overwriteable.

AllowOverwrite on

 # Needed for NIS.

PersistentPasswd off

 # Default root can be used to put users in a chroot environment.
 # As an example if you have a user foo and you want to put foo in /home/foo
 # chroot environment you would do this:
#
 # DefaultRoot /home/foo foo

Voici les principales options à ne pas manquer:

- **ServerName** qui correspond au nom du serveur affiché par le FTP Client.
- **ServerType** qui correspond au mode de démarrage de Proftpd, mettez "StandAlone".
- **DefaultServer**, à mettre en général sur On, il s'agit d'indiquer sur quels serveurs FTP doit se diriger une connexion entrante étant donné qu'il y a en général qu'un seul serveur FTP, il sera par défaut.
- **Port** Il s'agit bien évidemment du port de connexion sur le serveur FTP.En général le port donné à un serveur FTP est le port 21.

- **Umask** Cette valeur représente un mask pour les permissions des fichiers et dossiers nouvellement crées. 022 par défaut.

- **MaxInstances** Cette option ne fonctionne qu'en mode StandAlone et vous prévient des attaques DDos, c'est le nombre maximum de processus lancé en même temps par le serveur. 30 est déjà très largement suffisant pour les petits et moyens serveurs FTP.

- **AllowStoreRestart** Permet d'accepter ou de refuser à un client de continuer ce qu'il avait déposé sur votre serveur.En général l'option par défaut est On.

- **User** et **Group** vont ensembles. Ce sont les droits que va avoir le serveur FTP sur votre système.Le minimum de droits est le mieux pour éviter tous piratages c'est pour cela que par défaut est indiquer l'utilisateur **User** et le groupe **Group** qui n'ont quasi aucun droits .

- **AllowOverwrite** Sur On si l'on autorise, en cas de même nom de fichier, à écraser l'ancien par le nouveau.

- **PersistentPasswd** Détermine par quel moyen Proftpd va faire les authentifications. Si la valeur est On, alors il lira le fichier **/etc/passwd** (Potentiellement si les passes sont crypté le fichier **/etc/shadow**) et **/etc/group**

N'oublions pas que nous n'avons pas encore indiqué de chemins ou le FTP doit partager les fichiers et nous-y venons.
Nous allons passer à quelques exemples typiques.

5 EXEMPLES DE CONFIGURATION DE PROFTPD.CONF

Exemple typique d'un serveur FTP pour plusieurs personnes ,voulant également partager des fichiers et permettre l'upload:

Avant tout, paramétrons le serveur . Nous commençons par indiquer dans proftpd.conf

#----------

ServerType StandAlone
DefaultServer On

Commençons à personnaliser notre serveur en lui indiquant le nom du serveur que l'on veut afficher au client:
ServerName "SERVEUR FTP LINUX"

Vous pouvez, ensuite lui donner l'emplacement d'un fichier pour qu'il puisse mettre toutes les connections, c'est a dire un fichier log :
SystemLog /var/log/proftpd/log-proftpd

Indiquons lui maintenant le chemin ou vous partagez vos fichiers :
DefaultChdir /var/ftp

Combien de clients connectés simultanément Proftpd peut-il accepter ?:
MaxClients 10

Attention à l'option précédente car un utilisateur peux ouvrir plusieurs session à son nom donc il faut limiter également le nombre de sessions par utilisateur connectés (1 suffit, 2 pour la double-connection que beaucoup de clients FTP utilisent):
MaxClientsPerHost 1

L'option suivante est importante, elle permet de refuser les connections sur le serveur s'il s'agit de l'utilisateur ROOT et permet une sécurité supplémentaire :
RootLogin Off

Encore une option importante dans ce cas de serveur FTP. Sachant que vous allez créer des comptes pour des amis ou autres, il faudra préciser que ce sont des "faux" comptes, c'est à dire des compte qui ne sont pas "local" (question de sécurité également) :
RequireValidShell Off

Fameuse option qui permet à un utilisateur de continuer son upload :
AllowStoreRestart On

La même qui permet de continuer son Download:
AllowRetrieveRestart On

Cette option vous permet de limiter le nombre de tentatives de connexion au serveur en cas de login / mot de passe incorrect :
MaxLoginAttempts 2

Un peu de FUN, un message de bienvenue qui s'inscrit sur le client FTP en cas de connexion acceptées :
AccessGrantMsg "BIENVENUE SUR LE SERVEUR FTP , %u ..."

La même chose mais en cas de refus de connexion:
AccessDenyMsg "Nous sommes désolés %u, vous n'avez pas accès à ce serveur ..."

Petite option préventive, elle évite de donner des infos du serveur FTP:
DeferWelcome On

Evidemment, le port de communication à préciser :
Port 21

Le masque pour les fichiers déposés (-rw-r-r--**).:**
Umask 022

Les droits accordés au lancement du serveur :
User nobody
Group nogroup

#--

Beaucoup d'options sont évidemment encore possible.

Exemple pour un accés anonymous

- Il faut créer la section *anonymous* pour que les clients puissent se connecter sans authentification, en fait au nom de l'utilisateur **ftp**, dont le répertoire personnel est **/var/ftp**, et qui n'a pas de shell, comme le confirme l'examen de */etc/passwd*
- Voici une configuration de base, permettant le téléchargement à partir du rép. /var/ftp, et les dépots sans droit de lecture (uploads) dans */var/ftp/depot*

```
<Anonymous ~ftp>
User ftp
Group ftp
UserAlias anonymous ftp
MaxClients 10
RequireValidShell off
AnonRequirePassword off
# pas de droit en écriture en général
<Limit WRITE>
  DenyAll
</Limit>

# le sous-rép. de dépot est en écriture seule
<Directory depot/*>
  <Limit READ>
    DenyAll
  </Limit>
  <Limit STOR>
    AllowAll
  </Limit>
</Directory>
</Anonymous>
```

6 ANNEXES

- Pour limiter l'accès des clients du serveur à leur répertoire personnel, décommenter la dernière ligne et la modifier :
- `DefaultRoot ~`

  Puis relancer le service : *service proftpd reload*, se reconnecter au serveur et vérifier.

killall -HUP xinetd. Pour relancer xinetd..
Le service de Proftpd se trouve dans */etc/xinetd.d/proftpd-xinetd*)

tail –f /var/log/proftpd pour voir les log en live

Programmation SHELL

1 PROGRAMMATION SHELL

Le shell est un ensemble de commande et constitue un véritable langage de programmation. Il permet d'écrire des utilitaires , permettant l'aide à la maintenance, à effectuer des taches…

Le shell le plus connu est le bash (Bourne Again SHell) et le plus utilisé car il est facile d'utilisation, libre de droit et gratuit.

2 STRUCTURE ET EXECUTION D'UN SCRIPT

Les différentes instructions et commandes sont réunis dans un fichier script.
Ce fichier script est un simple texte qui peut être édité par un éditeur comme vi, emacs, kedit…

Par convention, un script commence par une ligne de commentaire qui contient le nom du langage que l'on va utiliser.

Ainsi, on précise quel shell va exécuter le script.

Ex :
```
#!/bin/sh
#!/bin/ksh
```

Dans le premier cas, on utilisera le Bash shell, et dans l'autre le Korn shell.

Les commentaire commence toujours par « # »

```
#nous allons faire un ls
ls  #ceci est la commande ls ;
```

Le « ; » permet de signaler la fin d'une intruction dans une ligne.
Le fichier script est enregistré généralement avec l'extension « sh » pour shell script.

Pour exécuter le script :
```
sh monscript.sh
```

Pour le rendre exécutable :
```
chmod +x monscript.sh
```

Pour le lancer, taper :
```
./monscript.sh
```

Pour éviter le ./ , il faut définir le chemin ou se trouve le script dans PATH
```
PATH=$PATH :.
```

3 LE PASSAGE DE PARAMETRES

Il peut être utile de spécifier un ou des arguments à un script pour en adapter ou modifier son comportement.

Le ou les paramètres à « passer » sont placés après le script sur la ligne de commande.
Ex:

```
$ cat test1.sh
#!/bin/sh
echo le paramètre \$1 est $1
echo le paramètre \$2 est $2
echo le paramètre \$3 est $3

./test1.sh aaa bbb 123
```

Résultat :

```
le paramètre $1 est aaa
le paramètre $2 est bbb
le paramètre $3 est 123
```

Les variables $1 , $2 … $9 contiennent des mots passés sur la ligne de commande.
Ce mot peut être une chaîne de caractères avec des caractères de séparations tels que l'espace, le point virgule et la tabulation, mais ce mot devra être mis entre des guillemets.

Ex : ./test1.sh "hello everybody ; bonjour" aaa bbb

Vous pouvez voir aussi que nous avons placé le caractère \ avant $1 $2 et $3 pour avoir \$1 \$2 \$3

En effet, sans ce caractère, la variable aurait été interprété et nous aurions eu la valeur.
Le caractère \ permet de « protéger » ou « échapper » le caractère qui le suit. Ainsi ,le caractère n'est pas interprété.

Ex :
Pour avoir le résultat de la forme "aaa" "bbb" "ccc" , nous allons placer les guillemets dans le script :

```
#!/bin/sh
echo le paramètre \$1 est "$1"
echo le paramètre \$2 est "$2"
echo le paramètre \$3 est "$3"
```

Cela ne fonctionnera pas car les guillemets vont être interprétés par la commande echo
Il font les échapper.

```
#!/bin/sh
echo le paramètre \$1 est \"$1\"
echo le paramètre \$2 est \"$2\"
echo le paramètre \$3 est \"$3\"
```

Maintenant, lancer le script avec * pour argument.

```
./test *
```

Vous pouvez constater que le shell remplace le caractère * par la liste des fichiers du répertoire courant.

En fait, toute substitution est possible.

Les caractères spéciaux :	*	toute suite de caractères
	?	un caractère quelconque (Ex : ??a ?? *a ? ??x*)
	[bwe]	l'un des caractères b, w ou e
	[d-z]	les caractères de d à z

Un autre exemple :

```
./test \*
```

Quel sera le résultat ? Pourquoi ?

Comment faire avec plus de 9 arguments ?

Il faut utiliser la commande *shift* qui permet de décaler vers la droite les arguments.
$1 reçoit le deuxième argument
$2 reçoit le troisième argument
…
$9 reçoit le dixième argument

EX :

```
#!/bin/sh
echo le paramètre 1 est \"$1\"
shift
echo le paramètre 2 est \"$1\"
shift
echo le paramètre 3 est \"$1\"
shift
echo le paramètre 4 est \"$1\"
shift
echo le paramètre 5 est \"$1\"
shift
echo le paramètre 6 est \"$1\"
shift
echo le paramètre 7 est \"$1\"
shift
echo le paramètre 8 est \"$1\"
shift
echo le paramètre 9 est \"$1\"
shift
echo le paramètre 10 est \"$1\"
shift
echo le paramètre 11 est \"$1\"
```

```
./test 1 2 3 4 5 6 7 8 9 10 11
```

Résultat :

```
le paramètre 1 est "1"
le paramètre 2 est "2"
le paramètre 2 est "3"
le paramètre 4 est "4"
le paramètre 5 est "5"
le paramètre 6 est "6"
le paramètre 7 est "7"
le paramètre 8 est "8"
le paramètre 9 est "9"
le paramètre 10 est "10"
le paramètre 11 est "11"
```

Il faut remarquer qu'une variable est perdu pour chaque SHIFT
Pour pouvoir les conserver, il faudra utiliser des variables.

Récapitulatif :

$# : donne le nombre d'argument passé
$0 : donne le nom du script lui même;
$1 à **$9** : donne les 0 premiers arguments passés; s'il y en a plus de neuf, il faut utiliser la commande *shift* qui décale sur la droite les arguments : $1 reçoit $2, ..., $9 reçoit le dixième argument;
$* : donne les n paramètres passés en 1 chaîne de caractères;
$@ : donne les n paramètres passés en n chaînes de caractères.

4 LES VARIABLES

Nous avons vu les variables spéciales pour passer des paramètres à un script.

4.1 Définir vos propres variables, appelés variables utilisateurs

Ex :
$ message=hello

Une variable est déclarée dès qu'une valeur lui est affectée.
L'affectation est effectuée par = , sans espace avant ou après.

On peut déclarer une variable vide :
Ex :

$ message=

Le premier caractère ne peut être un chiffre.
Vous pouvez utiliser des chiffres, des lettres majuscules ou minuscules, des « underscore » .
Par convention, on utilise des minuscules, pour différencier les variables utilisateurs des variables système.
La taille du nom est illimitée.

4.2 Afficher le contenu d'une variable

Pour afficher le contenu d'une variable, il suffit de la précéder du caractère $.

```
$ message=allo
$ echo $message
```

```
$ chemin=/var/log
$ ls $chemin
```

TP

Taper la commande

echo $PATH

Vous pouvez deviner que PATH est une variable déjà définie.
Cette variable contient la liste des répertoires qui indique au shell où trouver les programmes.
Les répertoires sont séparés par « **:** »

Ecrire une commande permettant de rajouter le répertoire de votre homedirectory dans le PATH.

R : (Pour le homedirectory de l'utilisateur user1)

```
# ajout au contenu de PATH le chemin /home/user1
PATH=$PATH:/home/user1
```

Modifiez cette commande pour permettre à l'utilisateur de lancer tout programme du répertoire dans lequel il se trouve.

Ecrire un script incluant cette commande, en affichant le contenu de PATH avant/après avec la commande echo

Vérifier le contenu de PATH , que constatez vous ?

Voir la commande export

4.3 Les variables d'environnement

Nous avons vu les variables utilisateurs définies par lui-même. Le shell est lancé avec des variables prédéfinies. Elle sont accessibles pour les utilisateurs.
Le contenu de ces variables sont modifiables par l'utilisateur, en prenant toutefois attention car certaines d'entre elle peuvent avoir une incidence sur le comportement du système.

`$HOME`	le homedirectory de l'utilisateur.
`$PATH`	liste séparée par ':' des noms répertoires inspecté par le shell lors de la recherche d'un fichier exécutable
`$PS1`	invite primaire
`$PS2`	invite secondaire, défaut '>'
`$PWD`	répertoire courant
`$OLDPWD`	répertoire précédent le dernier `cd`
`$IFS`	séparateur de champs *Internal Field Separtor*, défaut SPC, TAB, NL
`$LOGNAME`	nom du login utilisé lors de la connexion.
`$UID`	uid de l'utilisateur
`$EUID`	uid effectif de l'utilisateur
`$TERM`	type de terminal
`$LANG`	langue utilisée
`$MAIL`	chemin et fichier contenant les messages de l'utilisateur
`$MAILCHECK`	intervalle en secondes pour la vérification de présence d'un

	nouveau courrier. Si 0 alors la vérification est effectuée après chaque commande.
$DISPLAY	serveur X11
$SHELL	chemin complet du shell en cours d'exécution
$SHLVL	compteur incrémenté de 1 à chaque fois qu'un shell est démarré
$USER	nom de l'utilisateur en cours.

Pour avoir la liste des variables, on utilise la commande set . Elle liste la liste des variables utilisateurs et systèmes ainsi que son contenu.

4.4 Exportation de variables – La commande export

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

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.

TP

Taper les commandes suivantes :

```
a=messagedeserge
echo $a
```

Le message s'affiche

Ecrire le script suivant testvar.sh et lancer le:

```
# !/bin/sh
echo la variable a vaut : $a
```

Que constater vous ?

Taper la commande export a

```
export a
```

relancer le script
Que concluez vous?

4.5 Suppression et protection de variables

On peut supprimer une variable avec la commande unset.

On protège une variable en écriture par la commande readonly. Une fois protégé, il est impossible de la repasser en écriture, encore moins de la supprimer. Le seul moyen sera de quitter le shell.

TP

a=serge
b=avrillon
echo $a $b

unset b
echo $a $b
La variable b n'existe plus

readonly a

Testons la réécriture de la variable a
a=xenon

Testons la suppression de la variable
unset a

4.6 Saisir la valeur d'une variable

Il peut être intéressant de saisir une valeur pendant l'exécution du script.
Ceci est réalisable grâce à la commande read .

TP

Ecrire un script permettant de saisir le login et le mot de passe.

```
# !/bin/sh
echo "Entrez votre pseudo"
read pseudo
echo "Entrez votre mot de passe"
read password
echo "Votre pseudo est $pseudo et votre mdp est $password"
```

Pour le mot de passe, utilisez l'option –s de read. Ainsi la saisie du mot de passe ne sera pas apparent.

5 LES ACCOLADES

Les accolades de base « {} » permettent d'identifier le nom d'une variable.

TP

Créer le fichier liste1.
Créer une variable fichier contenant le texte liste.
Le but est d'utiliser le contenu de la variable fichier pour copier le fichier liste1 vers 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.

6 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
rien est affiché, car la variable nom n'a pas été définie

$ echo ${nom:-serge}
serge est affiché…

$ echo $nom
mais la variable nom n'a pas été définie

$ echo ${nom:=serge}
serge est affiché

$ echo $nom
serge est affiché car la variable nom est définie

$ echo ${nom:+"Valeur définie"}
Valeur définie est affichée

$ unset nom

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

$ nom=serge

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

7 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}
```
Linux
```
$ machine=`uname -a | cut -d" " –f12`
$ echo $machine
```
intel

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

8 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.

8.1 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
```

Remarque :

Il existe une syntaxe allégée. On remplace le mot test par des crochets. Il faut respecter un espace avant et après les crochets.

Ex :

```
$ a=Jules
$ [ "$a" = Jules ] ; echo $?
0
```

8.2 Tests sur valeurs numériques

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

test valeur1 option valeur2

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
```

8.3 Tests fichiers

La syntaxe est test option nom_fichier
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

EXEMPLE :

```
$ 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
```

8.4 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" && echo "rep1: repertoire, droit en ecriture"
rep1: repertoire, droit en ecriture
```

En écriture abrégée:

```
$ [ -d "rep1" –a –w "rep1" ] && echo "rep1: repertoire , droit en ecriture”
```

9 IF ... THEN ... ELSE

La structure if then else fi est une structure de contrôle conditionnelle.

if
then

else

fi

On peut aussi préciser le elif, en fait un else if. Si la dernière condition n'est pas réalisée on en teste une nouvelle.

EXEMPLE : test de la présence de paramètres

```
$ cat param4.sh
#!/usr/bin/sh
if [ $# -ne 0 ]
then
echo "$# parametres en ligne de commande"
else
echo "Aucun parametre; set valeur1 valeur2 valeur3 valeur4"
set valeur1 valeur2 valeur3 valeur4
fi
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3 4=$4"
echo "Liste : $*"
```

```
$ ./param4.sh titi toto
```
2 parametres en ligne de commande
Nombre de parametres : 2
Parametres : 1=toto 2=titi 3= 4=
Liste : toto titi

```
$ ./param4.sh
```
Aucun parametre; set valeur1 valeur2 valeur3 valeur4
Nombre de parametres : 4
Parametres : 1=valeur1 2=valeur2 3=valeur3 4=valeur4
Liste : valeur1 valeur2 valeur3 valeur4

10 CHOIX MULTIPLES CASE

La commande case esac permet de vérifier le contenu d'une variable ou d'un résultat de manière multiple.

case Valeur in
Modele1) Commandes ;;
Modele2) Commandes ;;
*) action_defaut ;;
esac

Le modèle est soit un simple texte, soit composé de caractères spéciaux.
Chaque bloc de commandes lié au modèle doit se terminer par deux points-virgules.
Dès que le modèle est vérifié, le bloc de commandes correspondant est exécuté.
L'étoile en dernière position (chaîne variable) est l'action par défaut si aucun critère n'est vérifié.

Caractère	Rôle
*	Chaîne variable (même vide)
?	Un seul caractère
[...]	Une plage de caractères
[!...]	Négation de la plage de caractères
|	OU logique

```
$ cat case1.sh
#!/usr/bin/sh
if [ $# -ne 0 ]
then
echo "$# parametres en ligne de commande"
else
echo "Aucun parametre"
exit 1
fi
case $1 in
a*)
echo "Commence par a";;
b*)
echo "Commence par b";;
fic[123])
echo "fic1 fic2 ou fic3";;
*)
echo "Commence par n'importe quoi";;
esac
exit 0
```

```
$ ./case1.sh "au revoir"
```
Commence par a

```
$ ./case1.sh bonjour
```
Commence par b

```
$ ./case1.sh fic2
```
fic1 ou fic2 ou fic3

```
$ ./case1.sh erreur
```
Commence par n'importe quoi

11 LES BOUCLES

Elles permettent la répétition d'un bloc de commandes soit un nombre limité de fois, soit conditionnellement.
Toutes les commandes à exécuter dans une boucle se placent entre les commandes do et done.

11.1 Boucle For

La boucle for ne se base pas sur une quelconque incrémentation de valeur mais sur une liste de valeurs, de fichiers ...

for var in liste
do
commandes à exécuter
done

La liste représente un certain nombre d'éléments qui seront successivement attribuées à var.

Avec une variable

```
$ cat for1.sh
#!/usr/bin/sh
for params in $@
do
echo "$params"
done
```

```
$ ./for1.sh test1 test2 test3
```
test1
test2
test3

Liste implicite

Si on ne précise aucune liste à for, alors c'est la liste des paramètres qui est implicite. Ainsi le script précédent aurait pu ressembler à :

for params
do
echo "$params"
done

Avec une liste d'éléments explicite

```
$ cat for2.sh
#!/usr/bin/sh
for params in liste liste2
do
ls -l $params
done
```

```
$ ./for2.sh
-rw-r--r-- 1 oracle system 234 Aug 19 14:09 liste
-rw-r--r-- 1 oracle system 234 Aug 13 10:06 liste2
```

Avec des critères de recherche sur nom de fichiers

```
$ cat for3.sh
#!/usr/bin/sh
for params in *
do
echo "$params \c"  # \c permet d'omettre le passage a la ligne
type_fic=`ls -ld $params | cut -c1`
case $type_fic in
-) echo "Fichier normal" ;;
d) echo "Repertoire" ;;
b) echo "mode bloc" ;;
l) echo "lien symbolique" ;;
c) echo "mode caractere" ;;
*) echo "autre" ;;
esac
done
```

```
$ ./for3.sh
case1.sh Fichier normal
dump.log Fichier normal
for1.sh Fichier normal
for2.sh Fichier normal
for3.sh Fichier normal
lien_fic1 lien symbolique
lien_fic2 lien symbolique
liste Fichier normal
liste1 Fichier normal
liste2 Fichier normal
param.sh Fichier normal
param2.sh Fichier normal
param3.sh Fichier normal
param4.sh Fichier normal
read.sh Fichier normal
rep1 Repertoire
resultat.txt Fichier normal
toto.txt Fichier normal
voir_a.sh Fichier normal
```

Avec une substitution de commande

```
$ cat for4.sh
#!/usr/bin/sh
echo "Liste des utilisateurs dans /etc/passwd"
for params in `cat /etc/passwd | cut -d: -f1`
do
echo "$params "
done
```

```
$ ./for4.sh
Liste des utilisateurs dans /etc/passwd
root
nobody
nobodyV
daemon
bin
uucp
uucpa
auth
cron
lp
tcb
adm
ris
carthic
ftp
stu
...
```

12 BOUCLE WHILE

La commande while permet une boucle conditionnelle « tant que ».
Tant que la condition est réalisée, le bloc de commande est exécuté. On sort si la condition n'est plus valable.

while condition
do
commandes
done

ou
while
bloc d'instructions formant la condition
do
commandes
done

EXEMPLE :

```
$ cat while1.sh
#!/usr/bin/sh
while
echo "Chaine ? \c"
read nom
[ -z "$nom" ]
do
echo "ERREUR : pas de saisie"
done
echo "Vous avez saisi : $nom"
```

Par exemple une lecture d'un fichier ligne à ligne :

```
#!/usr/bin/sh
cat toto.txt | while read line
do
echo "$line"
done
```

ou

```
#!/usr/bin/sh
while read line
do
echo "$line"
done < toto.txt
```

13 BOUCLE UNTIL

La commande until permet une boucle conditionnelle « jusqu'à ».
Dès que la condition est réalisée, on sort de la boucle.

until condition
do
commandes
done

ou
until
bloc d'instructions formant la condition
do
commandes
done

EXEMPLE :

```
$ cat copie.sh
until cp $1 $2; do
     echo 'Echec de la copie. En attente…'
     sleep 5
done
```

Remarque: la boucle until peut etre convertie en une boucle while en utilisant l'opérateur « ! »

14 SEQ

La commande seq permet de sortir une séquence de nombres, avec un intervalle éventuel.

seq [debut] [increment] fin

EXEMPLE :

```
$ seq 5
1
2
3
4
5

$ seq -2 3
-2
-1
0
1
2
3

$ seq 0 2 10
0
2
4
6
8
10
```

15 TRUE ET FALSE

La commande true ne fait rien d'autre que de renvoyer 0.
La commande false renvoie toujours 1.
De cette manière il est possible de faire des boucles sans fin. La seule manière de sortir de ces boucles est un exit ou un break.

while true
do
commandes
exit / break
done

16 BREAK ET CONTINUE

La commande break permet d'interrompre une boucle. Dans ce cas le script continue après la commande done. Elle peut prendre un argument numérique indiquant le nombre de boucles à sauter, dans le cadre de boucles imbriquées (rapidement illisible).

```
while true
do
echo "Chaine ? \c"
read a
if [ -z "$a" ]
then
break
fi
done
```

La commande continue permet de relancer une boucle et d'effectuer un nouveau passage. Elle peut prendre un argument numérique indiquant le nombre de boucles à relancer (on remonte de n boucles). Le script redémarre à la commande do.

17 LES FONCTIONS

Les fonctions sont des bouts de scripts nommés, directement appelés par leur nom, pouvant accepter des paramètres et retourner des valeurs. Les noms de fonctions suivent les mêmes règles que les variables sauf qu'elles ne peuvent pas être exportées.

```
nom_fonction ()
{
commandes
return
}
```

Les fonctions peuvent être soit tapées dans votre script courant, soit dans un autre fichier pouvant être inclus dans l'environnement. Pour cela :
. nomfic
Le point suivi d'un nom de fichier charge son contenu (fonctions et variables dans l'environnement courant.
La commande return permet d'affecter une valeur de retour à une fonction. Il ne faut surtout par utiliser la commande exit pour sortir d'une fonction, sinon on quitte le script.

```
$ cat fonction
ll ()
{
ls -l $@
}
li ()
{
ls -i $@
}
```

```
$ .fonction
$ li
252 case1.sh 326 for4.sh 187 param.sh 897 resultat.txt
568 dump.log 409 lien_fic1 272 param2.sh 991 toto.txt
286 fonction 634 lien_fic2 260 param3.sh 716 voir_a.sh
235 for1.sh 1020 liste 42 param4.sh 1008 while1.sh
909 for2.sh 667 liste1 304 read.sh
789 for3.sh 1006 liste2 481 rep1
```

18 EXPR

La commande expr permet d'effectuer des calculs sur des valeurs numériques, des comparaisons, et de la recherche dans des chaînes de texte.

Opérateur	Rôle
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo
!=	Différent. Affiche 1 si différent, 0 sinon.
=	Egal. Affiche 1 si égal, 0 sinon.
<	inférieur. Affiche 1 si inférieur, 0 sinon.
>	supérieur. Affiche 1 si supérieur, 0 sinon.
<=	inférieur ou égal. Affiche 1 si inférieur ou égal, 0 sinon.
>=	supérieur ou égal. Affiche 1 si supérieur ou égal, 0 sinon.
:	Recherche dans une chaîne. Ex expr Jules : J* retourne 1 car Jules commence par J. Syntaxe particulière : expr "Jules" : ".*" retourne la longueur de la chaîne.

```
$ expr 7 + 3
10

$ expr 7 \* 3
21

$ a=`expr 13 ? 10`
$ echo $a
3
```

```
$ cat expr1.sh
#!/usr/bin/sh
cumul=0
compteur=0
nb_boucles=10
while [ "$compteur" -le "$nb_boucles" ]
do
cumul=`expr $cumul + $compteur`
echo "cumul=$cumul, boucle=$compteur"
compteur=`expr $compteur + 1`
done
```

```
$ ./expr1.sh
cumul=0, boucle=0
cumul=1, boucle=1
cumul=3, boucle=2
cumul=6, boucle=3
cumul=10, boucle=4
cumul=15, boucle=5
cumul=21, boucle=6
cumul=28, boucle=7
cumul=36, boucle=8
cumul=45, boucle=9
cumul=55, boucle=10
```

```
$ expr "Jules Cesar" : ".*"
11
```

19 UNE VARIABLE DANS UNE AUTRE VARIABLE

Voici un exemple :

```
$ a=Jules
$ b=a
$ echo $b
a
```

Comment afficher le contenu de a et pas simplement a ?
En utilisant la commande eval. Cette commande située en début de ligne essaie d'interpréter, si possible, la valeur d'une variable précédée par deux « $ », comme étant une autre variable.

```
$ eval echo \$$b Jules
```

20 TRAITEMENT DES SIGNAUX

La commande trap permet de modifier le comportement du script à la réception d'un signal.

commande	Réaction
trap " signaux	Ignore les signaux. trap " 2 3 ignore les signaux 2 et 3
trap 'commandes' signaux	Pour chaque signal reçu exécution des commandes indiquées
trap signaux	Restaure les actions par défaut pour ces signaux

21 COMMANDE « : »

La commande « : » est généralement totalement inconnue des utilisateurs unix.
Elle retourne toujours la valeur 0 (réussite).
Elle peut donc être utilisée pour remplacer la commande true dans une boucle par exemple :

```
while :
do
...
done
```

Cette commande placée en début de ligne, suivie d'une autre commande, traite la commande et ses arguments mais ne fait rien, et retourne toujours 0. Elle peut être utile pour tester les variables.

```
$ : ls
: ${X:?"Erreur"}
X : Erreur
```

22 DELAI D'ATTENTE

La commande sleep permet d'attendre le nombre de secondes indiqués. Le script est interrompu durant ce temps. Le nombre de secondes et un entier compris entre 0 et 4 milliards (136 ans).

```
$ sleep 10
```

23 PERSONNALISATION DE VOTRE ENVIRONNEMENT

Vous avez la possibilité de personnaliser votre environnement, ceci grâce à des scripts de connexion.

Lorsque le login et le mot de passe a été saisi, plusieurs scripts sont lancés dans l'ordre suivant :

1) **`/etc/profile`** commun à tous les users, y compris root. (On y trouve notamment la définition de umask).

2) **`/etc/profile.d/`** qui contient plusieurs scripts qui seront tous exécutés par /etc/profile (ex : alias.sh , numlock.sh) .

3) **`$HOME/.bash_profile`** (la variable $HOME correspond au « home directory » de l'utilisateur connecté). Il s'agit d'un fichier de démarrage personnel et personnalisable.

4) **`$HOME/.bashrc`** qui est lancé par $HOME/ .bash_profile ou l'on y place toutes les fonctions et alias perso.

5) **`/etc/bashrc`** qui est lancé par $HOME/.bashrc et qui contient tous les alias globaux (en autre la définition du prompt)

A la fermeture d'un shell :

`$HOME/.bash_logout` est lu . (ex : supprimer des fichiers temporaires)

TP :

Modifier votre environnement pour indiquer la date et l'heure à chaque démarrage d'un shell.

Info : La commande pour donner la date et l'heure est **`date`**
$(…) pour obtenir le résultat de l'exécution de la commande incluse

Remarque : Pour relire votre .bash_profile, utilisez la commande source
`source .bash_profile`
ou bien la commande synonyme : **`. .bash_profile`**

Pour en avoir la liste des variables d'environnement, utilisez la commande **`env`**