Cours Linux

1) Historique

1969 : Ken Thompson et Dennis Ritchie écrivent une première version du noyau d'un système d'exploitation pour les laboratoires BELL. Cet OS est conçu de façon à appréhender l'ensemble de la machine de la façon la plus homogène qui soit, et éviter tout particularisme. Les concepteurs y appliquent l'ensemble de leurs convictions sur ce que doit être un OS, et le nomme UNIX, par humour (ils travaillent à l'époque sur un Système nommé MULTICS).

Le même Dennis Ritchie invente en 1973 un langage de programmation nommé le langage C (après une version A, puis une version B). Peu de temps après, Thompson et Ritchie ré-écrivent le noyau d'UNIX en C "rompant avec une tradition qui voulait que le noyau d'un système d'exploitation soit écrit en langage assembleur"[1]. L'avantage est celui de la portabilité de l'OS, qui peut être adapté et recompilé sur une autre plate-forme, dès qu'un compilateur C est disponible.... Le langage C est plus abordable que l'assembleur, peu lié au matériel (c'est le compilateur qui l'est), et couvre les besoins de bas et de haut niveau.

C'est l'entreprise ATT qui fournit les premières versions commerciales de cet OS. Diverses entreprises s'intéressent à ce marché, et plusieurs versions apparaissent (Sun et son SunOS, qui deviendra Solaris, IBM et son AIX, HP et son HPUX...)

L'université californienne de Berkeley travaille également sur Unix, et lui apporte des atouts en réseau (les commandes r, par exemple, rlogin, rsh, rcp...

De son coté, ATT sort une version majeur, la version 5.

Les Unix d'aujourd'hui sont les héritiers des versions BSD (berkeley) ou ATT (System V), soit un peu des deux. Sun est de nos jours un acteur majeur d'Unix (voir notamment NFS, NIS...)

Évolution AT&T et BSD

System V (SCO XENIX AIX Solaris) BSD (Réseau)

2) Descriptif Unix générique

Quels sont les points communs des systèmes Unix ?

Tout d'abord, il s'agit de systèmes d'exploitation Multitâche Multi-utilisateurs. Conçu à l'origine des mainframes, il permet donc à de multiples utilisateurs de travailler sur le même ordinateur. Sur les micro ordinateurs, historiquement, les systèmes étaient plutôt mono-tâche et mono-utilisateur. C'est Windows 3.x qui a introduit la notion de multi-tâche (et encore, assez mal géré), et il a fallut attendre Windows 95 pour un vrai multi-tâche préemptif. Là encore, l'observateur attentif a pu remarquer qu'une simple tache "plantée" suffit à cracher entièrement l'OS. Dans le monde Microsoft, seul Win NT est plus sécurisé (utilisation des Ring 0 et 3 du processeur, voir cependant la problématique de la carte VGA). Malgré toutes ces avancés, le mode multi-utilisateur est toujours absent des solutions Microsoft (Il faut reconnaître que cela n'apporte qu'un petit plus). L'extrême

1 Extrait de "L'environnement de programmation UNIX", un must-read de Kernighan et Pike...

fiabilité du noyau Linux (on parle bien ici du noyau, c'est à dire du coeur du système, qui doit résister à tous les crashs des applications) a certainement poussé la firme de Redmond dans le sens d'une meilleure écriture de ces OS, afin de pouvoir tenir la comparaison.

Le noyau est écrit en C, ce qui permet un portage, ou une adaptabilité. Il suffit parfois de recompiler son OS (si vous disposez des sources bien sûr !!!) avec les bonnes options pour l'adapter parfaitement à votre machine (les OS microsoft sont compilé quant à eux de manière générique, avec les options correspondant au moins puissant des matériels supportés : voir la notion de logiciel 16 bits par exemple...)

Un autre point important est l'optique générique choisie. Tout ce que peut gérer l'OS doit l'être de façon identique. On a coutume de dire que sous UNIX tout est fichier... La gestion des périphériques (disquette, clavier souris écran, disque dur, réseau, etc.) est identique à la gestion des fichier. Cette notion très troublante quand on vient d'un autre monde permet dès son appropriation la réalisation d'action pour le moins étonnante (duplicata de disque dur, "formatage" de fichier, transferts insolites comme par exemple la copie de l'entrée son d'un PC sur la sortie haut parleur d'un autre PC en réseau, etc...)

Shell programmable : UNIX est un système d'exploitation. Il permet de faire fonctionner un ordinateur, et lui permet d'assurer un certain nombre de services. Parmi ces services, le shell, ou interpréteur de commandes, va permettre à un utilisateur de lancer des commandes de façon interactif. C'est le programme chargé d'assurer le dialogue avec l'utilisateur. Sous UNIX, vous disposez de plusieurs shells, vous choisirez celui qui est le plus pratique pour vous. La tendance actuelle est à l'utilisation du BASH (Bourne Again Shell). Les autres shells connus sont le SH (le Bourne Shell), le Korn shell, le C shell... Tous ces shells sont capables d'assurer l'exécution d'une suite de commandes, ce que l'on appelle un script. Tous ces shells sont de véritables langages de programmation...

Le réseau : Les liens entre UNIX et le réseau sont très importants, de par les avancées de BSD. TCP/IP a été porté très tôt sous UNIX, et une majorité des ordinateurs serveurs de l'Internet sont sous UNIX.

Quant à l'interface utilisateur, il existe aussi une interface graphique sous UNIX : Il s'agit de X Window, qui a la particularité d'être client serveur et orienté réseau. Ce système qui peut sembler assez lourd permet cependant d'étonnantes opérations, comme par exemple lancer de sa machine un programme qui s'exécutera sur une autre machine, alors que l'affichage graphique se fera sur une troisième !!! Un autre avantage du client serveur est la possibilité de choisir parmi de multiple Window Manager (programme chargé du comportement et de style d'affichage). Un utilisateur X-Window d'une distribution standard de Linux se servira de l'interface KDE, GNOME, TWM, Enlightment ou autre WindowMaker, SAWFISH ou IceWM selon sa convenance et le plaisir qu'il y trouvera.

Et Linux dans tout ça ? Linux est un produit libre, développé par des étudiants et maintenant des entreprises (dont IBM, Corel, etc) qui est né en 1991. Très en vogue actuellement (en réaction au monopole de Microsoft), il est très répandu sur Internet, dans les écoles et Universités, et entre petit à petit dans le monde de l'entreprise (IKEA, la Poste, les armées et autres lieux "sensibles à la sécurité"...)

Ses avantages : toutes les sources sont fournies, il est libre,très stable, très réactif (la détection d'un bug est très rapidement corrigée), il est gratuit, compatible Posix, c'est un challenger et une alternative très sérieuse en ce qui concerne les serveurs, il est très lié au hardware (on peut recompiler les programmes pour les adapter parfaitement à sa machine), il est multi-plate-forme

(Alpha, PowerPC, Macintosh, IBM BULL etc), et enfin il offre des performances très impressionnantes...

Ses inconvénients : Il est austère (il faut aimer la ligne de commande), touffu (il faut plusieurs années pour vraiment avoir une vue d'ensemble d'un système UNIX en général), pas ou peu supports techniques (en fait, il y a une documentation énorme fournie avec Linux, mais il faut se donner la peine de la lire, et c'est parfois très long), il n'y a pas tous les drivers (en effet, très peu de fabricants développent des pilotes pour Linux, il faut donc attendre que des bénévoles hyper compétents les réalisent et parfois, sans aucune doc du fabricant), il y a une certaine méfiance des entreprises (encore que cela change, voir par exemple le coup d'un virus comme I LOVE YOU, qui, à lui seul, montre le danger de la mono-culture microsoft), et enfin qu'en est il de la pérennité du produit (un produit gratuit, fait par des bénévoles, peut il survivre à long terme ? En fait, la question serait plutôt l'inverse : un produit propriétaire, fabriqué par une entreprise qui pourrait disparaître, peut il survivre à long terme ? (voir CP/M, Dbase, Quicken, etc))

Une petit mesure de la taille de la documentation sous Linux : 101 Millions d'octets de texte (presque) pur !!!

2.a - Prise de contact

Allons y : démarrer votre machine. Après un éventuel LILO (c'est un petit programme au démarrage de la machine, qui permet de démarrer soit Linux, soit Windows. En fait il permet de démarrer presque n'importe quoi, mais disons que le choix Windows/Linux est ce que l'on trouve habituellement...).

Après le défilement de nombreuses informations (Linux est bavard, ce qui est pratique pour détecter une panne !!!), vous arrivez à un écran de login, c'est à dire de connexion.

Entrez votre nom de login, et votre password : vous êtes connecté au système.

Bon, maintenant, qu'est ce qu'on fait ? Commençons par arrêter la session : tapez **logout**. Vous êtes déconnecté, l'écran repasse en mode de login.

Avant d'aller plus loin, revenons à des notions théoriques. Sous Linux (et sous UNIX en général), vous êtes dans une ARBORESCENCE. C'est à dire qu'il y a une organisation hiérarchique de répertoires, qui peuvent contenir des fichiers et/ou des répertoires. Cette notion est simple, puisque le monde Microsoft l'a repris. Par contre, il n'y a pas de C: ou de A:. Dans le monde Windows, il y a

une arborescence pour chaque support. Sous Linux, Il n'y a qu'une et une seule arborescence !!! Et en plus c'est toujours la même, que vous soyez sur un système qui n'a même pas de disque dur (juste une clé USB par exemple), ou sur un super ordinateur avec 50 TOctets en ligne !!!

Cette arborescence a une racine, un début, un sommet... Celui ci est noté /

On l'appelle aussi root. On trouve donc sous root toujours les mêmes répertoires : Chacun d'eux a un rôle bien défini. Ainsi, on trouvera :

/etc	Tous les **fichiers de configuration** sont rangés ici...
/bin	Les **programmes importants**.
/sbin	Les **programmes nécessaires au démarrage de la machine** ! Le *S* ici sous entend que les programmes sont statiques (compilés de façon à être autonomes, et ne pas dépendre de librairies (les .so, équivalents des DLL sous Windows). Il semble pourtant que certains programmes présents soient dynamiques (c'est à dire dépendants de bibliothèques) (*Nota : pour savoir de quelles so dépent un programme, utilisez la commande ldd*)
/boot	Le noyau est parfois rangé ici(noyau = coeur du système d'exploitation)
/usr	Ici, des programmes et des informations plus propre à cette machine...
/usr/bin /	Programmes moins important que /bin...
/usr/local	Logiciels propres à cette machine (jeux, services, etc)
/home	**Répertoires des utilisateur**s. Attention, ce répertoire est protégé, ce sont les sous répertoires à l'interieur qui sont spécifiques à chaque utilisatuer
/home/...	Chaque utilisateur a un répertoire pour stocker ses données. Il a les droits sur ce répertoire.
/tmp	Comme son nom l'indique... Accessible à tous
/var	Administration... Souvent les traces et suivi d'événements...
/var/log	L'historique des connexions, incidents, utilisation, requêtes, accès....
/var/spool	Le tampon d'impression ou de courrier par exemple
/mnt	Un répertoire de **montage** (pour l'accès aux périphériques amovibles... Maintenant, c'est /media qui a ce rôle (à cause du programme udev, qui créé à la volée le répertoire correspondant à chaque ressource détectée)

Comme vous le voyez, tout est bien défini. De plus, selon votre identité, vous aurez plus ou moins de droits d'accès sur chacun de ces répertoires : cela permet d'assurer une certaine stabilité du système, car un simple utilisateur ne peut pas effacer de programmes dans /bin par exemple, ni propager de virus.... Nous verrons cette gestion des droits, propre aux systèmes moderne, plus loin.

Dernier petit point : nous verrons plus tard comment utiliser un cdrom, ou une disquette. Disons simplement ici qu'il faut "monter" un périphérique dans un répertoire pour accéder à son contenu. C'est à dire qu'il faudra dire : "La disquette, ou la clé USB, qui est de type Microsoft Windows, lis la, et fais moi apparaître son contenu dans tel répertoire". Après cette opération, et jusqu'à nouvel ordre, le répertoire représentera la disquette. On peut "monter" des périphériques dans n'importe quel répertoire créé pour l'occasion. En fin de travail, il ne faudra pas oublier de "démonter" le support, pour libérer le répertoire, et permettre à l'OS de stabiliser le contenu du support.

Remarquez enfin que l'on peut indiquer le type d'organisation du support magnétique (par exemple fat, vfat, minix, ntfs...). Si l'OS connaît ce type, alors il est capable d'en lire le contenu, et vous permet de le gérer !!!

2.b - Initiation aux droits (exécuter, lire, écrire, traverser, etc.)

Linux étant multi utilisateur, il offre la possibilité de sécuriser l'accès aux données. Ainsi, pour chaque fichier ou répertoire (et même pour un périphérique, car n'oublions pas qu'UNIX considère tout comme des fichiers), il va être possible de gérer les droits de lecture écriture exécution. Bienvenue dans le monde merveilleux des ACCESS DENIED...

Chaque utilisateur dispose d'un home directory, un répertoire qui lui est réservé... Si je me loggue sous le nom de sylvain, je disposerai d'un répertoire /home/sylvain. Je pourrais y enregistrer des données, et je pourrais gérer la sécurité d'accès à ces données... Un objet (fichier, répertoire, etc), appartient toujours à quelqu'un.... Et ce quelqu'un peut en gérer les droits. Il existe cependant un utilisateur un peu particulier (**root**) qui peut tout faire. C'est l'administrateur du système.

L'administrateur du système (**root**) est chargé de la gestion de la machine, et des utilisateurs. C'est lui qui crée votre compte, active les services, organise la sécurité et les sauvegardes. Le système lui offre de regrouper les utilisateurs en groupes, car il sera plus facile ensuite de donner des droits à un groupe plutôt qu'a chaque utilisateur.

Attendez vous donc à être un utilisateur, et à appartenir à un (voire plusieurs) groupe(s). Vous disposerez donc d'un nom de login, d'un UID (User Identifier, un numéro unique) et d'un GID (Group Identifier). De même, chaque objet que vous verrez aura un propriétaire (un USER). Il existera des droits pour ce propriétaire, des droits pour un groupe précis (pas toujours le groupe du propriétaire d'ailleurs) et pour les autres...

2.c - Démarrage Arrêt de la machine

montage-cache-services-**niveau init**

Au démarrage, il est possible de choisir le niveau de démarrage (le niveau d'init). Cela peut faire penser au mode de démarrage de Windows. Selon les cas, vous pouvez démarrer la machine en mode autonome (vous êtes seul sur la machine), en mode multi-utilisateur, ou encore en mode multiutilisateur avec connexion graphique....

L'ensemble des services est lancé une fois pour toutes au démarrage, ce qui explique la multitude de messages Starting.... qui apparaissent au début. N'oubliez pas que Linux est un vrai serveur !!! La machine sur laquelle je tape ce texte (un simple P200 avec 48 Mo de RAM, du moins lors de sa première version :-)) dispose de 5 serveurs WEB, 1 serveur FTP, accepte des connexions réseau (telnet, et autres commandes r...), un SGBDR (PostGres), etc...

De plus, au démarrage, l'accès aux supports magnétiques est lancé, puis mis en cache (c'est à dire que le transfert de données se fera au moment le plus opportun, afin d'offrir un temps de réponse très court).

En ce qui concerne l'arrêt de la machine, pour des raisons de stabilisations (flush ou vidage du cache) et parce que Linux est un serveur, il faut bien demander l'arrêt de la machine. On peut le faire très rapidement en pressant CTRL ALT SUPPR. L'arrêt de tous les services est alors lancé, la synchronisation de tous les supports magnétiques est effectuée, et le système s'arrête....

2.d - Écrans Virtuels

Linux est un vrai système multi-utilisateurs : Il est donc possible de s'y connecter plusieurs fois. Votre PC sous Linux offre de base 6 écrans texte et un écran graphique. Comment faire pour y accéder alors que vous n'avez qu'un seul clavier écran ?

Réponse : en utilisant les touches ALT et les touches de Fonctions : Alt-F1 donne le premier écran, Alt-F2 le second, Alt-F3 le troisième...Alt-F6 le sixième et Alt-F7 pointe sur l'écran graphique (si le X Window est en route, bien sur)

Attention : Pour ressortir de X et repartir vers les écrans virtuels textes, utilisez la combinaison Alt+CTRL+Fx.

Bien sur, vous pouvez être un utilisateur différent sur chacun des écrans, ou le même partout, ou un mélange des deux. L'ensemble de ces écrans est indépendant, et il est souvent pratique de lancer un long travail sur l'un d'eux, et de travailler en interactif sur un autre.

On utilise aussi cette technique pour surveiller une tache, voir même débloquer un écran (si une tache se bloque, passez sur un autre écran, et "tuer" (arrêter) la tache bloquée...

Il est possible de lancer plusieurs sessions graphiques. Pour ce faire, allez dans un écran texte, et lancer un nouveau startx (il faut écrire exactement startx –:1). Le double tiret veut dire qu'on ne veut pas lancer un nouveau serveur X, mais juste une nouvelle interface. Et le :1 indique le deuxième écran graphique (et oui, le premier, c'est:0). Attention, si l'interface graphique est sur la console 7, alors, cette nouvelle sera sur l'écran 8... Et ainsi de suite (:0 correspond à l'écran 7,:1 à l'écran 8,:2 à l'écran 9...)

2.e - Démarrage et arrêt de session

commande **login**

Login est un programme qui est lance sur chaque terminal offert par votre pc. Celui vous demande votre nom (attention a la casse, c'est a dire les majuscules minuscules) puis votre mot de passe. Ce couple est ensuite compare aux informations contenues dans les fichiers charges de la sécurité des utilisateurs, c'est a dire /etc/passwd (nom de utilisateur, numéros, répertoire de base, et programme lance au démarrage) et /etc/shadow (mot de passe crypte). Si la paire est valide, vous vous retrouvez dans le programme détermine dans /etc/passwd (il s'agit souvent d'un shell, c'est a dire un programme qui attend que vous saisissiez des commandes), et votre répertoire actif est le home. Il existe d'autres systèmes d'authentification centralisés, dans lequel votre machine se tourne vers un système central pour que l'utilisateur obtienne ses accréditations.

commande **who**

Cette commande permet de connaître la liste des utilisateurs actuellement connectes a votre machine. Si vous êtes connecté sur plusieurs terminaux virtuels, vous le verrez...

commande **logout**

Fin de connexion, cette commande ferme tous les travaux en cours, et remet le terminal dans son état de départ (attente de login).

Commande **exit**

Sortie du programme en cours, ce qui peut correspondre à une déconnexion (un logout)... mais pas toujours...

2.f - Commandes de répertoires

pwd : Print Working Directory, cette commande vous indique le nom du répertoire actif, c'est a dire celui dans lequel vous êtes en train de travailler. On ne peut être qu'a un seul endroit a la fois...

ls : affiche le contenu du répertoire courant (ou du répertoire demande). Équivalent de DIR sous Dos.

cd : changement de répertoire. L'appel de .. permet de remonter d'un niveau. cd tout seul vous repositionne dans votre home... ATTENTION aux notions de RELATIF et d'ABSOLU (pour savoir où vous êtes, et comment se déplacer vers votre objectif (*voir plus loin la complétion*)

mkdir : permet la création d'un ou plusieurs répertoires

rmdir : permet la suppression d'un ou plusieurs répertoires (vides bien sur)

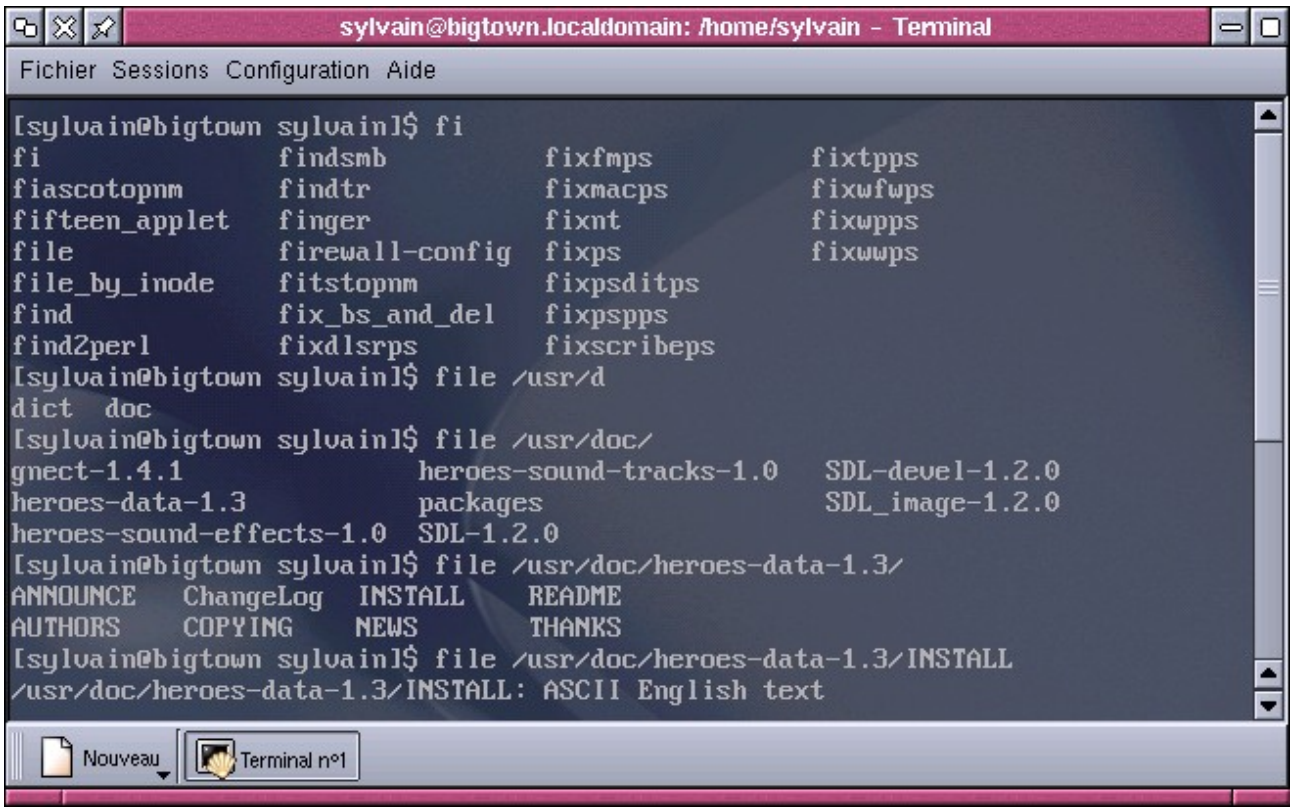
ls -al, ls -d : Ce sont des options de ls. Consultez le man pour en savoir plus (**man** ls)

le ~ (tilde) et les chemins (absolu et relatifs)...

Il faut bien comprendre l'arborescence des répertoires, et les fichiers. Les fichiers sont un ensemble cohérent de données, que l'utilisateur a décidé d'enregistrer. Afin de retrouver son travail, il doit donner un nom à ces données. Ce nom peut ou non avoir un rapport avec les données. Il n'est pas stocké dans les données, mais en tant que nom symbolique d'accès à ces données. Ces données (ou ce fichier, comme on veut) sont rangées quelque part : dans un répertoire. Le vrai nom des données est constitué de l'assemblage du nom du répertoire et du fichier (par exemple, le fichier truc dans le répertoire /usr/local/bin : Le nom complet du fichier est /usr/local/bin/truc). Ce nom est complètement qualifié, ou encore absolu. Prenez l'habitude de bien référencer vos données, de bien comprendre ou vous vous situez dans l'arborescence des répertoires, et d'être capable de vous déplacer dans cette arborescence.

la complétion et l'édition de la ligne de commande

Toute la puissance de la ligne de commande (l'interface texte) est accessible grâce à la complétion. En utilisant la touche TAB, le bash (l'interpréteur habituel sous Linux) ira chercher pour vous tous les noms qui correspondent au début de votre saisie.. Si il y a une solution unique, le bash complète pour vous.... Sinon, la machine beep !!! Une seconde pression sur TAB donne soit un beep (il n'y a pas de mot commençant par votre saisie !!) soit vous obtenez la liste de toutes les solutions possibles !!



ici, pour obtenir la frappe de la commande **file /usr/doc/heroes-data-1.3/INSTALL**, je n'ai tapé que

fi TAB TAB ***l*** TAB ***/u*** TAB ***d*** TAB TAB ***o*** TAB TAB ***h*** TAB ***-d*** TAB TAB ***I*** TAB

Il n'y a plus de faute de frappe (puisque c'est la machine qui saisie !), et on fait directement la lecture du contenu des répertoires en même temps que l'on saisie la commande ! De plus, on a une ligne de 37 caractères en tapant seulement 11 touches...

2.g - Commandes de fichiers

ls : liste le contenu d'un répertoire : Attention, certains fichiers sont cachés (ceux dont le nom commence par un .) L'option -a permet de voir tous les fichiers. L'option -l permet d'avoir plus d'informations sur chacun de ces fichiers

file : commande mineure, qui permet de savoir le type d'un fichier (Il est préférable de se baser sur le contenu réel du fichier plutôt que sur l'extension de son nom!!!!)

cp : copie les fichiers (comme copy en dos). *cp origine arrivée*. *Origine* peut être un ou plusieurs fichiers *arrivé* peut être un fichier (si une seule origine), ou un répertoire (si une ou plusieurs origines)

mv : comme cp, si ce n'est qu'il efface l'origine

rm : suppression du ou des fichiers donnés en argument

cat more et *less*: affiche le contenu du ou des fichiers donnés en argument. Cat est la commande la plus simpliste, less la plus élaborée...

touch : commande mineure, qui permet de changer la date de dernier accès à un fichier, sans risque de modifier son contenu (penser aux sauvegardes). Utilisé souvent car il permet de créer le fichier si celui ci n'existait pas (il sera vide, 0 caractères)

3) les aides

La commande **man :** man commande permet d'avoir accès à l'aide sur cette commande : toujours présenté selon une norme (NOM, SYNOPSYS, DESCRIPTION, OPTIONS, BUGS), elle est assez complète (souvent plusieurs pages écrans). On peut se déplacer grâce aux touches de directions, ou Espace et Entrée pour avancer, rechercher un mot avec la touche / suivi du mot, et on sort avec la touche Q.

la commande **apropos** (ou whatis, si il y a eu un makewhatis) : apropos commande affiche une ligne d'information sur l'objectif de la commande en question.

La commande **info** : Commande plus récente, elle permet, à la mode EMACS (éditeur de RMS, très très complet) d'accéder à l'aide, hypertextuelle et parfois encore plus complète que man. Le curseur vous permet de vous déplacer sur des mots clefs (précédés d'une étoile), de valider pour afficher la page correspondante, n et p pour les pages suivantes et précédentes (Next et Previous), espace et backspace pour avancer et reculer dans la page, et q pour quitter l'aide.

Le **--help** : en général, les programmeurs ont prévu une aide succincte dans la commande, accessible par l'option --help (ou -?)

les **HOWTO** : Souvent rangés dans /usr/share/doc, les HOWTO (et mini HOWTO) sont des description assez complètes d'ordre générique (comment fonctionne le SCSI, qu'est-ce qu'un FileSystem, comment graver des Cds, etc..)

4) L'entrée et la sortie standard

4.a - Redirections

redirections : toute commande bien née utilise l'entrée standard (**stdin**, ou 0), la sortie standard (**stdout**, ou 1), et la sortie d'erreur (**stderr**, ou 2). Il s'agit du ficher en entrée (en programmation : Shell commande read, ou cin (en C++), ou scanf (en C), ou System.in(en Java)), en sortie (write, cout ou printf ou System.out), et du fichier qui va récupérer les messages d'erreur (perror).

En général, c'est le clavier qui est considéré comme **stdin**, et l'écran est assigné à **stdout** et **stderr**. C'est pour cette raison que les messages d'erreur apparaissent en même temps que les bons résultats. Dès que l'on a compris cette notion de **stdin**, **stdout**, et **stderr**, on peut s'amuser à ne visualiser que les sorties qui nous intéresse, ou encore utiliser une autre entrée que le clavier... Par exemple, demandons à la machine de nous afficher le contenu des tous les répertoires et sous répertoires de /home... Évidement, nous allons avoir énormément de réponses. Mais si nous supprimons le **stdout**, et ne gardons que le **stderr**... Nous verrons tous, et seulement, les messages d'erreur... En général, il s'agira des fichiers et répertoires auquel l'accès nous est interdit... Vous devinez l'intérêt d'une telle commande ?

Comment rediriger les entrées et sorties ? Grâce aux signes suivants

> : redirige stdout vers le fichier indiqué après >

>> : comme >, mais si le fichier existait, on ajoute à la suite (mode append)

2> : redirection de stderr vers le fichier indiqué

1> : comme >

< : redirection d'entrée, le fichier indiqué après < remplace stdin (donc, souvent, se substitue au clavier)

exemples :

ls -al >/tmp/fichier

who >>/tmp/fichier

fdisk /dev/hda </tmp/reponses_prevues

Sachez que vous pouvez jongler avec la puissance d'Unix, et donc utiliser la base de ce système : TOUT EST FICHIER. Il existe des fichiers pratiques, tel que /dev/null (le trou noir, qui absorbe tout), /dev/zero (qui fournit une infinité de caractères 0), etc... L'exemple donné plus haut, qui permet la rechercher de tous les endroits interdits, s'écrit de la façon suivante :

ls -Rl /home >/dev/null

4.b - pipes et commandes filtres associées

filtres (briques simples) **grep, cut, sort, head, tail, uniq, wc, tr, tee**...

Plus amusant encore, il est possible d'enchaîner les commandes.. afin qu'une commande travaille sur le résultat de la commande précédente... Il s'agit en fait de brancher le **stdout** d'une commande sur le **stdin** de la commande suivante... et ainsi de suite... Habituellement, on connaît cette instruction (sous MSDOS) :

DIR | MORE

La sortie de la commande DIR alimente la commande MORE. Ce système, mal implémenté sous DOS, vient d'Unix, ou il est particulièrement efficace (il sous-entend automatiquement le multitache !). le « pipe » est obtenu en combinant la touche ALTGR et la touche 8. Ce système permet la construction de commandes élaborées, basées sur une succession de commandes simples, « pipées » entre elles... On évite ainsi de passer par des fichiers temporaires, ce qui ralentirait le déroulement de l'opération. Les « pipes » peuvent bien sur s'enchaîner les uns les autres...

Les « pipes » utilisent souvent des filtres, c'est à dire des commandes qui modifient, découpent, réorganisent ou simplifient le flux entrant (c'est à dire ce qui arrive sur **stdin**). Les filtres sont les commandes suivantes : **grep, cut, tr, sort, uniq, wc, tail, head, sed..**. On utilise aussi **wc** (compte les mots, lignes et autres caractères), et **more** ou **less** (pager, qui permettent le découpage lisible)

4.c - les filtres

more, less : ce filtre est un pager : il découpe le flux entrant en pages... pratique pour éviter de se faire engloutir par de trop nombreuses informations : less est supérieur, car il permet de revenir vers

le début.

grep : *grep 'pattern'* recherche dans le flux les lignes qui contiennent le 'pattern' : il les laisse passer. Si la ligne ne contient pas le parttern, elle disparaît du flux. Il existe des options intéressantes, comme le -v, qui inverse le fonctionnement (les lignes qui correspondent au pattern disparaissent, ce sont les autres qui passent le filtre). Commande très importante et utile.

tail, head : ne laisse passer que le début (ou la fin) du flux. En général, 10 lignes.

tr : permet de substituer des caractères (en général, 1 par 1) : par exemple tr 'b' 'a' transformera tous les b du flux qui passe en a. Cela ne fonctionne pas sur des mots, par contre. Options utiles -s (pour squeezer (tasser) un caractère (l'espace par exemple, comme dans *tr -s ' '*), ou encore -d pour carrément supprimer un caractère du flux (*tr -d 0* supprimera tous les zéros du flux).

sort : tri les lignes du flux. L'ensemble de la ligne est considéré par la commande. Attention aux problèmes des nombres (voir l'option -n)

uniq : élimine les lignes en double dans le flux : Attention, les lignes doivent se suivre (utilisé en conjonction avec sort)

wc : Word Count, compte les caractères, les mots et les lignes du flux. Pratique pour faire des stastistiques.

sed : mini-éditeur qui travaille sur les lignes à la volée du flux : Utilise les commandes de l'éditeur ed. Par exemple, *sed s/auteur/'Sylvain Cherrier – copyright 2008'/* transformera chaque occurrence du mot auteur dans le flux par Sylvain Cherrier – copyright 2008.

5) VI

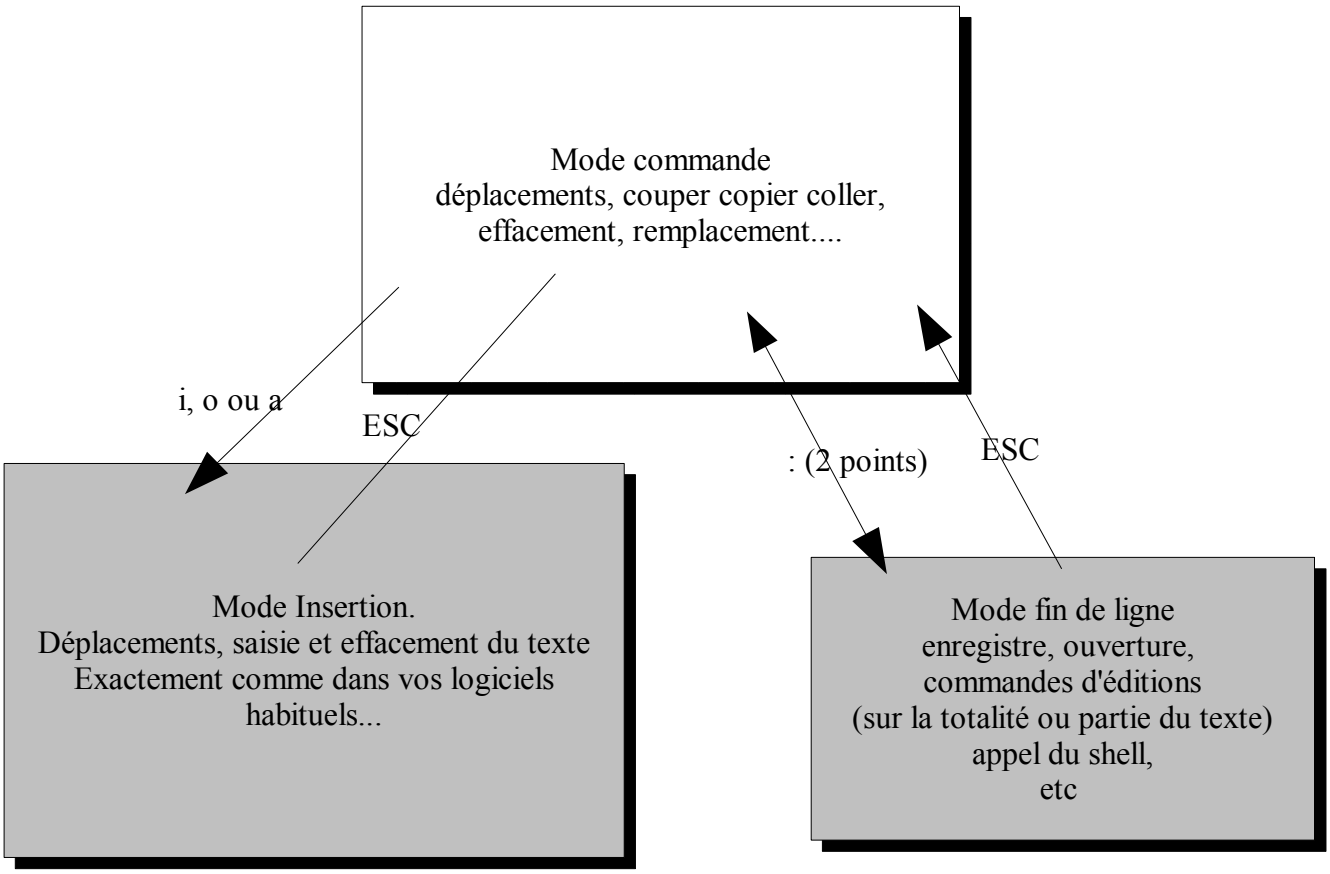
Vi est un éditeur de texte, c'est à dire un programme qui permet l'écriture et la gestion de textes. Il est très versatile, et très répandu. Sa mauvaise réputation n'a d'égale que sa puissance. En effet, il ne s'agit pas d'un éditeur WISIWYG. Il fonctionne en mode texte, bien que des versions graphiques existent. Il dispose d'une multitude de commande, et d'extension. Il peut servir à éditer n'importe quel fichier d'administration (sous Linux, les fichiers de configuration sont en général de simples fichiers textes, ce qui permet la gestion de la machine et des services sans passer par un frontal quelconque (bien qu'il en existe plein !!!)), mais aussi des fichiers en HTML (avec vue traduite ou brute) et des sources de programmes (avec analyse syntaxique, et même lancement de compilation...).

Ce qui lui est reproché est son manque d'intuitivité (il faut se souvenir des commandes par coeur). Heureusement, il s'agit souvent d'initiales de commandes (dw pour Delete Word), présenté sous forme d'alphabet assemblable (d pour delete, c pour change, w pour mot, $ pour jusqu'à la fin, d'où c$ pour changer toute la fin d'une phrase). De plus, l'apprentissage peut être facilité si on l'utilise fréquemment cet éditeur, c'est pourquoi bash permet de gérer la ligne de commande à la façon vi pur et dur (soit par la commande set o vi, soit en complétant le .inputrc avec la ligne set editing-mode vi (dans le home directory de l'utilisateur).

La difficulté d'apprentissage sera un jour récompensée par la vitesse dans vos travaux.

5.a - Le principe de fonctionnement.

VI dispose de trois modes : commandes, insertion, et fin de ligne. Au démarrage, il est en mode commandes. Pour y revenir, appuyer toujours sur la touche ESC.



La trame générale du fonctionnement de VI...

5.b - Le mode commande :

Il permet de visualiser l'ensemble du texte, et de s'y déplacer. Les touches de déplacement sont normalement les suivantes :

h va à gauche

l va à droite

k remonte

j descend

Souvenez vous surtout de k et j, car elles seront utiles sous bash (lorsqu'il est en mode vi). En fait, votre vi a beaucoup de chance d'être en fait vim (vi improved) : les touches de curseurs seront alors gérées (flèches en haut, bas, gauche et droite ainsi que PageUp et PageDown).

5.b.1) Les objets :

pour traiter du texte, il faut savoir comment le logiciel appelle les différentes parties manipulées. Ainsi, vi appelle w un mot (un word), $ indique de la position courante jusqu'à la fin de la ligne, e indique un mot et l'espace derrière...

5.b.2) Voyons quelques commandes maintenant :

Commande	***Explication***	***Mode d'emploi***
r	Permet d'échanger par une lettre	Se positionner sur la lettre à changer, tapez r suivi de la bonne lettre
R	Bascule en mode de remplacement (pour changer un certain nombre de caractère)	Se positionner, appuyer sur R (un indicateur REPLACE apparaît en bas) et taper le nouveau texte. Sortir par ESC
d	Efface l'objet indiqué (w $ e ..) et en profite pour le mettre en buffer (équivalent au couper, donc)	Se positionner au début de l'objet à modifier, puis tapez d suivi de l'objet
dd	Variante du d, cette commande supprime la ligne entière	Se positionner n'importe où sur la ligne, et taper dd

Commande	***Explication***	***Mode d'emploi***
c	Remplace l'objet indiqué (w $ e ...)	Se positionner au début de l'objet à modifier, puis tapez c suivi du code de l'objet, et taper le nouveau texte (on passe en mode insertion -> sortie par ESC)
y	Copie l'objet indiqué (w $ e) (se dit yank). L'objet est alors dans le buffer.	Se positionner, et taper y suivi du code de l'objet.
yy	Variante du yank, yy copie la ligne entière	Se positionner n'importe où sur la ligne, et taper yy
p	Put : cette commande est l'équivalent du coller.	Se positionner à l'endroit voulu, et taper p

Toutes ces commandes peuvent être agrémentées de valeurs, ce qui permet d'étendre leur portée.

Ainsi d accepte une valeur avant le type de l'objet, de même que c et y... ce qui donne

d3w -> efface les trois mots.

c2w -> change les deux mots suivants.

De même, les commandes dd et yy acceptent une itération.

10dd supprime 10 lignes.

3yy copie 3 lignes.

5.b.3) Commandes de déplacement et de recherche.

Commande	***Explication***	***Mode d'emploi***
w	Déplacement de mot en mot	w
$	Aller directement en fin de ligne	$
^	Permet de revenir en début de ligne	^ (il faut taper deux fois sur la touche!!!)

Commande	***Explication***	***Mode d'emploi***
%	Magique pour les programmeurs, cette touche permet de trouver l'autre caractère particulier en regard de celui sur lequel se trouve le curseur (sert à trouver la parenthèse fermante lorsque l'on est sur l'ouvrante, par exemple)	Se mettre sur un caractère de type { [ou (et taper %.
G	Allez à la fin du texte	Tapez G
xxxG	Permet d'aller directement à la ligne xxx	Tapez directement xxxG : rien n'apparaît pendant la frappe, seul le saut est effectué au moment du G
Ctrl-g	Affiche des informations sur votre position actuelle dans le fichier	Tapez ctrl g, et lire la dernière ligne...

5.c - Passer en mode insertion

Pour passer dans ce mode qui est le stricte équivalent au traitement de texte que vous connaissez, vous avez plusieurs possibilités :

Commande	***Explication***	***Mode d'emploi***
i	Passe en insertion (saisie de texte) à l'endroit du curseur	Se positionner à l'endroit ou l'on veut insérer le texte, et taper i.
I	Passe en mode insertion en début de la ligne pointée	Se positionner n'importe où sur la ligne devant laquelle on veut insérez, et taper I.
a	Append : insert après le caractère courant...	
A	Insert en bout de ligne.	
O	Insert sur une nouvelle ligne, en dessous de la ligne courante.	Se positionner sur la ligne, et taper O pour saisir juste en dessous.

On ressort de ce mode par la frappe de la touche ESC.

5.c.1) Le mode fin de ligne.

On y accède par les touches : ou / ou encore !
C'est ici que l'on peut gérer les documents (enregistrer, ouvrir, faire des substitutions, insérer d'autres textes, ou des résultats de commandes). On l'appelle ainsi pour la simple raison que la frappe apparaît en dernière ligne de l'écran.
Souvent, on peut moduler la portée des commandes en utilisant a,b ou a représente la première ligne et b la dernière ligne mise en jeu dans la commande. Par exemple, w voulant dire enregistrer (write), la commande :5,7w toto devrait enregistrer le texte de la 5ème, 6ème et 7ème ligne dans un fichier nommé toto.

Commande	***Explication***	***Mode d'emploi***
w	Enregistre le fichier (sous son nom courant). Il est possible de taper un autre nom (la complétion de bash fonctionne !)	:w toto
e	Ouvrir un fichier, dont on donne le nom en argument	:e toto
!	Exécute la commande shell donnée en argument	:! ls exécute dans un sous shell cette commande, puis revient à vi
r	Insert le nom du fichier donné en argument sous la ligne courante	:r titi
q	Quitte vi.	:q
/	Recherche la chaîne donnée en argument. / tout seul affiche la prochaine concordance, de même que n (pour next).	La frappe de /essai place le curseur sur la prochaine occurrence (si elle existe) de essai dans le texte édité.
Help	Permet d'afficher un écran d'aide (ouf!)	:help
s	Substitution d'une chaîne par une autre. l'option g en fin de commande permet de changer de façon globale, sur la portée de la commande (voir exemple).	:s/Vim/Notepad/ remplace sur la ligne courante la première occurrence de la chaîne Vim par la chaîne Notepad. :s/Vim/Notepad/g fait la même chose pour TOUTES les occurrences de Vim dans la ligne courante. :5,10s/Vim/Notepad/g change TOUTES les occurrences de Vim de la ligne 5 à 10 (même si plusieurs par ligne)

6) Le système de fichiers de Linux (extended 3 FS)

Un système de fichiers permet la gestion de données sur un périphérique, en représentant ces données sous forme de fichiers, regroupés dans des répertoires organisés de façon hiérarchique : c'est l'arborescence des répertoires. Evidément, ceci est un grand mensonge, et une vue de l'esprit. Nous savons que le disque dur par exemple est constitué de pistes et de secteurs : Un fichier est stocké sur un ou plusieurs secteurs d'une ou plusieurs pistes, un point c'est tout.

L'astuce du file system est donc de présenter cette réalité d'une autre façon, en introduisant un ensemble d'informations attachées au fichier (son nom, sa position relative à l'ensemble c'est à dire l'endroit conceptuel de rangement, sa date de création, ses attributs ...).

Chaque OS utilise un ou plusieurs système de fichiers. Certains en comprennent plusieurs.

Le file système de dos (fat, ou fat16, ou vfat, ou fat32, selon les époques) permet de donner un nom de 8 caractères plus 3 (voire plus depuis vfat, mais il s'agit en fait d'une surcouche à fat16, puisque le système utilise un table supplémentaire entre le nom long (vfat, ou virtual fat), et le nom rééllement utilisé dans la machine (fat16)). Il permet de donner 4 attributs aux fichiers (lecture seule, système, caché et archive). Il n'est pas fait mention de propriétaire, ni de groupe.

Le système de fichier de Linux s'appelle ext3fs. En fait, Linux utilise un VFS (Virtual File System), qui est un pseudo système de fichier qui vient en surcouche du système réel. Cette habile construction vous permet de n'apprendre qu'un type de commandes, qui fonctionneront sur l'ensemble des files système de votre machine : Nous savons qu'il est possible de travailler sur des partitions DOS à partir de Linux : Sachez de plus que les commandes sont toujours les mêmes... C'est le VFS qui transformera les commandes utilisées en commandes réelles appropriées au file systèm utilisé... Un cp deviendra un copy sur une partition dos.

Ext2fs, comme tous les systèmes UNIX, accepte les types de fichiers suivants...

les répertoires :	ensemble de fichiers
les fichiers ordinaires :	programmes, données, etc
les fichiers spéciaux :	les périphériques, en mode caractères (clavier, souris, etc) ou blocs (DD,Dkt)

Le file system accepte des noms longs (256 car), composés des caractères suivant : ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.

Il est possible d'abuser du point, dans le but de commenter par exemple le type du fichier (une

sauvegarde d'un fichier passwd zippée pourrait être nommée passwd.old.z)

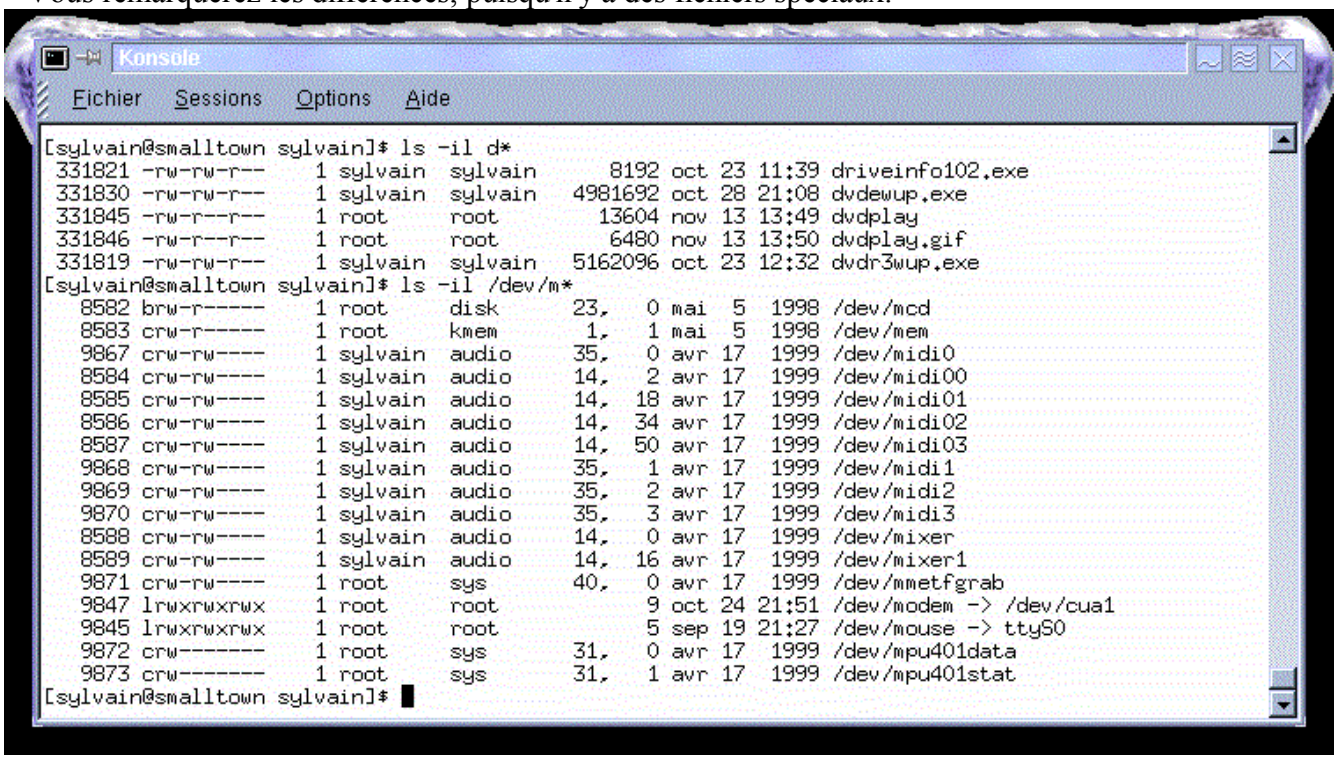
Évitez absolument le -?&><* et autres \, dont la gestion sera bien difficile ensuite (imaginez vous devoir effacer un fichier nommé -Rf par exemple !!!).

Le chemin d'accès à un fichier peut être soit relatif (dépendant de la position actuelle, en utilisant éventuellement . et ..), soit absolu (commençant donc par une référence à la racine, le /).

En fait, à chaque fichier, Linux (et les autres unix) font correspondre une inode. Une inode est un numéro qui désigne le fichier, son contenu (du moins, la liste des blocs correspondant sur le périphérique) et ses attributs (propriétaire, droits, dates...)

Sur l'écran suivant, nous voyons le résultat d'une commande ls -il, qui affiche les informations concernant les fichiers (inodes, droits, propriétaires, taille dates et noms).

Vous remarquerez les différences, puisqu'il y a des fichiers spéciaux.



Voir la commande **ls** (option *-i* pour inode).

Pour consulter les informations d'une inode, utilisez la commande stat (pas très standard, donc pas automatiquement installée).

Les droits sont gérés par les commandes suivantes :

chmod, **chown** et **chgrp** (droits, propriétaire, et groupe)

voir les modes symbolique et arithmétiques

voir le cas particulier des répertoires !!!

démonstration :

les utilisateurs pierre et sylvain appartiennent au groupe prof. L'utilisateur guest appartient au groupe users. Le fichier /tmp/essai existe, et est en lecture pour tout le monde. Sachant que la commande pwd nous affiche /d1, complétez les tableaux suivants :

drwxr-xr--	2	pierre	prof	d1
-rw-rw-r--	1	pierre	prof	d1/f1

commandes	***Pierre***	***Sylvain***	***guest***
cp /tmp/essai f1			
cp /tmp/essai f2			
chmod 666 f1			
rm f1			

drwxr-xr-x 2 pierre prof d1

-rw-r--r-- 1 pierre prof d1/f1

commandes	***Pierre***	***Sylvain***	***guest***
cp /tmp/essai f1			
cp /tmp/essai f2			
chmod 666 f1			
rm f1			

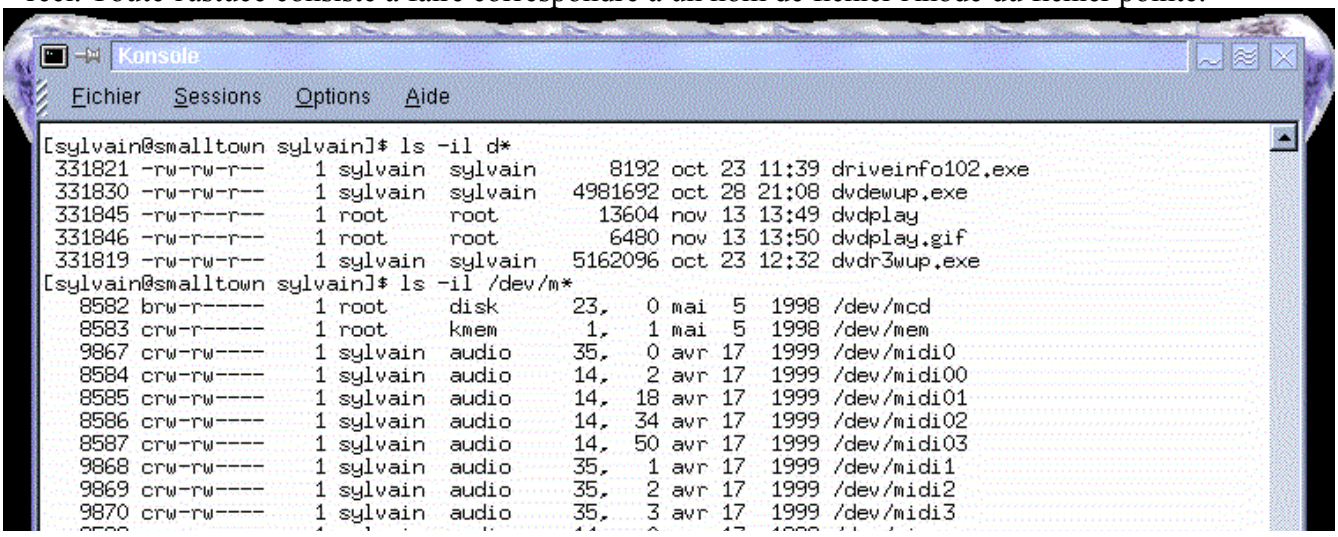
drwxr-xr-x 2 pierre prof d1

--------- 1 pierre prof d1/f1

commandes	***Pierre***	***Sylvain***	***guest***
cp /tmp/essai f1			
cp /tmp/essai f2			
chmod 666 f1			
rm f1			

6.a - Les liens

Les liens sont une solution élégante à divers problèmes de maintenance des fichiers, et de leurs emplacements dans l'arborescence. Il s'agit en fait d'une redirection d'un nom fictif vers un fichier réel. Toute l'astuce consiste à faire correspondre à un nom de fichier l'inode du fichier pointé.



La création d'un sosie de dvdplay.gif se fait par la commande ln nom sosie. Les deux noms

correspondent réellement au même contenu (voir le numéro d'inode). Remarquez le 2 qui apparaît en deuxième argument du ls -l. Le rm du premier fichier n'efface que la correspondance entre ce nom et l'inode. L'inode reste présente tant qu'un nom pointe dessus. Le contenu ne sera supprime que lorsque le dernier nom de fichier le sera. On peut avoir autant de liens que voulu.

Question : Si vous avez compris ce que sont les montages, quel problème peut on imaginer si on lie deux noms de fichiers de deux file système différents ? Quelle est la solution à utiliser ? (man ln)

6.b - La commande find

Cette commande permet de rechercher un ou plusieurs fichiers, selon des critères. Il est même possible de lancer des actions sur le résultat (effacement, copie, etc).

La syntaxe est la suivante : find [chemin] [critères]

les critères sont des expressions, qui si elles donnent vrai, entraîne la sélection du nom du fichier : les expressions sont des combinaisons des critères suivants :

Critères	***Explications***
-name nom	Nom du fichier recherché
-user nom	Le propriétaire du fichier est nom
-group nom	Même chose avec le groupe
-type	Répond vrai selon le type de fichier (b pour bloc, c pour caractère, f pour fichier et d pour répertoire, l pour un lien)
-size taille	Selectionne si la taille correspond (a utiliser avec + ou - pour plus grand que ou plus petit que, et c b k pour indiquer l'unité de mesure de grandeur...)
-inum numero	Donne tous les fichiers portant ce numéro d'inode
-atime numéro -mtime numéro -ctime numéro	Le fichier a été accédé, modifié ou création il y à moins de numéro jours. (voir aussi l'option daystart, qui permet de partir de minuit et non de 24 heures plus tot...)
(expr -or expr) (expr -and expr) ! expr	Il est possible de jouer avec des et et des ou, en regroupant les commandes dans un ou plusieurs groupes. ! inverse l'expression : ! -name toto veut dire qui ne s'appelle pas toto.

Une fois les noms sélectionnés, il est possible

- de les afficher (commande - print)
- d'exécuter une commande (-exec rm {} \;)
- de d'interroger l'utilisateur avant d'exécuter la commande (-ok rm {} \;)

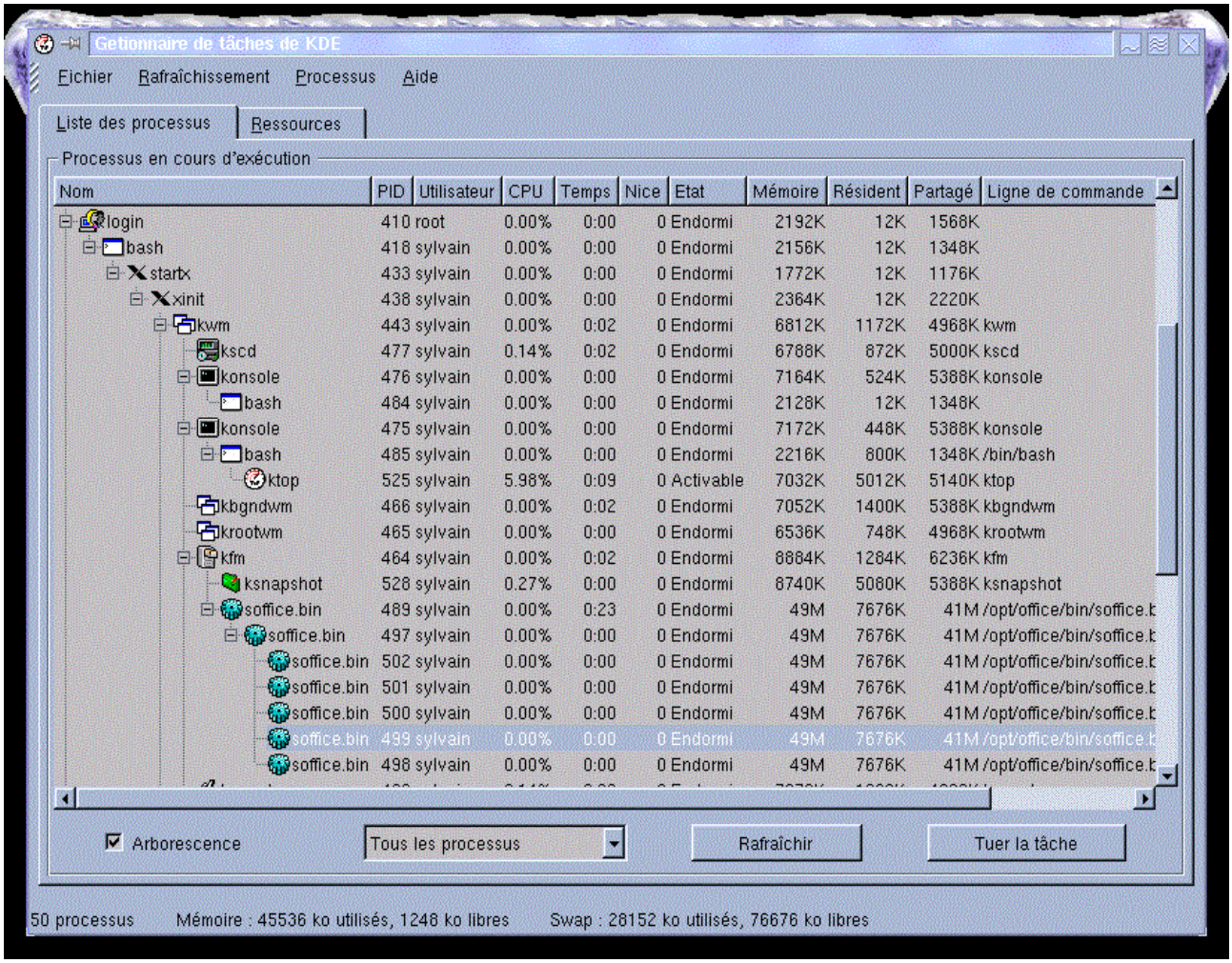
le {} indique le nom de chaque fichier trouvé, et il faut absolument finir par \; (ATTENTION a l'antéslash, il évite que le bash imagine que le ; le concerne !!!)Dans l'exemple, j'ai utilisé rm, mais on peut lancer n'importe quel commande bien sûr. Par exemple, voici une commande qui recherche tous les programmes sources en C, et qui en fait une copie (.old)... ***find . -name "*.c" -exec cp {} {}.old \;*** *(recherche tous les fichiers qui finissent par .c, et pour chacun d'eux, exécute copie nom en nom.old...)*

7) Les processus.

Unix est un système d'exploitation multitâches multi-utilisateurs. Unix utilise plusieurs processus, qui s'enchaînent les uns les autres, ou sont en concurrence. Le démarrage de la machine utilise ce système de processus qui lance d'autres processus (principe du bootstrap).

7.a - Qu'est ce qu'un processus ?

Un processus est l'image d'un programme en mémoire. Le programme est un fichier binaire, c'est à dire des commandes compréhensible par le processeur. Malheureusement, ces commandes seules ne suffisent pas, et votre programme s'appuie sur un ensemble d'informations fourni par le système d'exploitation. De même, lors de son exécution, ce programme va utiliser un espace mémoire de travail, et une pile (endroit où l'on stocke temporairement des données et/ou des adresses, par exemple lorsque l'on exécute une sous partie d'un programme (adresse de retour)).



Belle arborescence de processus, d'un login à une session X. Remarquez que l'on voit la commande en question (ktop) tourner, ainsi que le logiciel qui a permis de capturer l'écran (ksnapshot).

L'ensemble de ces informations s'appelle un processus. En résumé, un processus contient une zone utilisateur (répertoire courant, UID, références aux fichiers), une zone de code (le programme), une zone de données (zone de travail) et une zone de pile (paramètres passées entre différentes fonctions, adresses de retour, variables stockées à la volée et non mémorisées...)

Chaque processus a un numéro unique, un lien vers son père (celui qui l'a crée...), un numéro utilisateur, de groupe, la durée de traitement, sa priorité d'exécution, la référence à son répertoire de travail....

le premier processus qui est lancé est init, qui lance ensuite les suivants.... Votre bash est le fils de votre login, lui même fils d'un gettty sur votre terminal (attente d'un connexion), etc....

Lorsque vous tapez une commande, celle ci lance un nouveau processus, dont votre bash est le père.

La création des processus est régie par le principe suivant. un processus crée un processus fils, en forkant (appel système de fork (fourche), qui crée un double du processus père, en lui donnant un nouveau PID). Une fois le processus dupliqué, un appel système exec lance le programme appelé. Le processus père lui est en état d'attente (appel system wait). Exec, sur le processus fils, effectue ce que l'on appelle un 'recouvrement' : Le fils, qui était un clone parfait du père (a tel point qu'il faut faire attention en tant que programmeur, chaque processus croit être « le vrai » !) doit se transformer en un autre programme (c'est souvent ce que l'on veut faire : je suis dans un shell, je veux lancer un ping, je tape donc ping: Mon processus d'origine se dédouble (le shell est donc dupliqué !!). Le fils va ensuite se transformer en ping (puisque c'est ce que l'on voulait, un ping), tandis que le père (le vrai shell) se met en attente de son fils. Quand le fils aura terminé, il le signalera à son père, qui pourra alors reprendre son execution.

7.b - Les états d'un processus

R pour runnable

S pour Sleeping

D pour uninter. Sleep (attente Entrée Sortie)

Z pour Zombie (plus de père, 'parti dans les choux !!!')

7.c - Les tâches de fond.

De par ce principe de multi tâches, il est possible de lancer une tâche en arrière plan, et de continuer à travailler sur le terminal. Il suffit de taper un & en fin de commande. Le numéro du processus fils s'affichera, et l'appel système au wait est supprimé. On peut donc continuer à travailler. Attention cependant, les messages de sortie de la commande en arrière plan arriveront sur votre terminal...

On appelle parfois ce mode l'exécution de processus asynchrone (plus de dépendance de temps entre le père et le fils).

Attention : Le fait de vous déloguer arrêtera tous vos processus, dont ceux en tâches de fond...

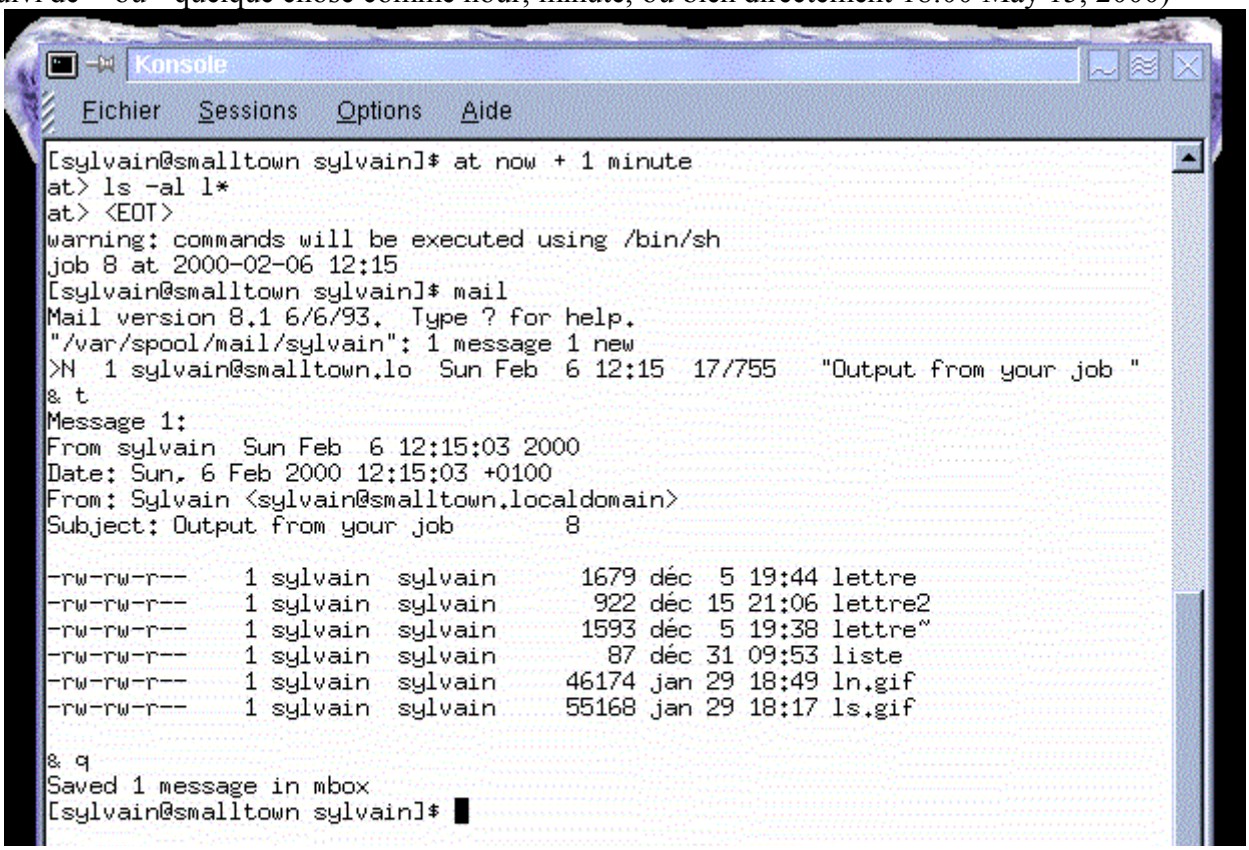
7.d - Manipulation des primitives du système

& n'est pas la seule possibilité de modifier le comportement du système multitâche de Linux. On peut aussi contrôler le fork (empêcher le fork de se faire, c'est donc notre processus courant qui sera 'recouvert'). On peut aussi, avec nohup, donner notre fils à un processus plus ancien (dans la pratique, nohup permet de lancer une commande qui continuera à s'exécuter même aprés notre déconnexion (nous n'en sommes plus le père)).

7.e - les processus différés

at permet le lancement de tâches à une heure précise. Le résultat du jobs sera dans votre mail (lancement par mail, lecture du mail par t, sortie par q).

at accepte de multiples façon d'indiquer l'heure (par rapport à maintenant, à midi, à minuit, aujourd'hui, demain, ou une date et une heure précise) : now, noon, midnight, today et tomorrow suivi de + ou - quelque chose comme hour, minute, ou bien directement 18:00 May 15, 2000)



exemple : dans 1 minutes, un ls -al de tout ce qui commence par l, puis une lecture du mail. Remarquez l'écriture de la commande at, avec un passage à la ligne, et fin de saisie par l'habituel CTRL D (EOF).

Le cron permet quand à lui la programmation de taches à intervalles réguliers. La tache demandé sera lancée automatiquement, sous l'identité d'un utilisateur, même si celui ci n'est pas connecté...

Le fichier est constitué de la façon suivante

min heure jour mois jourSemaine commande

la création et la gestion de ce fichier se fait par la commande suivante : crontab -e (pour éditer) ou -l pour l'afficher...

exemple de fichier crontab (en fait, il s'agit d'un fichier portant le nom du user, rangé dans /var/spool/cron)

*00 08 * * 6 updatedb*

*30 14 01 01 * shutdown -r now*

à 8 heures, tous les samedi (6 eme jour de la semaine), exécution de updatedb (base de données des noms de fichiers, consultable ensuite par locate.

le 1er janvier de chaque année, à 14H30, redémarrage de la machine...

7.f - les commandes.

Outre le & (lancement en tache de fond), voici des commandes utiles pour la gestion des processus :

ps affiche les informations sur les processus (option axl)

kill envoie un signal au processus (permet aussi d'arrêter la tache, selon le signal)

top affiche à intervalle régulier l'ensemble des taches

pstree affiche une arborescence des taches

time permet de mesurer les temps d'exécution

jobs affiche les taches en fond.

nice change la priorité d'une tache

renice permet la gestion de la priorité (même après son lancement)

sleep 'endort' une tache (la suspend)

nohup évite à une tache de fond d'être tuée par l'arrêt de son père

bg et **fg** permet d'envoyer en fond ou de récupérer une tache de fond en premier plan

wait met le processus courant en état d'attente de ses fils asynchrones

8) Les Files Systems

Linux reconnaît de multiples FS. Il est capable de travailler sur son FS de base (actuellement ext2fs), mais aussi sur les FS dos, vfat, et fat32. NTFS est traité en lecture seulement (problème d'accès à la

SAM)....

8.a - les commandes

8.a.1) Le formatage de bas niveau (les disquettes)

Il s'agit de la commande fdformat, suivi d'une device... ATTENTION : Il s'agit d'un nom de device plus complet que la normale : Il faut aussi indiquer la capacité (pour connaître le format). On utilise en général un pseudo nom (fd0H1440, ou fd0u1440)

Par exemple : fdformat /dev/fd1u1440

8.a.2) le formatage haut niveau (File System)

Une fois le support formaté (bas niveau), on peut avoir besoin d'y déposer un file système., par exemple msdos, ou ext2fs. Ce n'est nécessaire que si vous voulez y gérer des fichiers et/ou des répertoires. La commande mkfs permet de créer des files systèmes, pourvu que vous disposiez du driver approprié.

Par exemple : **`mkfs -t ext2  /dev/fd0`**, ou **`mkfs -t msdos /dev/fd1`**

En fait, mkfs est un frontal qui va rechercher le programme mkfs.fs correspondant (mkfs.msdos dans le deuxième exemple).

Pour les tricheurs, vous pouvez utiliser les mcommandes, c'est à dire le portage des commandes msdos sous Linux. Citons mformat, mcopy qui s'utilisent exactement comme leurs homologues MSDOS...

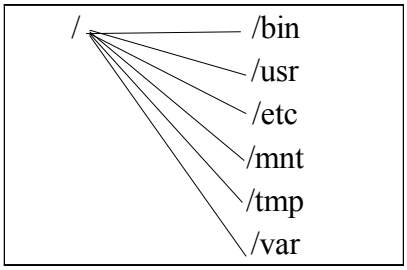
8.a.3) le montage

Pour utiliser un support magnétique formaté, vous allez devoir le monter, c'est à dire le faire apparaître à un endroit quelconque de votre arborescence. Vous allez désigner un répertoire qui, après l'opération de montage, fera apparaître le contenu (répertoires et fichiers) du support magnétique. En général, le répertoire /mnt est dédié à ce genre d'opération, et c'est pourquoi on y trouve deux sous répertoires : floppy et cdrom. Vous pouvez cependant monter un file système dans n'importe quel répertoire, et ce à chaque montage. Cela permet de multiples astuces d'administration. Remarquez que c'est le principe employé pour répartir vos données sur plusieurs partitions (système souvent utilisé dans les installations de serveurs).

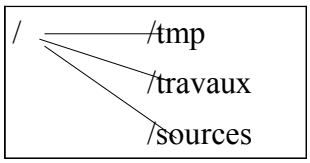
La commande est la suivante

mount [-t FS] device répertoire

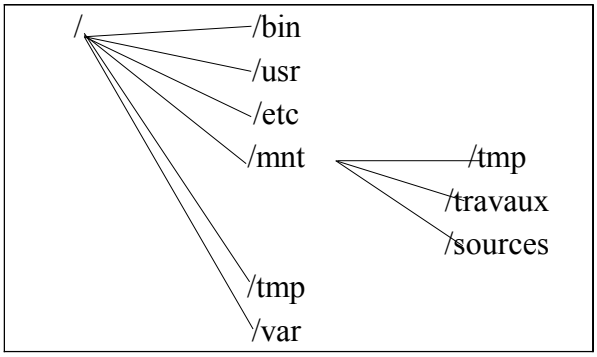
Un petit schéma explicatif du principe du montage



Une arborescence



Une disquette, qui contient un File Systeme...



*Arborescence résultante, après la commande **mount /dev/fd0 /mnt***

Pour lire une disquette dos et gérer son contenu dans /mnt/dkt (répertoire créé pour l'occasion)

mount -t msdos /dev/fd0 /mnt/dos

Ce répertoire est maintenant lié à la disquette. Si ce répertoire contenait des fichiers et/ou des sous répertoires, ceux ci seront masqués par le nouveau FS monté. Vous les récupérerez au démontage.

Lors que l'on veut enlever la disquette, il faut la démonter. Il ne faut plus qu'aucun processus ne travaille sur ce répertoire (ni aucun shell !!!). Il suffit de taper umount suivi soit du périphérique, soit du répertoire...

umount /dev/fd0

N'oubliez jamais de démonter un périphérique !!! Un flag se lève dans le superbloc à chaque montage, et n'est baissé qu'au démontage. En cas de flag resté levé, une vérification est automatiquement lancée. Remarquez à ce propos la gestion des cd-roms par Linux : Le lecteur ne s'ouvre pas tant que le cdrom est monté...

pour monter un CDROM, utilisez la commande suivante

mount -t iso9660 /dev/hdc /mnt/cdrom (si votre lecteur cd est le master du secondary IDE)

mount tout seul affiche tout les montages en cours....

mount est une commande réservé à root (sauf indication particulière dans /etc/fstab...)

8.a.4) Rôle de /etc/fstab

On peut parfois simplifier l'utilisation de mount grâce à /etc/fstab. Ce fichier contient l'ensemble des montages à effectuer en automatique, ou à connaître. En général, l'administrateur y a indiqué les montages les plus courants. Résultat : la frappe de mount /mnt/floppy ou de mount /dev/fd0 suffit pour monter la disquette

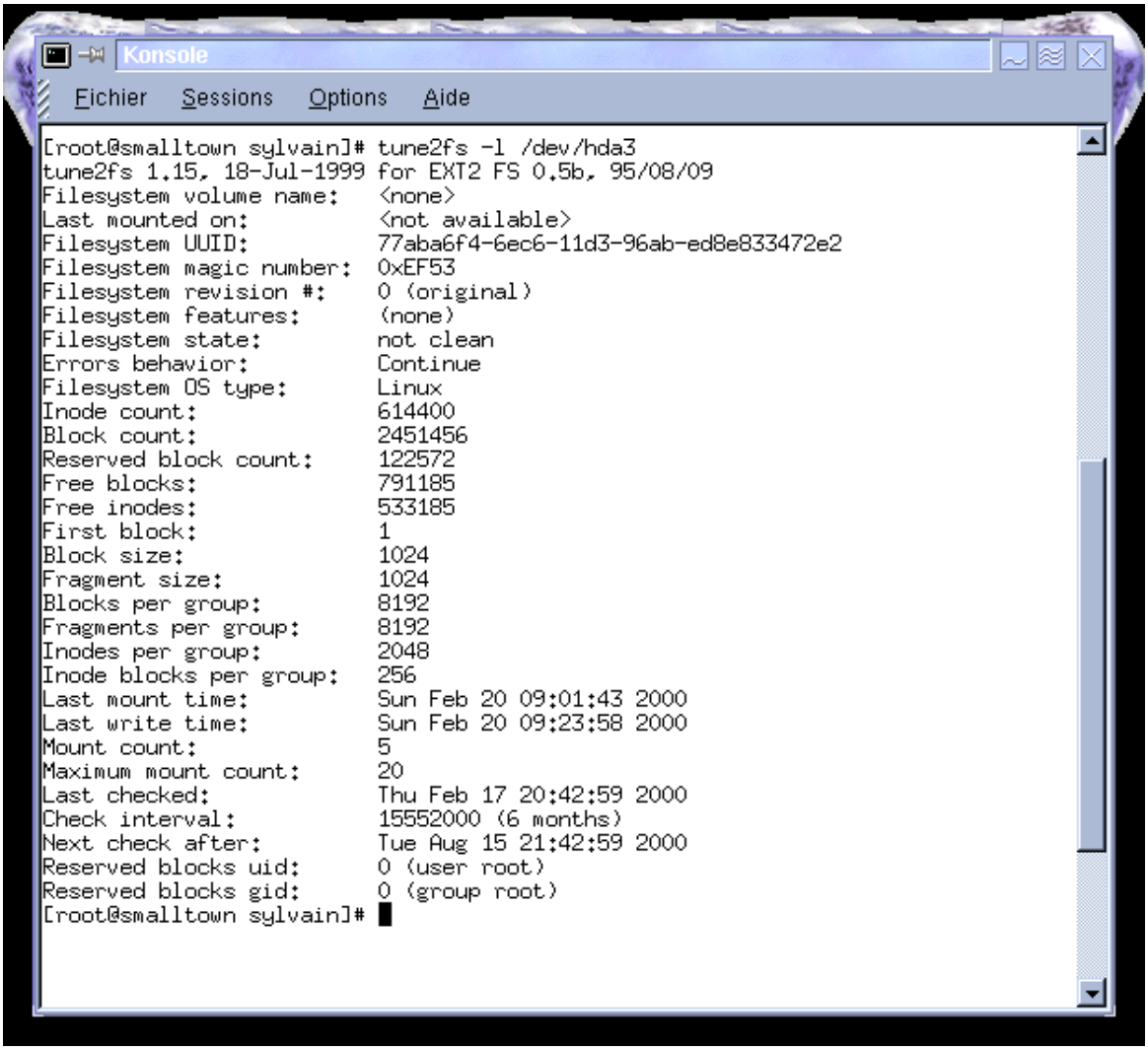
```
/dev/hda3       /           ext2    defaults                        1 1
/dev/hda1       /dosc       vfat    defaults                        0 0
/dev/hdb1       /dosd       vfat    defaults                        0 0
/dev/hdb5       /dose       vfat    defaults                        0 0
/dev/hda2       swap        swap    defaults                        0 0
/dev/fd0        /mnt/floppy auto    sync,user,noauto,nosuid,nodev,unhide 0 0
/dev/cdrom      /mnt/cdrom  auto    user,noauto,nosuid,exec,nodev,ro 0 0
none            /proc       proc    defaults                        0 0
#none           /dev/pts    devpts  mode=0622                       0 0
```

un exemple de /etc/fstab : chaque périphérique, son point de montage habituel et son type de FS apparaît

8.a.5) Autres programmes associés

tune2fs permet d'afficher les informations concernant un file système ext2fs. On peut y contrôler la taille des blocs, le nombre d'inodes, l'état de la partition, la configuration de contrôle (nombre de montage depuis la dernière vérification, vérification automatique tous les x montages, ou après telle date...). Attention, il est recommandé de travailler sur une partition NON MONTEE, si l'on veut pouvoir modifier les réglages.

tune2fs d'une partition montée (ne pas faire de modif dans ce cas !!!) : ce qui explique le not clean...



8.a.6) le cas particulier du swap

LA partition de swap est utilisé par Linux pour stocker temporairement des données lorsque la mémoire RAM est saturée. On peut utiliser soit un fichier de swap (comme sous windows), ou plutôt ce système qui est plus propre. La partition de swap peut être une partition étendue. Sa taille ne peut pas dépasser 128 Mo (mais il est possible d'avoir autant de partitions swap que l'on veut), et est comprise habituellement entre 1 à 2 fois la taille de la RAM.

Pour créer une partition swap, il faut utiliser **fdisk**, et utiliser le type de partition 82. Cette partition doit ensuite être formatée par la commande **mkswap** *device*. L'utilisation ou l'arrêt de ce swap se fait par **swapon** *device* et **swapoff** *device*. En général, tous ces réglages sont lancés automatiquement au démarrage de la machine... Voyez à ce propos le fichier /etc/fstab...

8.b - File System, ou pas ?

Il n'est pas obligatoire de créer un file systèm sur un support. Ce n'est utile que si vous voulez gérer plusieurs fichiers sur ce support. Si il ne doit contenir qu'un seul ensemble de données, on peut considérer qu'il est inutile de créer le fs.
Par exemple, pour créer une disquette de boot Linux, il suffit de "cracher" le noyau Linux sur la disquette. La commande qui permet d'écrire en direct est dd (disk dump). Ses arguments important sont if (input file : l'entrée) et of (output file : la sortie). On peut préciser bs (block size, taille du bloc) et count (qui permet de limiter à un certain nombre de blocs).

Petits exemples :
Copions un noyau Linux (vmlinuz, situé dans le répertoire /boot) sur une disquette, afin de booter le systeme... L'option bs correspond à la taille d'une piste d'une disquette : La copie est alors plus rapide...

```
dd  if=/boot/vmlinuz   of=/dev/fd0   bs=18k
```

l'équivalent d'un DISKCOPY A: A: de MS-DOS se réalise via un fichier temporaire.

```
dd    if=/dev/fd0    of=/tmp/copydkt      bs=18k
dd     if=/tmp/copydkt     of=/dev/fd0     bs=18k
```

Ce système peut sembler lourd, mais pour apprécier sa puissance, remarquez que l'on peut ensuite dupliquer l'image à volonté (2ème ligne seulement). D'autre part, il est possible des faire des dd de n'importe quel support (partition, disque dur complet, cd, dvd...) vers n'importe quoi (fichier, partition, disque dur, cd...) Par exemple, pour copier la MBR, utilisez dd sur les 512 premiers caractères du disque dur :

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

8.c - File System ou pas (suite)

Maintenant que nous doutons de l'utilité d'un file système sur un support, inversons notre propos. Et pourquoi ne pas créer un fichier, qui contient lui même un file système (c'est à dire une organisation de répertoires et de fichiers).

Dans un but pédagogique, créons un tel fichier
tout d'abord créons un fichier d'une taille déterminée (admettons un méga octet)

```
dd if=/dev/zero  of=/tmp/fichierFS  bs=1k  count=1024
```

la device /dev/zero est une des petites favorites : Elle fournit une infinité de zéro. Elle a une petite soeur, /dev/null, connue sous le nom de trou noir, qui avale tout ce qui y est copié (vous connaissez certainement son utilité pour supprimer les messages d'erreurs...)

Créons maintenant un File System dedans !!!!

```
mke2fs  -F  /tmp/fichierFS
```

Le -F permet de forcer la création du FS, bien qu'il ne s'agisse pas d'un périphérique bloc.

créons un répertoire point de montage, et montons le fichier FileSystem dedans

```
mkdir /FSFile
mount  -o  loop  /tmp/fichierFS  /FSFile
```

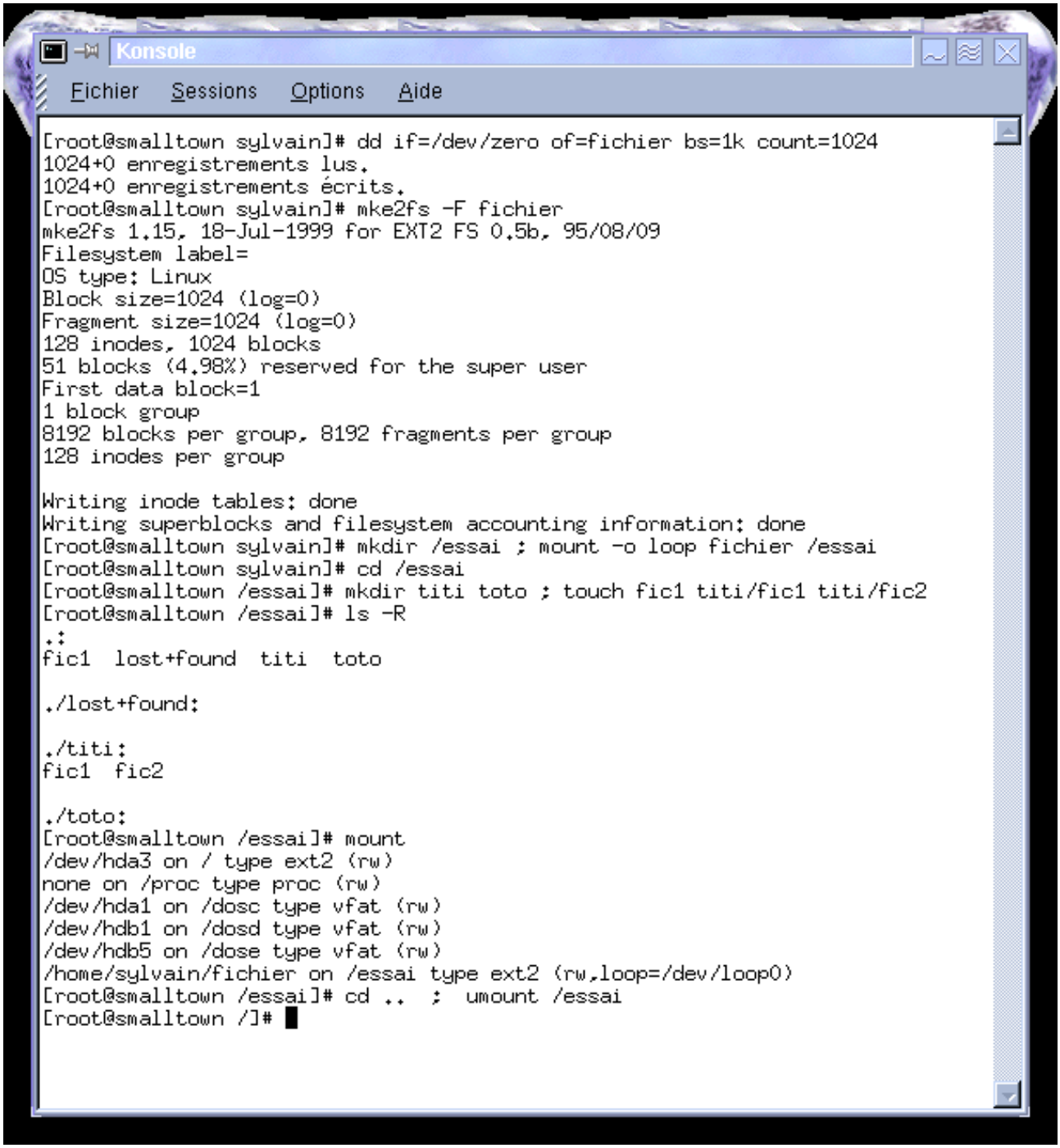
l'astuce repose sur l'utilisation de l'option loop, qui permet cette acrobatie. Elle peut prendre des options (cryptage...)

et maintenant, amusons nous un peu

```
cd   /FSFile
mkdir  toto  titi
touch  fic1  /titi/fic1  /titi/fic2
ls \R
```

cela fonctionne parfaitement. Il faudra ensuite démonter /FSFile

Notez enfin que ce système est utilisé couramment dans les installations de Linux : La deuxième disquette d'install (la root) contient en fait un tel fichier zippé (1,2 MégaOctet). Si vous le dézippez, et que vous le mounter, vous trouverez une arborescence complète Unix, contenant les binaires nécessaire à l'installation. Cela sert aussi à crypter tout un file system (man mount)



Un file système dans un autre, ou pourquoi faire simple quand on peut faire compliqué....

9) Sauvegardes et Packages

Linux, comme tous les unix, offre de outils de sauvegarde de fichiers. Ces systèmes permettent la création de fichiers archives, qui permettront aussi l'échange de logiciels complet (package). Il existe aussi des outils spécifiques pour la gestion des packages (ajout et suppression de logiciels)

9.a - dd

nous venons de voir dd, qui permet la création de copies physiques. Cet outil est assez satisfaisant pour le travail de bas niveau (création d'image disques, duplicata de partition complète), mais n'est pas toujours pratique (la copie d'une partition fait la taille de cette partition, même si elle est utilisée à 10%). Par contre, dd récupère une image stricte, ce qui permet de dupliquer même des file system inconnus...

9.b - tar

tar est une commande très courante, raccourci de Tape Archiver.
Tar permet la création de sauvegarde d'une liste de fichier soit dans un fichier archive, soit sur un support (/dev/fd0 par exemple). Un de ses avantages est sa récursivité. Il peut descendre dans les sous répertoires.

Les options de tar	***effets***
cc	Créer une archive
tt	Test une archive
xx	Extrait le contenu de l'archive
vv	Mode verbeux
ff	l'argument suivant sera le nom du fichier archive (régulier, ou spécial...)
zz	l'ensemble est compressé (par gzip)
M	Sauvegarde en multi volume (pratique si l'archive est plus grande que le support amovible)

l'utilisation de tar est la suivante :
tar option [fichier archive] [liste des fichiers à sauvegarder]
par exemple, sauvegardons l'ensemble des home directory sur la disquette (sans file systèm : la disquette n'est donc pas montée !!!)

```
tar   cvf    /dev/fd0   /home
```

même chose pour une archive zippée, placée cette fois ci dans /tmp

```
tar   cvzf /tmp/archive.tgz   /home
```

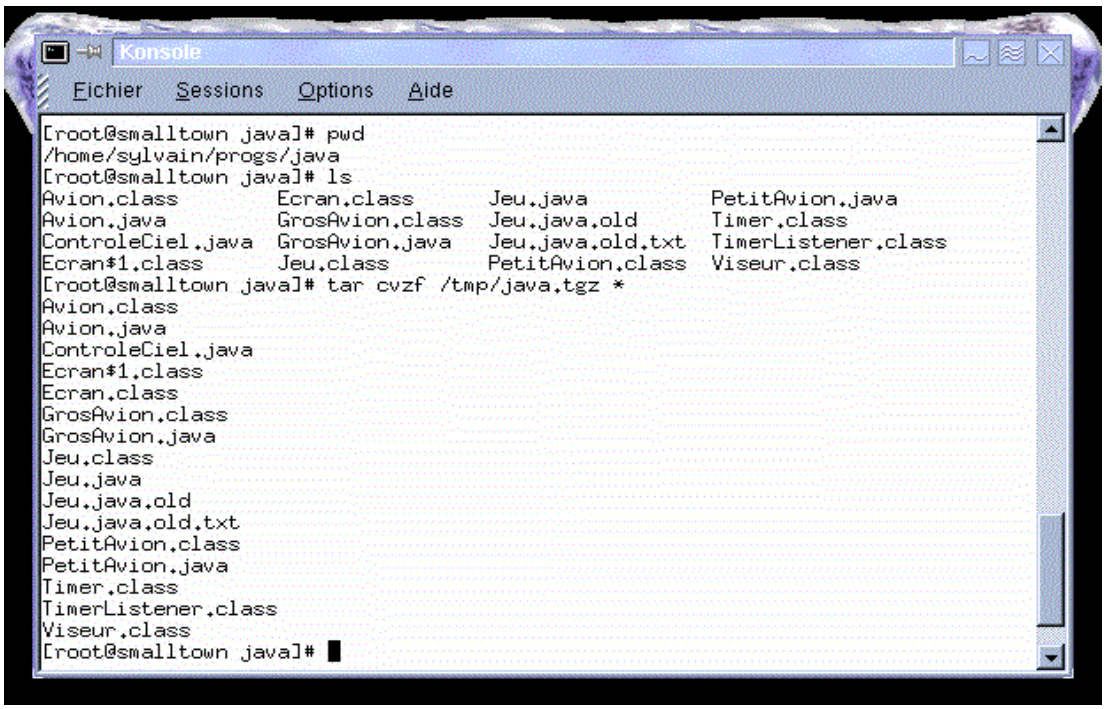
remarquez l'extension .tgz, qui permet de comprendre qu'il s'agit d'un tar zippé (mais en cas de doute, utiliser toujours file)
test d'une archive

```
tar   tvzf  /essai/sav/tgz
```

restauration de l'archive situé sur la disquette (nom montée !!)

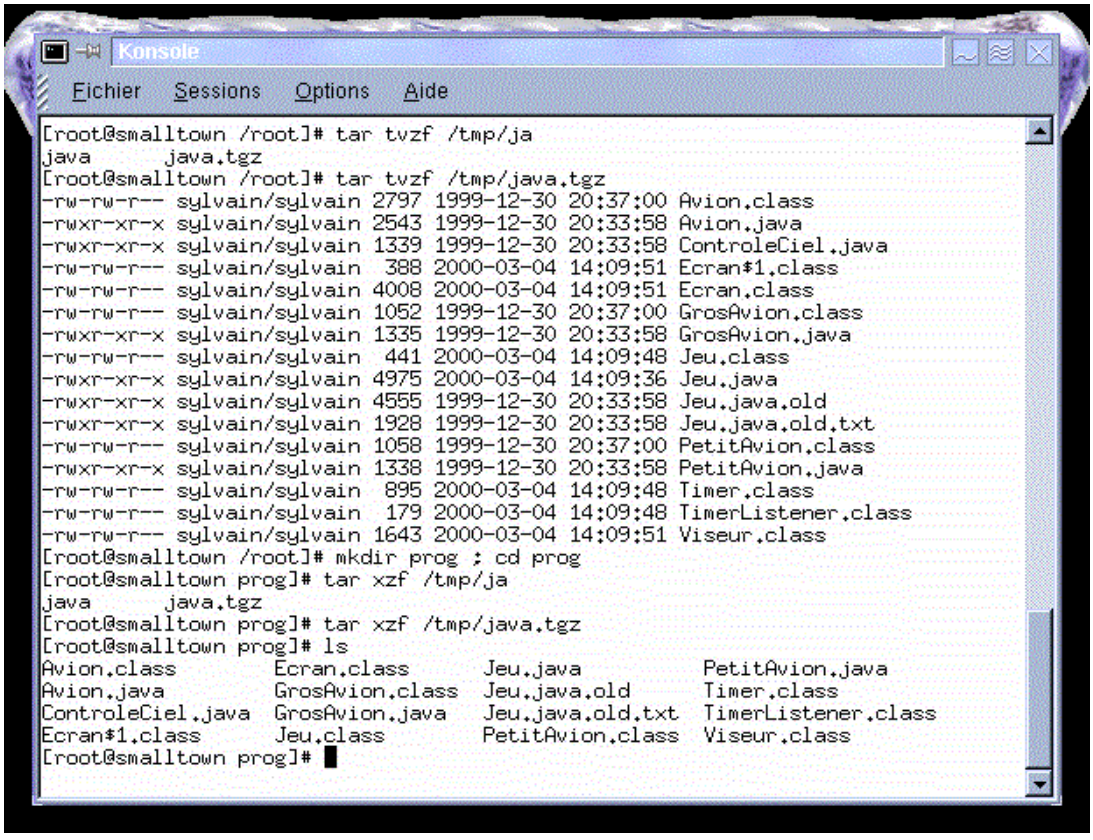
```
tar   xvf /dev/fd0
```

Attention : Puisqu'il est récursif, tar mémorise le chemin d'accès au fichier, selon l'endroit duquel il a été lancé... Vérifiez toujours ce que l'archive contient, avant de l'extraire, ce afin d'éviter que les fichiers se restaurent n'importe ou !!!



création d'une archive zippée (enregistrée dans /tmp) du répertoire courant...

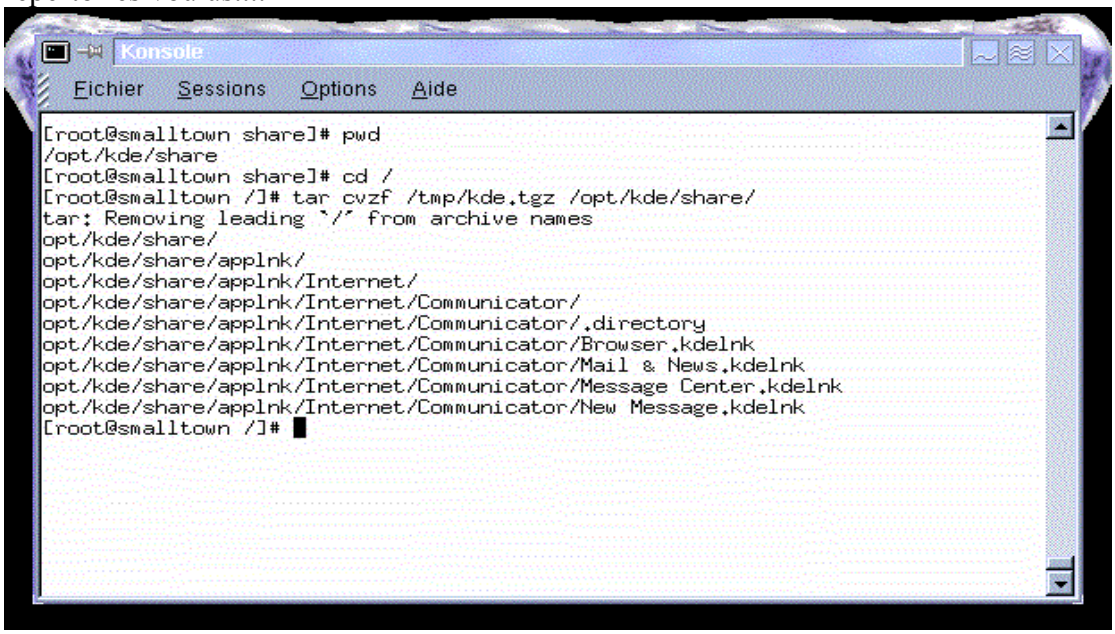
L'utilisation d'une telle archive impose à la personne qui la récupère de créer un répertoire pour y déposer les fichiers.



après un test, l'utilisateur s'aperçoit que les fichiers sont stockés directement dans l'archive. Il décide donc de créer un sous répertoire pour y décompacter l'archive

Très souvent, on préfère se positionner au dessus du répertoire à sauvegarder, afin de mémoriser le

chemin, ou une partie du chemin, dans l'archive. A la récupération, cela créera automatiquement les répertoires voulus....



fichiers sauvés, avec leurs chemins...

Pour récupérer le contenu de cette archive, l'utilisateur pourra se positionner dans /, et extraire le contenu...
Outre les sauvegardes, tar sert aussi à la diffusion des produits. On appelle ces versions des tarballs. Les versions de la Slackware sont à l'origine distribuées de la sorte...

9.c - cpio

cette commande sert aussi pour la gestion des données. L'option -o créé des sauvegardes, tandis que -i restaure les archives...
cpio est plus efficace que tar en cas de pépin sur la sauvegarde (disquette problématique, pb de bandes...).
En mode création de sauvegarde, cpio attend une liste de fichiers sur son entrée standard. On utilise par exemple des pipes pour l'alimenter. Le fichier archive sort sur sa sortie standard... A vous de le rediriger, par exemple dans un fichier.
En mode restauration, cpio attend un fichier archive sur son entrée standard, et restaure dans le répertoire courant.
Exemple
création d'une archive.tgz

```
ls | cpio -o >/tmp/toto.cpio
```

la liste des fichiers du répertoire courant est donnée à cpio, qui crée alors une archive /tmp/toto.cpio

restauration

```
cpio -i </tmp/toto.cpio
```

déplacement de répertoire

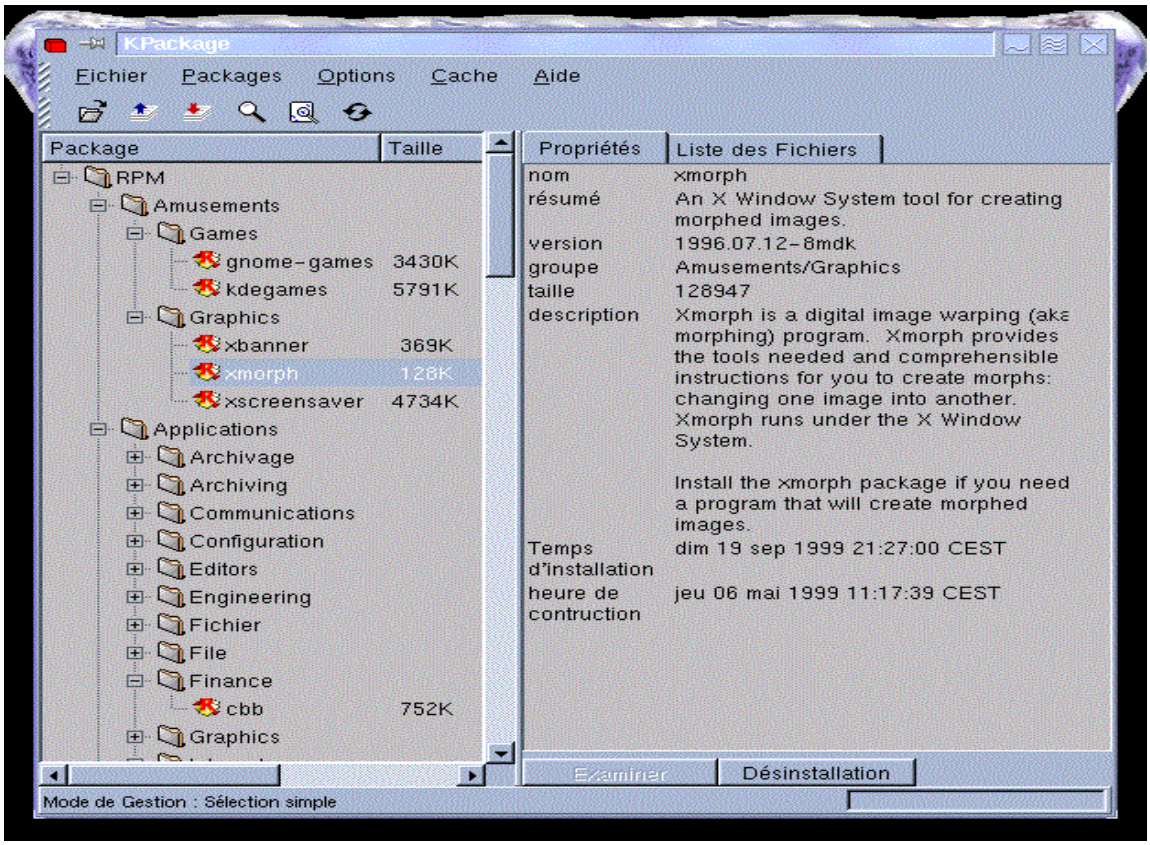
```
(cd /root ; ls | cpio -o ) | cpio -i
```

9.d - Les packages

La distribution des logiciels est souvent organisées sous forme de packages. L'intérêt du système est d'automatiser l'installation, en intégrant l'ensemble des fichiers nécessaires, les divers emplacements de copie, ainsi éventuellement qu'une vérification de la présence d'autres packages indispensables... De plus, certains outils sont distribués sous forme de binaires (compilés donc pour un processeur précis, avec des appels à certaines librairies, etc...), parfois sous forme de sources à recompiler... L'essor de Linux, et des diverses distributions utilisées, a donné lieu à la création de plusieurs systèmes de packages. On trouve ainsi des packages sous format tarball (des tgz), ou sous format rpm (Red hat Package Manager), voire encore sous le format Debian (DEB)...
En ce qui concerne les tarballs, reportez vous à la commande tar, plus haut. Il n'y a pas de vérification des dépendances... Testez d'abord l'archive, puis décompactez là, et cherchez la doc (README, ou encore install...)

9.d.1) RPM

En ce qui concerne les rpms, un processus de vérification permettra de s'assurer que les ressources nécessaires sont bien présentes sur la machine. Le principe du rpm est d'utiliser une base de données qui contient la liste des packages installés, et permet la vérification des dépendances...
la commande rpm permet l'installation, la mise à jour ou la suppression des packages installés...
rpm -i [package] permet d'installer un package (souvent ivh, vérification plus hashing)
rpm -U [package] met à jour un package déjà présent...
rpm -q [options] interroge la base de données de packages installés...



le kpackage, qui interroge les packages installés...

Comme toujours, l'outil manuel est très efficace. Il existe cependant une version graphique très simplifié. Sous KDE, il s'agit de kpackage. Il peut être invoqué de deux façons : soit en cliquant sur

un package : lancement automatique de kpackage, avec possibilités d'installer le package, soit en lançant kpackage, et en indiquant l'endroit ou sont les packages non installés. Chaque environnement graphique offre un frontal pour ces installeurs...

9.d.2) DEBIAN

l'installation sous Debian et ses dérivés utilisent un ensemble d'outils dont la base est dpkg (l'installateur en lui même) et divers frontaux (de dselect à apt-get).
Les fichiers à installer sont sous format .deb.
dpkg -i *package.deb* lance l'installation de ce package, en rangeant tout au bon endroit et en mettant à jour la base de données de votre machine... **dpkg** -r *package* le desinstalle, tandis que **dpkg -i -R** répertoire lance l'installation de tous les packages contenus dans le répertoire. L'intérêt de cet outil est sa puissance, puisqu'il tente de vérifier les dépendances, et d'installer en conséquence. Si malgré tous les tests, votre installation s'interrompt pour cause de dépendances non satisfaites, il est possible de relancer la fin d'installation d'un package qui resterait inaboutie, avec **dpkg --configure** *package*. Pour finir l'installation d'un ensemble de package qui se serait planté, utilisez **dpkg --configure --pending.**
Autres options : **-l** pour lister les packages installés sur votre machine, **-L** package pour lister les fichiers mis en jeu par ce package, **-S** fichier pour retrouver de quel package dépend le fichier, et enfin **-I** *package.deb* pour récupérer des informations détaillées sur le package en question...
apt-get est un frontal très simple qui permet la maintenance et la mise à jour de votre système. Ce programme interroge un fichier qui s'appelle *sources.list*, qui lui indique ou il trouvera les données d'installation standard (en général, les CD, ou un serveur dans votre réseau, voire même les serveurs Debian !!!). Il interrogera ensuite la base de données des installations et vous demandera d'introduire les Cds correspondant, ou se connectera au seveur voulu.
Apt-get install *package* lance l'installation de ce package, après vérification des dépendances, et de la pertinence de cet installation (numéro de version).
Apt-get remove *package* l'enlève.
Apt-get update met à jour l'installation, et **apt-get dist-upgrade** se connecte sur les serveurs, et met à jour tous les packages à partir des serveurs Debian...
apt-get source *package* récupère les sources du package voulu...
Attention, apt-get "rouspète" si vous avez fait des installations "sauvages"...
Enfin pour connaître (un peu) les packages disponibles, vous pouvez utiliser **apt-cache**, et notamment son option **search** qui interroge la base de données en comparant avec le mot clef fourni...
Aptitude est peut être plus interessant, car il regroupe toutes les commandes (contrairement à apt-get, *search* fait partie des commandes d'aptitude). Il gère aussi mieux les problèmes de dépendance.
Enfin, en mode graphique, de nombreux outils permettent la gestion des paquets (synaptic sous gnome, kadept sous kde, etc...)

10) La gestion des utilisateurs

Sous Linux, comme sous tous Unix, et autres systèmes multi utilisateurs, vous devez gérer des utilisateurs et des groupes. La sécurité du système est basée sur l'identification de l'utilisateur, et des droits qui lui sont associés...

La sécurité est basé sur deux voire trois fichiers. Il s'agit de /etc/passwd (qui contient la liste des utilisateurs, de leur nom, de leur home directory...), de /etc/shadows (qui contient les mots de passe en crypté, et qui n'est accessible qu'au root), et de /etc/groups qui contient le code et le nom des groupe.

Un utilisateur est défini par son ID (un numéro). Root a le numéro 0. Souvent, les simples utilisateurs ont un numéro supérieur à 100. L'utilisateur a un groupe par défaut. Un groupe est défini par son numéro (le GID). Il peut regrouper plusieurs users. Le système s'arrête là. Vous retrouvez ces notions dans la gestion des fichiers (user, group et other).

10.a - /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
lists:x:500:500:BeroList:/dev/null:/dev/null
gdm:x:42:42::/home/gdm:/bin/bash
xfs:x:100:102:X Font Server:/etc/X11/fs:/bin/false
sylvain:x:501:501:Sylvain Cherrier:/home/sylvain:/bin/bash
jeux:x:1002:1002:Connexion pour les jeux:/home/jeux:/bin/bash
```

les champs de ce fichier sont les suivant :

login:mot de passe crypté (ou x si shadow actifs):IDentifiant :Group Identifiant:Nom en clair (Gecos):Home directory:programme lancé à la connexion

Certains users sont des pseudo utilisateur, qui servent au lancement de programme ou pour la sécurité (xfs, ftp, shutdown, lists, halt).

10.b - /etc/group

Ce fichier contient la liste des groupes. On y trouve son nom, suivi du mot de passe du groupe (très peu utilisé), puis le Group Identifieur (qui est repris dans /etc/passwd), et enfin la liste des users appartiennent au groupe...

La question souvent posée est la suivante : comment faire pour qu'un utilisateur appartienne à plusieurs groupes ?

Réponse : root appartient aux groupes suivant :

le groupe 0 (d'après /etc/passwd, c'est donc son groupe par défaut, et il s'agit du groupe root).

Les groupes bin, daemon, sys, adm, disk, wheel, etc... puisqu'il apparaît dans le dernier champ dans /etc/group (groupes additionnels)

```
root::0:root
bin::1:root,bin,daemon
```

```
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
utmp:x:101:
lists:x:500:
floppy:x:19:
console:x:11:
gdm:x:42:
xfs:x:102:
pppusers:x:230:
popusers:x:231:
slipusers:x:232:
slocate:x:21:
sylvain::501:
audio:x:60:
jeux:x:1002:
```

10.c - /etc/shadow

fichier privilégié, qui contient les mots de passe cryptés, et qui n'est accessible qu'à root

```
root:$1$npqvHBzG$okPjWk/:10853:0:99999:7:::134542424
bin:*:10853:0:99999:7:::
daemon:*:10853:0:99999:7:::
adm:*:10853:0:99999:7:::
lp:*:10853:0:99999:7:::
sync:*:10853:0:99999:7:::
shutdown:*:10853:0:99999:7:::
halt:*:10853:0:99999:7:::
mail:*:10853:0:99999:7:::
news:*:10853:0:99999:7:::
uucp:*:10853:0:99999:7:::
operator:*:10853:0:99999:7:::
games:*:10853:0:99999:7:::
gopher:*:10853:0:99999:7:::
ftp:*:10853:0:99999:7:::
nobody:*:10853:0:99999:7:::
lists:!!:10853:0:99999:7:::
gdm:!!:10853:0:99999:7:::
xfs:!!:10853:0:99999:7:::
sylvain:LBiacHdehApY5.:10853:0:99999:7:::
jeux:2vbmkHtB4t5CPA:10955:0::7:::
```

La gestion des utilisateurs peut de faire de deux manières : en direct dans les fichiers (avec votre éditeur préféré), ou bien en passant par un frontal.

Habituellement, on trouve un utilitaire qui s'appelle useradd, et qui permet d'être guidé en mode texte. Adduser permet d'entrer dynamiquement un utilisateur. Depuis peu, certaines distributions prennent la liberté d'ajouter d'autres outils (bonne idée), et supprime les outils standards (mauvaise idée). C'est pourquoi vous pouvez tomber sur userconf (sur RedHat et Mandrake). Il existe aussi des outils graphiques (sous KDE).

/etc/skel est un répertoire qui fournit une config par défaut pour les utilisateurs nouvellement créés. Vous pouvez modifier son contenu, selon vos besoins...

11) les scripts SHELL

Le Shell est un interpréteur de commandes. Lorsque vous tapez une commande, il effectue différentes opérations et transfert la commande interprétée au noyau du système d'exploitation pour exécution. Dans le cas où la commande est simple, il suffit de la taper directement après le prompt (ie : en ligne de commande). Mais parfois une commande nécessite plusieurs arguments, des tests et des redirections, auquel cas il est préférable d'écrire un programme. Ce programme est en fait un simple fichier texte qui contient une suite de commande Unix. Il doit être exécutable (bit x activé). Utilisez l'agréable vi pour parvenir à vos fins.
Ce chapitre présente la façon dont est structuré un programme Shell et les structures auxquelles il fait appel. Plusieurs exemples de programmes Shell seront proposés en fin de chapitre afin d'illustrer ces propos.
Avant de s'attaquer au problème des scripts, allons plus loin en ce qui concerne le shell interactif.

11.a - Les jokers

Comme vous l'avez certainement remarqué, le shell gère ce que l'on appelle des caractères joker. C'est lui qui à la charge de convertir ces jokers en noms de fichiers, qu'il passera alors aux commandes invoquées.

```
$ ls
toto
titi
tutu
```

si je tape la commande cp * /mnt/dkt (afin de copier ces fichiers sur la disquette), c'est le shell qui s'occupera du remplacement par les noms des fichiers. En aucun cas, la commande cp n'a à gérer le joker *.

```
$ cp * /mnt/dkt

(le shell transforme cette ligne en cp toto titi tutu
/mnt/dkt)
```

Joker	***rôle***	***exemple***
*	Remplacement d'une chaîne de 0 à n caractères	**ls ab*** *ab1 ab2 ab3 ab4 ab10 abbaye.html about.txt*
?	Remplacement strict d'un caractère	**ls ab?** *ab1 ab2 ab3 ab4*
[xyz]	Définit un ensemble (ici x y et z)	**ls ab[13578]** *ab1 ab3*
[x-z]	Ensemble de x à z	**ls ab[1-3]?** *ab10*
[!x-y]	Tout sauf l'ensemble de x à y	**ls ab[!1-3]*** *ab4 abbaye.html about.txt*

11.b - Le quoting, la déspécialisation et l'évaluation

Il s'agit ici de mettre en place des protections face à l'interpréteur du shell. Selon les cas, on peut vouloir (ou ne pas vouloir) bloquer cet automatisme de replacement.

```
$ ls *
toto    titi    tutu
$ echo utilisez * pour tout remplacer
utilisez toto    titi   tutu pour tout remplacer
```

Embétant, non ? Pour parer à ce problème, vous allez pouvoir déspécialiser les caractères spéciaux (c'est à dire les rendre inactifs)
Première solution : l'échappement. L'utilisation de l'antéslash permet d'inactiver (de déspécialiser) le caractère qui suit.

```
$ echo utilisez \* pour tout remplacer
utilisez * pour tout remplacer
```

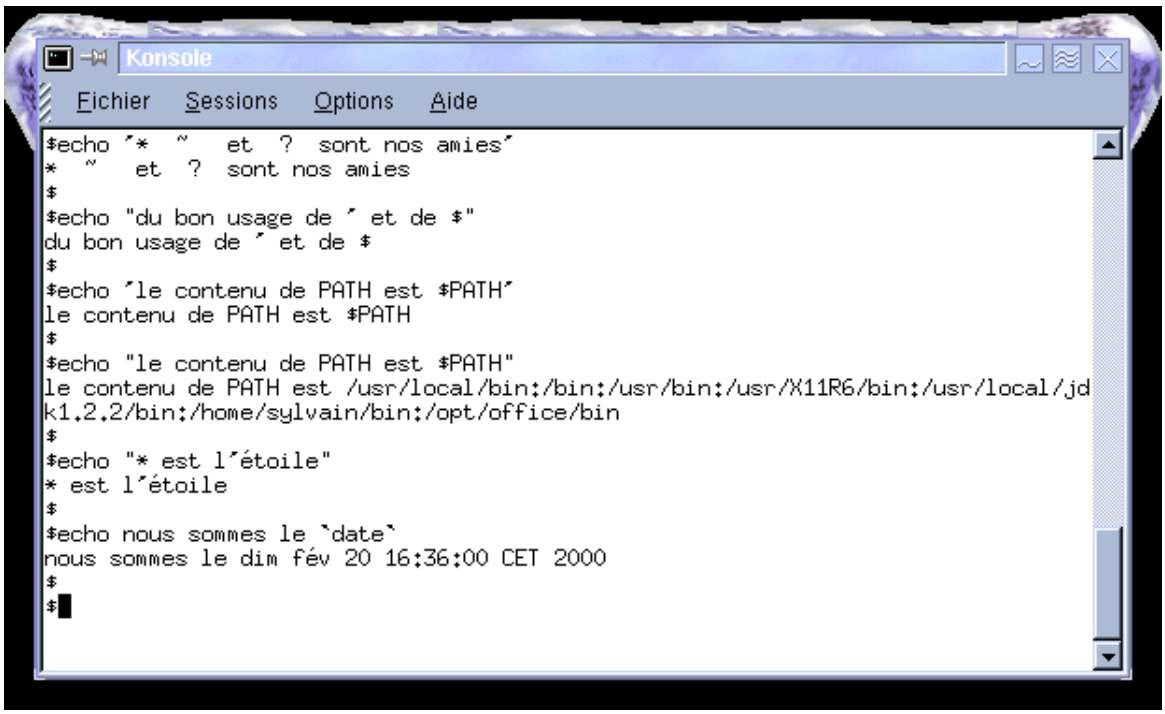
Très efficace, mais sachant que * ~ " ' et ? peuvent ou seront interprétés, ce système peut rapidement devenir lourd. Si par exemple vous devez afficher la phrase suivante : ** ~ " ' et ? sont interprétés ; okay ?*, voici ce que vous devriez taper :

```
$ echo \*   \~   \'   et  \?  sont interprétés \;  okay \?
```

Les apostrophes sont là pour vous aider : mais attention : il en existe 3 sortes : les simples quotes ('), les doubles (") et les inverses (`)
les simples bloquent tout, sauf elles mêmes (sinon comment s'arrêter ?)

```
$ echo '*  ~  et  ? sont nos amies...'
```

les doubles bloquent tout sauf $ et \ et elles même bien sûr. Elles servent donc à imprimer '.
Les inverses permettent au contraire de forcer l'évaluation. L'interprétation de la chaîne sera forcée (pratique pour mélanger des textes et des résultats de commandes...)



Des expressions qui apostrophent....

11.c - Élaboration d'un programme SHELL

11.c.1) Procédures à suivre

1. Il est bon (mais pas indispensable) de mettre l'instruction suivante en première ligne du programme :
   #!/bin/bash
   Cela indique à l'interpréteur que ce qui va suivre est du Shell (en l'occurrence du Bourne Again Shell - bash).
2. Le fichier contenant le code doit être rendu exécutable avant de lancer le programme. Ou alors, il faut le lancer en argument d'un shell ($ **`bash ./prog1`**)
3. Pour lancer le programme, il faut taper la ligne suivante :
   $ *prog* si votre variable PATH intègre le chemin du répertoire dans lequel vous lancez le programme
   $ *./ prog* Sinon...
   $ sh ./prog Au cas ou le fichier ne soit pas exécutable....

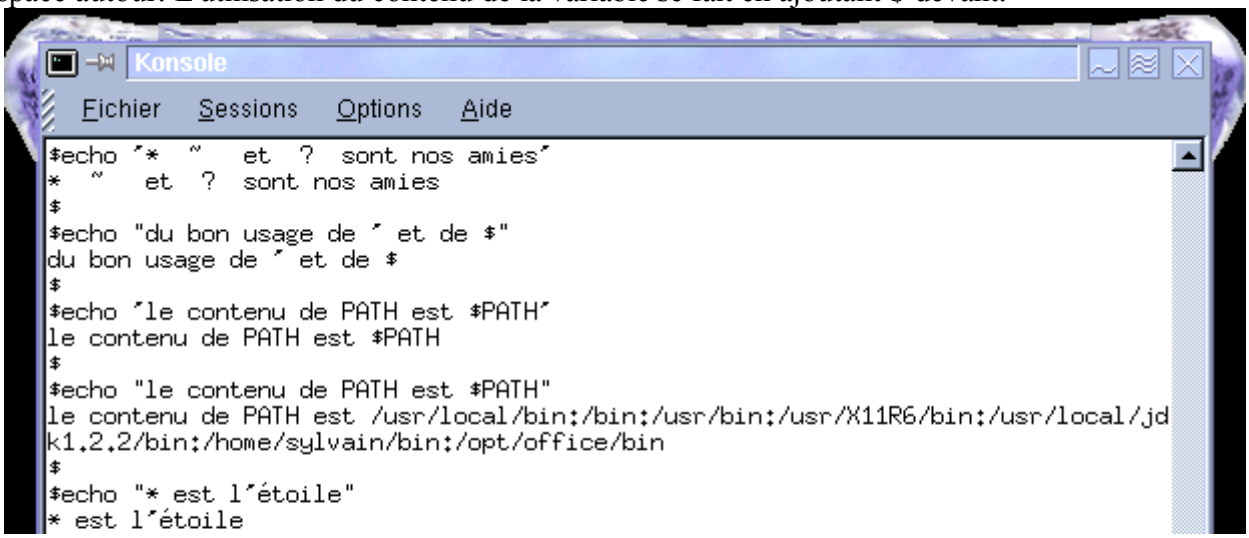
11.d - Les arguments

Comme tous programmes, les scripts acceptent des arguments. Pour récupérer et traiter le contenu de ces arguments dans vos scripts, vous utiliserez le signe $

$1 contient le premier argument, **$2** le second, et ainsi de suite, jusqu'à **9**.
$*** ou **$@ indique tous les arguments...
$0 indique le programme lui même...
$# renvoie le nombre d'arguments passés à la commande.

11.e - Les variables

Dans vos programmes, vous aurez parfois besoin de stocker des valeurs. Vous pourrez alors créer des variables (non typés). Une variable a un nom. L'affection se fait en utilisant l'opérateur = sans espace autour. L'utilisation du contenu de la variable se fait en ajoutant $ devant.



ce passionnant programme affiche son nom suivi de son argument. Remarquez comme le contenu des variables est perdu...

Attention cependant au notion de visibilité de la variable. Une variable n'est valable qu'à l'intérieur du shell ou elle est exécutée. Lorsque le sous shell rend la main, la variable est perdue (ou alors, il faut faire exécuter le shell par le shell courant, grâce à .).
Il est possible de protéger le nom de la variable grâce aux accolades

```
$ MOT=chat
$ echo le pluriel de $MOT est ${MOT}s
le pluriel de chat est chats
```

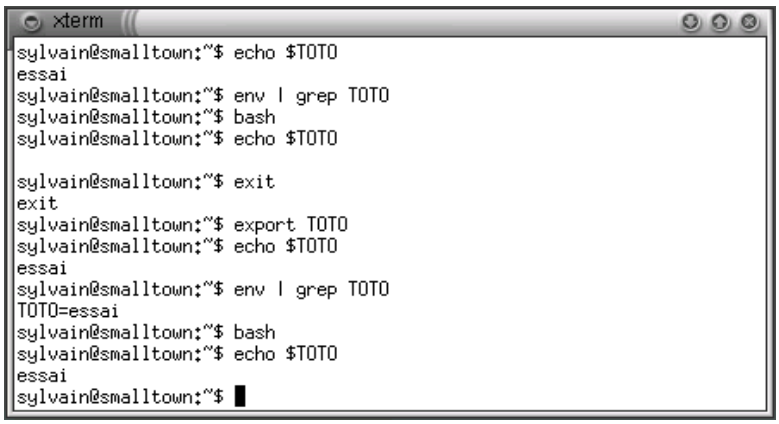
Si vous voulez affecter une chaîne de plusieurs mots à une variable, protégez la avec des guillemets.

```
NOM="Cherrier Sylvain Claude Philippe"

echo $NOM

Cherrier Sylvain Claude Philippe
```

11.e.1) la commande *export*.

Autre problème, la visibilité des variables... Comme pour d'autres langages de programmation, les variables ont une certaine portée, et ne sont utilisables que selon certaines règles. En général, une variable est utilisable dans le shell courant... Elle ne le sera pas pour un processus fils... Si on veut la rendre accessible à tous les fils, il faut l'exporter...

Exemple de portée d'un variable $TOTO... Si elle est exportée, elle est accessible dans les sous shells...



11.e.2) La commande *set*

Lors de la conception d'un programme il est très utile de voir s'afficher les instructions exécutées. La commande ***set***) qui est interne au shell (built in command) permet de réaliser cela.

Syntaxe

set [-efnvx [arg..**]]** `(Il existe d'autre options à la commande set)`

- ***set -e*** sort dès que la première commande retourne le statut faux (il faut être en mode non-interactif).
- ***set -f*** permet d'ignorer les méta-caractères
- ***set -n*** lit les commandes mais ne les exécute pas
- ***set -v*** affiche les lignes shell telles qu'elles sont lues, avant de les exécuter.
- ***set -x*** affiche les commandes et leurs arguments telles qu'elles sont exécutées (elles ont déjà été lues et interprétées).

Exemples

```
$ cat programme
#!/bin/bash
wc $1
pwd

$ sh -x programme fic1
sh -x programme fic1
+ sh -x programme fic1
+ wc fic1
12 432 235 fic1
+ pwd
/user/fred/src/html/

$ sh -n programme fic1
sh -n programme fic1
+ sh -n programme fic1
```

L'option -n n'a pas beaucoup d'intérêt ici car le programme n'est pas exécuté, or c'est à l'intérieur de celui-ci que se trouvent les commandes pertinentes.

Remarques

- Pour annuler toutes les options de la commande **set**, tapez : set -
- Toutes les options données à l'invocation du shell ($ set -x par exemple) sont conservées dans la variable **$-**

```
$echo $-
imxH
```

- set peut aussi servir à l'intérieur de script pour remplir les variables 1 2 3 etc. Solution parfois très pratique pour gérer ensuite le découpage des résultats d'une commande.

```
$ set `date` ; echo "nous sommes le $3  $2"
nous sommes le 20 fév
```

11.f - La commande read

Cette commande permet la saisie au clavier (ou plutôt l'entrée standard) pour le renseignement des variable. La commande read lit la ligne et remplit (selon le séparateur) toutes les variables. Si le nombre de mot est supérieur au nombre de variable, c'est la dernière variable qui contient toute la fin de la ligne. Si il y a plus de variable que de réponses, les variables en trop ne sont pas définies...

```
$ echo "nom prénom" ; read nom prénom
nom prénom CHERRIER Sylvain
$ echo $prenom $nom
Sylvain CHERRIER
$ echo "nom prénom" ; read nom prénom
nom prénom Baron SALIERE Alphonse Edouard
$ echo $prenom ; echo $nom
SALIERE Alphonse Edouard
Baron
```

11.g - La commande test :

11.h - Syntaxe

```
test expression
ou
[ expression ]
```

La commande **test** évalue *expression* , et retourne le résultat de cette expression (vrai ou faux).
Si aucune expression n'est donnée alors le résultat retourné prend la valeur faux.
Dans le cas de l'utilisation avec les crochets , il faut bien penser à mettre des espaces entre *expression* et les crochets [et].

11.h.1) Les tests sur les nombres

[i **-eq** j]	: est vrai si i est égal à j (i et j sont des entiers)	equal
[i **-ne** j]	: est vrai si i est différent de j	never
[i **-gt** j]	: est vrai si i est supérieur à j	Greater than
[i **-lt** j]	: est vrai si i est inférieur à j	Less than
[i **-ge** j]	: est vrai si i est supérieur ou égal à j	Greater or equal
[i **-le** j]	: est vrai si i est inférieur ou égal à j	Less or equal

11.h.2) Les tests sur les chaînes

test chaîne	vrai si chaîne n'est pas la chaîne nulle
test **-z** chaîne	vrai si chaîne est vide (0 caractère)
test chaîne1 = chaîne2	vrai si chaîne1 et chaîne2 sont identiques
test chaîne1 **!**= chaîne2	vrai si chaîne1 et chaîne2 sont différentes

11.h.3) Les tests sur les fichiers

test **-w, -r, -x***fichier*	vrai si *fichier* existe et est autorisé en écriture, lecture ou est exécutable (respectivement)
test **-d** *fichier*	vrai si *fichier* existe et est un répertoire
test **-f** *fichier*	vrai si *fichier* existe et n'est pas un répertoire
test **-s** *fichier*	vrai si *fichier* existe et a une taille non nulle

11.h.4) Exemples de tests

```
  Un test peut être effectué sur les variables ou sur un fichier.
  Le résultat (vrai-faux;0 ou 1),est généralement utilisé pour
  prendre une décision à l'intérieur d'une structure de contrôle.
  (if test then ....)


  TESTS SUR LES VARIABLES
  Exemples de tests:
1)La variable 3 contient-elle quelque chose?
    test -n $3
2)La variable LOGNAME est-elle égale à "bernard"?
    test $LOGNAME="bernard"

LOGNAME est une variable d'environnement ,il faut la taper en majuscule.
Idem pour les variables HOME, PATH, TERM
3)$? donne le code retour de la dernière commande. Si le résultat donne 0
alors la commande s'est déroulée sans erreurs.Sinon le résultat est 1.
Pour connaître la valeur du code, il faut entrer en ligne de commande:
    echo $?

L'exemple suivant test si le répertoire /usr existe. Echo affiche le
code de retour./usr existe alors il affiche 0.
    test -d /usr/;echo $?
    0


4)L'exemple suivant test si le nombre d'argument ($#)n'est pas
   égale à 1.Si le test est vrai alors il affiche la syntaxe usage et
   sort du script avec exit.

    if [ $# != 1 ];then
    echo "usage:$0 \"nom_utilisateur\""
    exit
```

```
fi
```

11.i - Les structures de contrôle

11.i.1) Définition

Les structures de contrôles d'un Shell sont des commandes internes qui permettent d'imbriquer plusieurs commandes et fabriquer ainsi un bloc. Ce bloc est vu ensuite comme une simple commande.
On pourra ainsi rediriger le résultat d'un bloc vers une sortie quelconque comme on le fait déjà avec une commande unique.
Les structures de contrôle, comme leur nom l'indique, permettent d'effectuer des tests (ou contrôles) sur les arguments passés et réagissent en fonction de ces test.

11.i.2) La structure for

La structure de contrôle for est utilisée pour exécuter des commandes avec une liste d'arguments.

Syntaxe

```
for var [ in liste ]
do
      commande1
      commande2
 ...
done
```

où *var* est le nom d'une variable et *liste* est la liste des valeurs que prendra cette variable.

Exemple

```
ls *.html
essai.html   intro.html   fin.html
cat prog
#!/bin/bash
for nom in *.html
do
       cp $nom /user/asi98/$1/src/
       echo "fichier $nom copié dans $1"
done
$ prog julien
fichier essai.html copié dans julien
fichier intro.html copié dans julien
fichier fin.html copié dans julien
```

Remarque

Si la liste n'est pas indiquée, alors c'est la liste des éventuels arguments du programme qui sera évaluée.

11.i.3) La structure if

Syntaxe

```
if (test1)
then
     commande1
[ elif (test2)
   then
         commande2 ]
  [ else
         commande3 ]
fi
```

où *test1* et *test2* sont des commandes que le shell évalue avant de faire son choix.
(Voir le chapitre sur les tests)

Exemple

```
if find /etc/ -name toto
then
   echo "il y a bien un fichier toto dans le
         répertoire /etc/"
else
   echo "il n'y a pas de fichier toto dans le
         répertoire /etc/"
fi
```

si le résultat de la commande find est vrai (code retour 0) alors la première ligne est affichée, sinon c'est la deuxième ligne qui est affichée.

11.i.4) La structure case

Syntaxe

```
case variable in
    pattern1 ) commande1 ;;
    pattern2 ) commande2 ;;
esac
```

où *pattern1 et pattern2* sont des modèles. Le Shell vérifie si la valeur de la variable "rentre" dans le modèle, et si oui, exécute la commande. Remarquez la fin de définition des pattern par le signe ;;. Il indique le changement de pattern (de cas...)

Exemple

```
case $choix in
    1 )  echo "Vous avez choisi l'option 1"
         echo 'vous avez bien fait';;
    2 )  rm -Rf *
         ;;
    9 )  echo "Cette option est invalidée";;
esac
```

11.i.5) La structure while

Syntaxe

```
while commande
do
    commande2
    commande3
done
```

Tant que *commande* est bien exécutée (code retour égal à 0) alors *commande2* et *commande3* sont exécutées.

```
while [ "$reponse" != "36412" ]
do
     echo "Entrez le chiffre magique"
     read reponse
done
echo "tricheur..."
```

11.i.6) La structure until

Syntaxe

```
until commande
do
  commande2
  commande3
done
```

La structure until a la même syntaxe que la boucle while, mais les commandes sont exécutées jusqu'à ce que lorsque le test soit vérifié.

```
until [ "$reponse" = "36412" ]
do
     echo "Entrez le chiffre magique"
     read reponse
done
echo "tricheur..."
```

(On aurait pu utiliser la boucle while :
tant que test n'est pas vérifié <=> jusqu'à ce que test soit vérifié
while [! test] <=> until [test]
)

11.i.7) La commande exit

exit [n**]**
Cette commande permet de sortir du Shell avec un code retour égal à l'entier n spécifié.
Si *[n]* est omis, le code retour de la sortie est celui de la dernière commande.

11.i.8) La commande break

break [n**]**
Cette commande arrête le tour de la boucle dans laquelle elle est insérée et sort de la boucle.
Dans le cas où on est en présence de plusieurs blocs imbriqués, le nombre **n** spécifie le nombre de

blocs desquels on veut sortir. Par défaut, on sort du bloc courant.

11.i.9) La commande continue

continue [n**]**
Cette commande permet de reprendre au début de l'itération suivante.
Si n est précisé c'est pour indiquer dans quel bloc imbriqué on souhaite reprendre l'itération.

11.i.10) La commande expr

Syntaxe

expr *argument*
Dans *argument* il peut y avoir des opérations arithmétiques ou des évaluations entre variables. La commande test récupère le résultat de ces opérations ou évaluations.

Exemples

```
Perimetre = ` expr $L + $l + $L + $l`
echo " le perimetre du rectangle est $perimetre"
```

Remarque

Puisque certains caractères utilisés dans les opérateurs arithmétiques ou logiques sont des caractères reconnus par le Shell, il faudra penser à les inhiber à l'aide du caractère \

```
surface = ` expr $pi \* $R \* $R `
echo "la surface d'un cercle de rayon $R est
$surface"
```

11.i.11) La commande . (point)

. executable
Le fichier exécutable est lu et ses commandes sont exécutées dans le shell courant. L'environnement du shell courant peut être modifié.

```
$ cat prog
echo "changement de répertoire"
cd /usr/bin/
$ . prog
changement de répertoire
$ pwd
/usr/bin/
```

11.i.12) La commande : (2 points)

C'est la commande qui ne fait rien et qui renvoie un code retour égale à 0. Très élégant pour écrire une boucle infinie...

```
while :
do
      commandes
done
```

11.i.13) La commande exec

exec [commande]
La commande *commande* est exécutée à la place du Shell courant (il n'y a pas de création de processus). Cela peut provoquer des sorties non désirées car lorsque *commande* est terminée, le processus (le Shell) est arrêté.
Une autre option possible d'exec est la redirection de toutes les sorties ou/et des entrées dans un fichier

```
exec > /tmp/fic
ls
...
exec < /tmp/fic2
read début fin
.....
exec
```

ici, le résultat des commandes est redirigé une fois pour toutes vers fic1, et le script lit les informations à partir du fichier fic2. A l'appel d'exec seul, il rebranche tout en standard.

Voir aussi les tubes nommés...

11.i.14) les redirections

Nous connaissons déjà les redirections (> >> < et <<), et nous avons juste au dessus l'utilisation de exec pour tout rediriger.

Il existe une solution assez élégant de redirection pour l'affichage d'un ensemble d'informations (par exemple dans le cas de procédure d'install...). Il s'agit du double redirecteur d'entrée...

```
#!/bin/bash
cat <<FIN
ce script vous permet d'afficher
l'ensemble   des   infos   que   vous
souhai..
...
(..) et tout ça.
FIN
ls \l
who
```

Dans cet exemple, on évite la multiplication des commandes echo... Tout ce qui est écrit est recopié sur la sortie standard, jusqu'à l'arrivée du mot clef (ici FIN)

11.i.15) la commande shift

cette commande permet de décaler le contenu des variables $0 1 2 et ainsi de suite... Si les variables $1 et $2 contiennent respectivement début et fin, $1 contiendra fin après l'utilisation de shift (et $2 sera non définie...)

Cette commande est utilisée dans des boucles, ou avec la commande set

11.i.16) la commande set

cette commande permet de remplacer le contenu des variables d'arguments ($0 $1 $2...) par le résultat d'une commande. Cela permet d'éviter de découper les lignes avec cut.... (ce qui est parfois laborieux...)

```
#!/bin/bash
set `who am i`
echo $5 $1
```

Ce script donne le temps depuis lequel vous êtes connectés, et le nom de log...

11.i.17) le remplacement de variables

On a souvent besoin dans un script interactif (type installation) de demander à l'utilisateur ses choix. Il est souvent lourd de devoir traiter tous les cas de figure (réponse pas donnée, incorrecte...).

Pour traiter ces cas automatiquement en diminuant les tests, il est possible d'utiliser le remplacement de variables. Ce remplacement est une option de ${var} (que nous avons déjà vu dans les variables).

${var:-argument}	permet de régler une valeur par défaut : si la variable est vide, elle vaudra argument
${var:=argument}	comme au dessus, mais créé en plus la variable si elle n'existait pas.
${#var}	donne la longueur de la chaîne
${var%chaine}	Extrait le début, jusqu'au dernier "chaine" Par exemple, si var vaut /usr/local/bin/local et chaine vaut local*, on obtiendra /usr/local/bin/
${var%%chaine}	Extrait le début, jusqu'au premier "chaine" Par exemple, si var vaut /usr/local/bin/local et chaine vaut local*, on obtiendra /usr/
${var#chaine}	Même système, mais en partant de la fin (avec l'option ## aussi)

Petite variante :

$(($var + 2))	Solution plus élégante au expr... il donne le même résultat

11.i.18) La valeur de retour

Afin de contrôler la bonne exécution d'une commande, il est possible d'utiliser le code de retour. Il s'agit d'un indicateur qui est habituellement positionné à 0 en cas de succès et prend une autre valeur sinon. Vous verrez qu'il est recommandé aux programmeurs de créer des programmes qui renvoient toujours une valeur de retour (pour les initiés, en C, le main n'est plus un void mais un int : il passe une valeur de retour au shell qui l'a invoqué...).
On peut visualiser la valeur du code de retour, qui est stocké dans la variable $?. On peut aussi directement tester la bonne exécution de programmes qui respectent cette norme

```
cp /tmp/toto .
echo $?
```

```
if cp /tmp/toto .
then
      Echo 'tout est ok'
else
      echo 'un pépin'
fi
```

11.i.19) Les sous programmes

on peut écrire des petites fonctions, dans un fichier à part (que l'on pourra lancer alors à la demande) ou plus élégant, à l'intérieur du script lui même (on appelle alors ce sous programme une fonction...).

On déclare la fonction avant son utilisation

```
exemple () {
      instruction 1
      instruction 2
      instruction 3
      ...
      return x
}
```

pour renvoyer une valeur, vous devez utiliser return (la où on utilisait un exit pour un script...). On pourra donc tester la valeur de retour ($?) ou faire des if de la fonction.... Il est possible de passer des arguments à la fonction, de la même façon que pour un shell (exemple arg1 arg2 arg3). La récupération dans la fonction se fera aussi par $1, $2 etc.

11.i.20) les astuces

renommons tous les fichiers .gif en .old.img

```
for image in *.gif
do
      mv $image ${image%%gif}old.img
done
```

bouclons sur les arguments

```
while [ $# -gt 0 ]
do
      echo $1
      shift
done
```

créons un fichier temporaire unique (constitué donc du numéro du Pid du shell

```
nomfic=/tmp/fichier$$
touch $nomfic
```

12) INIT, ou comment tout démarre

Après exécution du POST (Pré Operating System Test, test du bios), et l'affichage du bilan du setup, un PC lit (grâce aux routines fournies par le BIOS) le MBR du disque dur afin d'y trouver un exécutable. Si il en trouve un, il l'exécute (bonjour les virus !!!), sinon, il lit la table des partitions (placée à la fin du MBR) pour chercher la partition active. Il saute alors à l'endroit indiqué, et exécute le contenu du Boot Sector de cette partition. Si c'est un OS, alors celui ci est lancé...

Les loaders (comme lilo, Ntloader, System commander) permettent l'affichage d'un menu pour choisir parmi différents OS.

Lors du boot de Linux, le noyau est chargé en mémoire et son exécution commence... (voir plus loin la compilation du noyau). Dans le noyau est indiqué la rdev, c'est à dire la device root, ou la racine. Le noyau sait donc quelle partition il doit monté sur /. Ce système permet le démarrage de la machine avec une disquette[2], suivi du lancement de tout le système de démarrage du disque dur !!! (vous bootez sur /dev/fd0, et le rdev du noyau est par exemple /dev/hda2). Ce système peut d'ailleurs être surchargé en indiquant root=/dev/xxxx juste derrière le nom de l'image au prompt de lilo (par exemple linux root=/dev/hda6)

Bref, une fois le noyau lancé, et le root connu, le programme init va être lancé. C'est le père de tous les processus. Il va lancer à son tour toutes les autres taches de démarrage, puis de connexion.

Init est paramétré par le fichier /etc/inittab. Il existe plusieurs mode de lancement d'init

0 pour l'arrêt
1 pour single, (mode mono utilisateur réservé aux réparations), maintenant niveau 1...
de 2 à 5, différentes options (sans réseau, en graphique, etc).
6 pour le reboot

Le mode de démarrage par défaut est souvent 3 (multi utilisateur avec réseau en texte) ou 5 (même chose, en mode graphique). Ceci est définit dans inittab, à l'option initdefault.

Analysons un démarrage de RedHat ou Mandrake :

```
# inittab  This file describes how the INIT process should set
up
# the system in a certain run-level.
#   Author:                         Miquel   van   Smoorenburg,
<miquels@drinkel.nl.mugnet.org>
# Modified for RHS Linux by Marc Ewing and Donnie Barnes

# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have
networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
id:3:initdefault:
```

Jusqu'ici, le fichier décrit les différents modes, puis indique que le mode par défaut est 3

```
# System initialization.
```

2 Voir la commande dd pour la création d'une disquette de boot

```
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

Init lancera ensuite l'exécution du script /etc/rc.d/rc.sysinit. Ce script n'est utilisé qu'une fois au démarrage de la machine (n'oubliez pas qu'un serveur Linux peut tourner très très longtemps sans s'arrêter ;-))

Ensuite, selon le niveau d'exécution, le script /etc/rc.d/rc sera lancé avec un argument (le runlevel)

le script rc descendra dans un sous répertoire (/etc/rc.d/rc***RUNLEVEL***.d) et exécutera l'ensemble des scripts qui s'y trouvent (voir /etc/rc.d/rc)

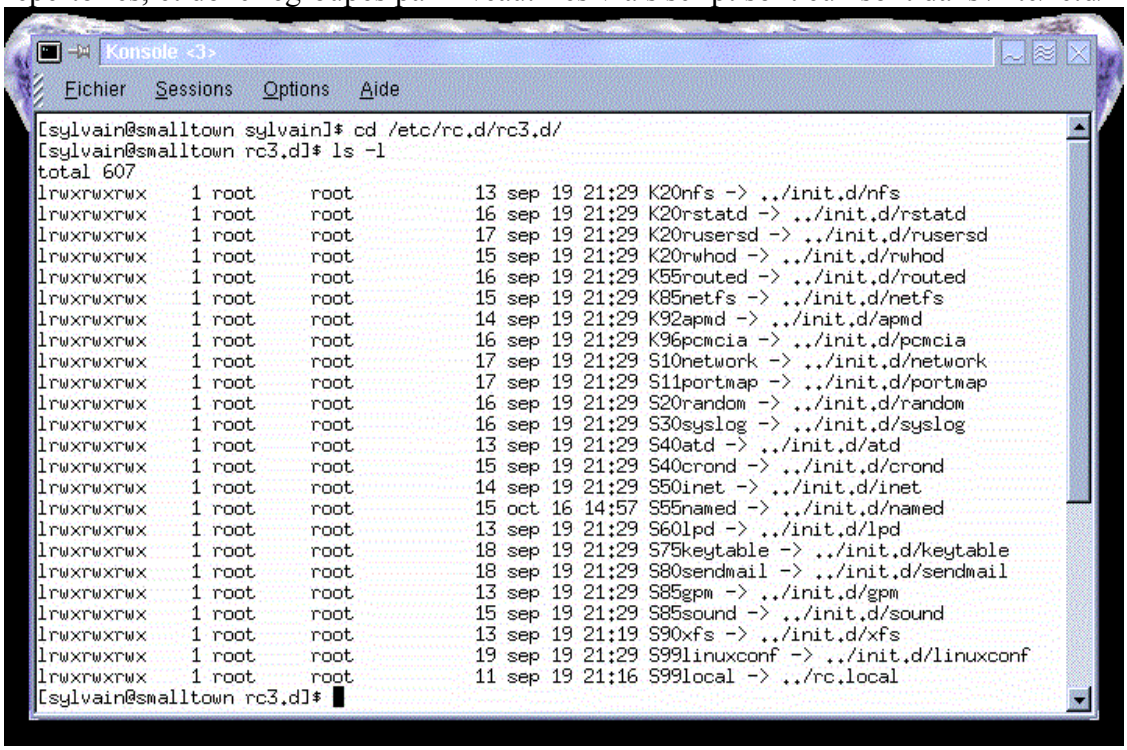
```
# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few
minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and
your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System
Shutting Down"

# If power was restored before the shutdown kicked in, cancel
it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown
Cancelled"
```

Puis description du TRAP du "three finger salute", et explication des comportements à suivre selon les différents signaux UPS (Onduleur et batterie de secours...)

```
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
Enfin, le lancement des différents écrans virtuels (tty1 à tty6)
en mode respawn (qui renait de ses cendres, en cas d'arrêt...)
```

En résumé, l'ensemble des informations de démarrage est configuré dans /Etc/inittab, puis ensuite dans /etc/rc.d. Toute l'astuce, c'est que l'ensemble des scripts à exécuter sont placés dans des sous

répertoires, et donc regroupés par niveau. Les vrais script sont eux sont dans /Etc/rc.d/init.d.



Dans chacun des /etc/rc.d/rc***RUNLEVEL***.d, répertoires qui seront visité par le programme /etc/rc.d/rc, on trouvera des liens symboliques vers les vrais scripts (dans /etc/rc.d/init.d). On pourra ainsi lancer apache lors d'un démarrage en mode 3, mais pas en mode 5, selon nos caprices... Les vrais scripts (ceux dans /etc/rc.d/init.d) acceptent souvent des arguments du type stop, start, reload, restart, ce qui permet toutes les variations possibles. D'où la convention de nommage des liens dans rc***RUNLEVEL***.d : S***numéro*** ou K***numéro***. S pour START, K pour KILL et le numéro pour définir l'ordre dans lequel les services sont démarrés.

Ce mode de démarrage est celui du System V (l'UNIX ATT). Il existe un utilitaire graphique qui permet la gestion des script (Editeur de System V Init)

le gestionnaire graphique des services lancés dans chaque runlevel. Simple drag'n drop de la zone service vers le runlevel concerné

Lors de l'installation d'un nouveau service, celui devrait mettre ses scripts de lancement et d'arrêt dans /etc/rc.d/init.d. Vous pouvez donc lancer et arrêter ce service à la main, en invoquant simplement

```
/etc/rc.d/init.d/monservice  start
/etc/rc.d/init.d/monservice  restart
/etc/rc.d/init.d/monservice  stop
```

Si tout fonctionne comme prévu, il vous suffira de configurer les différents runlevel pour y ajouter ce nouveau service.

12.a - Résumé du lancement:

- boot du noyau
- montage de /
- Lancement de init, avec le initdefault
- lancement de rc.sysinit
- lancement de tous les scripts Sxxxxx du répertoire /etc/rc.d/rc***RUNLEVEL***.d
- lancement de rc.local
- lancement des terminaux virtuels...

13) La compilation du noyau

Une fois votre Linux installé, il est tentant de parfaire le système, soit pour configurer des périphériques récalcitrants sous Linux (n'oubliez pas que peu de fabricants propose des drivers : il faut donc que des bénévoles en écrivent, le plus souvent sans documentation...), soit pour l'optimisation du système (Comme tout système clef en main, le noyau fourni est passe partout : compilé pour un 486 par exemple, il inclut énormément de drivers inutiles sur votre machine).

Le principe du logiciel libre, qui offre ces sources, va vous permettre de recréer votre propre Linux, adapté parfaitement à votre machine. Pour recompiler le noyau, encore faut il disposer des sources en C. Celles ci sont rangées dans /usr/src/linux. Vous remarquerez qu'il s'agit d'un lien symbolique sur un autre répertoire, dont le nom indique clairement la version qu'il contient. A l'écriture du support, il s'agit du noyau 2.2.14.

Lors qu'une nouvelle version sort, il suffit (presque ;-)) de la télécharger, de l'installer, puis de lier symboliquement ce nouveau répertoire à /usr/src/linux. Il va vous falloir aussi les librairies, et le compilateur C (gcc). Voyez l'installation pour plus d'informations...

La création d'un noyau Linux est découpé selon les étapes suivantes :

1. *configuration de la compilation*
2. *nettoyage, et création des dépendances*
3. *compilation du noyau*
4. *si besoin est, compilation des modules , et installation des modules*
5. *modification de lilo, afin qu'il pointe sur le nouveau noyau*
6. *reboot de la machine*

13.a - configuration

il s'agit de créer un fichier .config qui contiendra toutes les directives de compilation. Pour le créer, il y a trois méthodes : textes, ncurses (pseudo graphiques) et X (graphiques). Pour les deux dernières, il faut évidement que soit ncurses, soit X soit bien installés !!!

pour le lancement de la config., il faut taper soit **make config**, soit **make menuconfig,** soit **make Xconfig.**

Ce programme va vous demander les options à installer.

Notions importantes : Cet utilitaire vous demandera, dès qu'il le peut, si vous voulez compiler sous forme de modules : Ce système permet de diminuer la taille du noyau. Les drivers ainsi compilés peuvent être chargés et déchargés à la volée, sans arrêt de la machine. Il est ainsi possible de récupérer un nouveau driver, de le compiler et d'installer un nouveau périphérique sans arrêter l'OS (pour peu que la partie Hard l'accepte aussi ;-)... La configuration est découpée en chapitre, concernant par exemple la carte mère, les disques durs, les cartes réseaux, etc.

Si vous compilez en modules, il peut être intéressant de demander kerneld, un daemon qui lance ou qui arrête tout seul les modules, selon les besoins (sinon, il faut gérer les modules à la main, avec les commandes **insmod lsmod** ou **rmmod**). N'hésitez pas à interroger l'aide, à vous aider des documentations. Il vous faut évidement bien connaître la machine destinataire du noyau, afin de pouvoir répondre à certaines questions. Remarquez qu'il est possible de compiler un noyau pour une autre machine !!!

13.b - dépendances et nettoyage

La configuration terminée, vous pouvez ensuite taper les commandes suivantes :

make dep

Cet ordre demande la vérification des dépendances : l'ensemble des programmes et librairies visées sont elles présentes ?

make clean

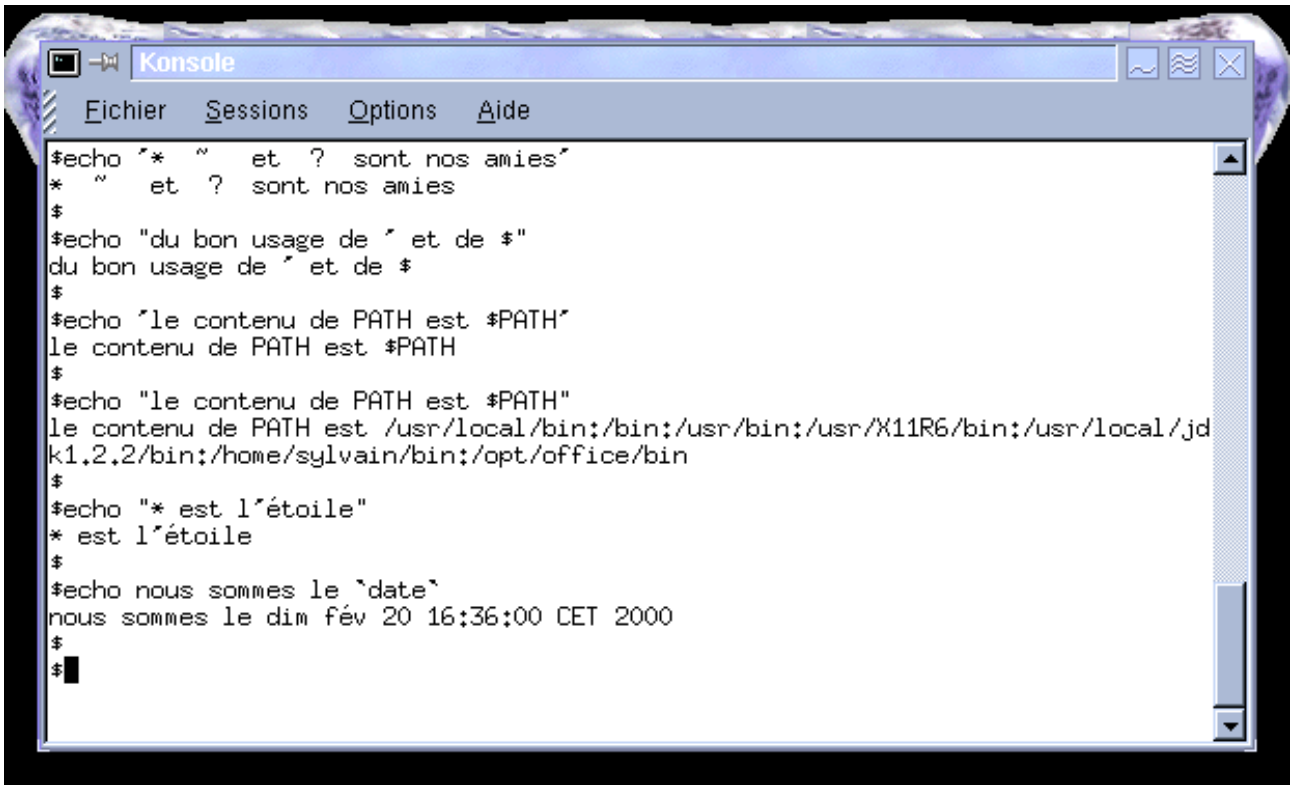
Cette commande efface les anciens programmes objets (résultats) qui traînent à cet endroit. Il s'agit du résultat des compils précédentes....

13.c - Compilation

Et c'est parti. Il faut maintenant se lancer. Pour compiler, il existe plusieurs options.

make bzImage est la commande standard.

Elle lance la compilation du noyau qui sera, si tout fonctionne bien, compressée selon la norme bzip2 (une évolution de la précédente, zip). Dans les versions précédentes, on utilisait donc make zImage. Le résultat (le noyau compressé) sera rangé dans /usr/src/linux/arch/i386/boot. Il s'appelle généralement bzImage. (du nom du make qui l'a invoqué).



Fin de compilation d'un noyau 2.2.13. La taille du kernel zipé est de 604 Koctets...

Si vous obtenez un tel message, vous avez réussi (on réussit très souvent, les seuls problèmes que je connaisse sont ceux d'un défaut à la récupération des sources, ou d'un problème de la machine. Très

rarement une mauvaise configuration....)

le fichier bzImage contient un binaire compressé, c'est le noyau Linux... Un tel système est déjà bootable, il suffirait de le copier sur une disquette (voir dd) pour démarrer la machine visée.

Pour finir proprement l'installation, passons aux modules.

13.d - Compilation et installation des modules.

Je vous rappelle que les modules sont des drivers qui ont la particularité d'être chargeable et déchargeable. On obtient donc un noyau plus petit (théorie du micro Kernel), autour duquel s'articulent les drivers. Le système est plus petit, plus fiable et plus sain. Les modules se gèrent avec les commandes **lsmod, insmod, rmmod** et **modprobe**. Le kernel peut s'occuper de la gestion des modules automatiquement grâce à kerneld.

Les commandes permettant la compilation et l'installation des modules sont :

make modules

make modules_install

Make modules lance la compilation de tous les modules, c'est à dire défile l'ensemble des sous répertoires de /usr/src/linux, et compile toutes les parties qui ont été indiquées modulaires.

Make module_install copie ensuite l'ensemble des .o résultants de la compilation des modules dans un répertoire nommé /lib/modules/NUMERO DE VERSION DU NOYAU.

Dernier point. Un fichier nommé /etc/conf.modules contient la liste de alias courants, et des drivers à lancer. (faites un man de modprobe)

13.e - Lancement du nouveau noyau

Linux est livré généralement avec un loader, nommé lilo (Linux Loader). C'est un programme qui génère un binaire qui s'installe soit dans la MBR, soit dans le boot sector de la partition active, et qui permet le lancement de plusieurs os. Une seconde partie de ce binaire est situé dans la partition Linux....

Le binaire généré par lilo dépend d'un fichier de configuration, nommé /etc/lilo.conf. Ce fichier décrit les différentes options de démarrage de la machine.

```
boot=/dev/hda3
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz.ok
```

```
      label=linux
      root=/dev/hda3
      read-only
image=/boot/vmlinuz-2.2.13-7mdk
      label=linux.old
      root=/dev/hda3
      read-only
other=/dev/hda1
      label=dos
      table=/dev/hda
```

Ce lilo.conf crée un binaire qui s'installera dans le boot sector de la 3 partition (hda3). Map indique ou est la suite du lilo. Install quant à lui indique le nom d'un fichier, copie du secteur modifié avant sa modification. Time out force le lancement de la première image si l'utilisateur n'est pas intervenu avant xx dixièmes de seconde.

Nous avons ensuite trois possibilités de démarrage : 2 Linux différents (bien que sur la même partition), et Windows. En ce qui concerne Linux,

- le mot clef image pointe sur le kernel (celui que nous avons crée tout à l'heure!!!).
- Label indique le nom qui sera proposé à l'utilisateur.
- Root indique quelle partition deviendra / (au boot, le kernel démarre, puis il faut lancer tous les scripts de démarrage : le kernel les trouvera sur la partition indiquée après root)
- Enfin read-only indique que nous désirons aller chercher le noyau sur un file system en lecture seule. Après que le système ait démarré, et divers taches de vérification terminées, le file système sera remonté en lecture écriture.

Pour les autres OS, indiquez avec le mot clef other la partition qui le contient, et donnez lui un label (un nom proposé au démarrage). Table indique simplement ou est la table des partitions (certains os en ont besoin...)

Pourquoi deux linux?

Simple précaution. Nous venons de compiler un nouveau noyau. Il est particulièrement optimisé, et ne contient que les drivers nécessaire. Que se passerait-il si nous avions fait une erreur ? Le système pourrait ne plus démarrer. Voila pourquoi, par tradition, on conserve souvent une ancienne version qui, si elle n'est pas des plus efficaces, à le mérite de fonctionner. Un petit linux.old au moment ou lilo nous demande notre choix nous permettra de faire fonctionner la machine, pour réparer....

L'opération de mise à jour consiste à copier le binaire kernel (le bzImage de tout à l'heure) soit dans la racine, soit dans le répertoire /boot (selon votre choix, et la distribution installée). Renommez le (mv bzImage vmlinuz.new) par exemple.

Vous pouvez ensuite modifier votre fichier de configuration avec votre éditeur favori. La création des deux fichiers résultants (le boot sector et le fichier map) se fera lors du lancement de la commande :

lilo

Cette commande lit le fichier lilo.conf, et si il est cohérent, génère les fichiers voulus.

Il ne reste plus qu'a rebooter, et à prier pour que cela redémarre...

13.f - Cas particulier du patch

dans un but d'évolution (pour avoir la dernière dernière version drôlement plus mieux !!!), il y a deux solutions.

Télécharger l'ensemble de la nouvelle version (en 60 et 80 Mo non compressé !!)

Télécharger les patchs (différentiels des versions) très petits, mais attention à bien respecter l'ordre des mises à jour....

La procédure d'application du patch est la suivante. Décompressez le patch si il l'est... Déplacez le dans /usr/src

tapez la commande suivante

patch -p0 < nomdupatch

Si il y a un problème, vous aurez des messages d'erreurs. Sinon, vous pouvez lancer la procédure de configuration et de compilation...

14) INETD, le super serveur

Linux permet la fabrication de serveurs très puissants sur des machines modestes. On peut rapidement arriver à offrir une multitude de services. Afin d'alléger la charge de travail, certains services qui ne sont pas trop demandés peut être invoqués à la volée, afin de ne pas multiplier les processus dormants.

Inetd permet de centraliser les demandes de services. C'est lui qui intercepte la demande, et après vérification, lance le demon correspondant, et le stoppe une fois la tache terminée. Dans inetd, vous trouverez les services tels que telnet, ftp, apache, samba, talk, etc...

Il est parfois préférable de faire tourner un démon en 'standalone'. Dans ce cas, il sera lancé par un script de lancement (automatisé ou non), et ne devra pas être présent dans inetd !! En cas de problème, lancez des ***ps alx afin de bien vérifier les PID et les PPID, qui devraient être parlant !***

Autre avantage d'inetd : la sécurité ! Ce programme utilise un wrapper, appelé tcpwrapper. Son petit nom est tcpd (le démon tcp). Il scanne chaque requête, puis applique la sécurité donnée dans deux fichiers important

hosts.allow, et hosts.deny

Chacun de ces fichiers définit des cibles, qui indique un ou plusieurs services, et une ou plusieurs machines. Si la cible est dans le hosts.allow, l'accès est autorisé. Si la cible est dans le hosts.deny, l'accès est interdit. Si ce n'est ni l'un ni l'autre, l'accès est autorisé...

Résumons nous :

/etc/inetd.conf

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Lines starting with "#:LABEL:" or "#<off>#" should not
# be changed unless you know what you are doing!
#
# If you want to disable an entry so it isn't touched during
# package updates just comment it out with a single '#' character.
#
# Packages should modify this file by using update-inetd(8)
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#echo       stream      tcp   nowait      root  internal
#echo       dgram udp   wait  root  internal
#chargen    stream      tcp   nowait      root  internal
#chargen    dgram udp   wait  root  internal
discard          stream      tcp   nowait      root  internal
discard          dgram udp   wait  root  internal
daytime          stream      tcp   nowait      root  internal
#daytime    dgram udp   wait  root  internal
time       stream      tcp   nowait      root  internal
#time       dgram udp   wait  root  internal
```

```
#:STANDARD: These are standard services.
ftp         stream      tcp   nowait      root  /usr/sbin/tcpd
     /usr/sbin/wu-ftpd -l

#:BSD: Shell, login, exec and talk are BSD protocols.
talk        dgram udp   wait  nobody.tty  /usr/sbin/tcpd
     /usr/sbin/in.talkd
ntalk       dgram udp   wait  nobody.tty  /usr/sbin/tcpd
     /usr/sbin/in.ntalkd

#:MAIL: Mail, news and uucp services.
#disabled#smtp          stream      tcp   nowait      mail  /usr/sbin/exim
exim -bs

#:INFO: Info services

#:BOOT: Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers."

#:RPC: RPC based services

#:HAM-RADIO: amateur-radio services

#:OTHER: Other services
netbios-ssn       stream      tcp   nowait      root  /usr/sbin/tcpd
     /usr/sbin/smbd
netbios-ns dgram udp   wait  root  /usr/sbin/tcpd    /usr/sbin/nmbd -a
#<off># swat            stream      tcp   nowait.400  root  /usr/sbin/tcpd
     /usr/sbin/swat
```

Ce fichier contient la description de tous les services écoutés, et l'indication du serveur à invoquer

/etc/hosts.allow, et /etc/hosts.deny

```
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                                 See  the  manual  pages  hosts_access(5),
hosts_options(5)
#                    and /usr/doc/netbase/portmapper.txt.gz
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
# If you're going to protect the portmapper use the name "portmap" for
the
# daemon name. Remember that you can only use the keyword "ALL" and IP
# addresses (NOT host or domain names) for the portmapper. See portmap(8)
# and /usr/doc/netbase/portmapper.txt.gz for further information.
#
esound: 127.0.0.1
esound: 127.0.0.1
```

```
# /etc/hosts.deny:  list  of  hosts  that  are  _not_  allowed  to  access  the
system.
#                  See the manual pages hosts_access(5), hosts_options(5)
#                  and /usr/doc/netbase/portmapper.txt.gz
#
```

```
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "portmap" for
the
# daemon name. Remember that you can only use the keyword "ALL" and IP
# addresses (NOT host or domain names) for the portmapper. See portmap(8)
# and /usr/doc/netbase/portmapper.txt.gz for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
ALL: PARANOID
```

ces fichiers réglent la sécurité du wrapper, qui est consulté par inetd. Le système est permissif (ce qui n'est pas interdit est autorisé).

/etc/init.d/inetd , /etc/init.d/portmap

Ces deux scripts permettent le contrôle des deux démons. (start, stop, reload, et restart)

/var/log/syslog , /var/log/messages , ou autres...

Fichiers de logs contenant les informations sur les événements (accès, refus...)

15) xinetd, une évolution du super serveur...

Il s'agit d'une version étendue, qui permet un paramétrage plus fin, service par service. Ce serveur se configure grâce à un fichier /etc/xinetd.conf, qui contient en général une configuration générique (par exemple, le nom du fichier dans lequel seront envoyés les logs...) et le nom d'un include, c'est à dire le nom du répertoire qui contiendra toutes les définitions spécifiques (ici, /etc/xinetd.d)

```
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{
      instances               = 60
         log_type                = SYSLOG authpriv
         log_on_success       = HOST PID
         log_on_failure       = HOST
}

includedir /etc/xinetd.d
```

Ici, on voit que tous les évènements seront consigné dans la rubrique authpriv... La consultation du fichier /etc/syslog.conf nous indique que le fichier de log correspondant est /var/log/secure...

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                          /dev/c
onsole

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none       /var/l
og/messages

# The authpriv file has restricted access.
authpriv.*                                   /var/log/sec
ure

etc etc etc...
```

Donc, le serveur xinetd renseignera ce fichier (/var/log/secure) pour tout événement...

Le serveur xinetd va donc visiter le répertoire /etc/xinetd.d pour récupérer des informations... En général, on trouvera dans ce répertoire un fichier de configuration par service... Dans mon répertoire, je trouve un fichier swat et un fichier telnet... Ceux ci sont placés ici lors de l'installation du paquetage, et (sur la redhat en tout cas) sont inactifs... Une ligne disable=yes est inscrite en fin du fichier... Il vous suffit de la supprimer, et de relancer le serveur pour que cela fonctionne...

le serveur est contrôlé comme d'habitude par un petit shell à la mode System V, placé dans /Etc/init.d...Il vous suffit de l'invoquer (en tant que root) avec les arguments habituels (start, stop, restart ou reload)

/etc/init.d/xinetd restart

```
# default: off
# description: SWAT is the Samba Web Admin Tool. Use
swat \
```

```
#             to configure your Samba server. To use
SWAT, \
#             connect to port 901 with your favorite
web browser.
service swat
{
      port  = 901
      socket_type = stream
      wait  = no
      only_from = 192.168.1.2
      user  = root
      server      = /usr/sbin/swat
      log_on_failure    += USERID
}
```

```
# default: on
#  description:  The  telnet  server  serves  telnet
sessions; it uses \
#     unencrypted    username/password    pairs    for
authentication.
service telnet
{
      flags       = REUSE
      socket_type = stream
      wait        = no
      user        = root
      server            = /usr/sbin/in.telnetd
      log_on_failure    += USERID
}
```

Ces deux fichiers de configuration sont assez intéressants, car on peut assez facilement s'apercevoir d'une importante différence entre ces deux services... L'un, telnet, est un serveur standard du super serveur xinetd (anciennement inetd). C'est pour cette raison que le nom du programme est in.telnetd, le in symbolisant la dépendance à inet... En ce qui concerne swat, il s'agit d'un serveur fourni par les concepteurs de samba, et que l'on met en dépendance de xinet. Il pourrait tout à fait tourner tout seul (ce que l'on appelle autonome), mais dans ce cas, on peut imaginer qu'il consomme des ressources... Comme on imagine qu'il est assez rarement utilisé (on ne règle pas samba tous les jours !!!), on le met en dépendance du super serveur...

XINETD est un extension de inetd... Les règles habituelles sont toujours en vigueur, notamment les autorisations par hosts.allow et deny...

16) IPCHAINS et IPTABLES

Ces deux programmes sont des firewalls et des gestionnaires de paquets. Ils analysent les paquets qui arrivent, qui sortent ou qui transitent sur votre machine, et décident, grâce à des règles, de leur avenir. Un paquet peut être accepté, rejeté, transmis ("forwardé")...

En général, il faut déterminer les ports TCP et UDP qui sont ouverts, en donner la liste à la machine en indiquant qui peut et qui ne peut pas se connecter dessus. Il est possible aussi de filtre de l'icmp...

Sur une RedHat, ce firewall est actif à l'installation, et vous joue parfois des tours (Mais n'est il pas préférable d'avoir une machine trop protégée que pas assez !!!).

Les règles de sécurités sont stockées dans un fichier dans /etc/sysconfig/ (dans une RedHat)... Il s'appelle ipchains, et en général est directement généré par les scripts de firewall (/etc/init.d/ipchains save)...

Vous pouvez cependant l'éditer à la main si vous savez ou vous allez, le script de lancement contrôlant la validité de vos ajouts...

```
# Firewall configuration written by lokkit
#   Manual   customization   of   this   file   is   not
recommended.
# Note: ifup-post will punch the current nameservers
through the
#        firewall; such entries will *not* be listed
here.
:input ACCEPT
:forward ACCEPT
:output ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -p tcp -b -s 192.168.1.1/32 901 -d 0/0 -j
ACCEPT
-A input -p tcp -b -s 192.168.1.* 137:139 -d 0/0 -j
ACCEPT
-A input -p udp -b -s 192.168.1.* 137:139 -d 0/0 -j
ACCEPT
-A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT
-A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT
-A input -p udp -s 0/0 -d 0/0 2049 -j REJECT
-A input -p tcp -s 0/0 -d 0/0 6000:6009 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 7100 -y -j REJECT
```

Voici mon fichier ipchains... Les lignes en gras sont celles que j'ai ajouté pour permettre la connexion sur le service 901 (swat), et le bon fonctionnement de samba (port 137 à 139, en udp et tcp). Remarquez la notation adressIP/mask (192.168.1.1/32 veut dire la machine précisément, ici mon serveur), et 0/0 veut dire le monde entier... Quant à 192.168.1.*, cela signifie toutes les machines du réseau 192.168.1.0. En effet, **NETBIOS** utilise le broadcast pour diffuser de l'information (partages, recherche du **MASTER BROWSER**). Un réglage trop stricte du firewall risque de vous priver de « Voisinage Réseau »

Après modif, un petit coup de **/etc/init.d/ipchains restart...**

17) SAMBA, le serveur NetBios sous Linux

17.a - Présentation

Samba est une solution libre qui permet d'offrir des services programme standard client réseau (partage de fichiers et d'imprimantes) de Micro$oft, c'est à dire NetBios.

Le paquetage SaMBa est ainsi nommé car NetBios parle le protocole SMB (Server Message Block) pour l'échange d'information entre les différentes machines... Samba peut donc remplacer d'un simple Window 98 à un serveur NT... Il permet de partager des répertoires sur le serveur, d'offrir des imprimantes, un home directory pour chaque utilisateur, et aussi la validation du compte utilisateur sur le serveur... Le fait que Linux et Windows n'utilise pas le même filesystem ne gène pas, puisque le client envoie au serveur toutes les informations à stocker, et que c'est le serveur qui s'occupe de ce stockage, à sa manière à lui (je rappelle que de toutes façons un client Windows et un serveur NT ne parle pas non plus le même FileSystem...). Samba sait aussi se connecter sur des ressources Netbios (imprimantes, serveurs de domaine, répertoires partagés, etc).

Le package Samba fourni un ensemble de programme, clients et serveurs.

Smbd est le serveur proprement dit, qui exporte les ressources

nmbd est le serveur WINS, une sorte de DNS propriétaire MicroSoft. Il permet dynamiquement (à la différence du DNS) la résolution des noms (à la différence qu'il s'agit de résoudre nom NetBios et adresse IP). Il participe aussi aux élections et à la notion de Browser Master...

smbclient est un outil client qui permet de simuler le voisinage réseau, et la connexion à des ressources... Une sorte de commande NET en gros...

smbpasswd gère quand à lui les comptes des utilisateurs. Il contrôle le fichier /Etc/samba/smbpasswd, qui contient les noms d'utilisateurs, la correspondance avec le compte unix, et le mot de passe crypté... Ce programme permet la mise à jour des mots de passe.

Smbstatus donne des informations sur l'état actuel du serveur : combien de connexion, qui est connecté, depuis combien de temps, quelles sont les ressources utilisées, par qui....

testparm permet de tester la validité de votre fichier de configuration. Il fait un bilan des ressources offertes, et signale toute erreur de syntaxe.

Nmblookup est une sorte de nslookup, mais sur le serveur wins au lieu du dns...

swat est un serveur web, tournant sur le port 901, qui permet la gestion complète et à distance du serveur SAMBA...

Samba est contrôlé par un fichier de configuration, habituellement stocké dans le répertoire dévolu à SAMBA, /etc/samba. Il s'appelle smb.conf, est de type texte, et sa structure est calquée sur celle des fichiers .INI chers à Windows...

```
[global]
      log level = 1
      max log size = 1000
      socket options = TCP_NODELAY IPTOS_LOWDELAY
      guest ok = no
[homes]
```

```
      browseable = no
      map archive = yes
[printers]
      path = /usr/tmp
      guest ok = yes
      printable = yes
      min print space = 2000
[test]
      browseable = yes
      read only = yes
      guest ok = yes
      path = /export/samba/test
```

Ce fichier smb.conf exemple configure samba de la façon suivante : On y trouve plusieurs sections, dont [global], [homes], [printers] et [test]. Toutes ces sections correspondent à des partage, sauf global. Dans [global] sont stockés les informations générales, concernant le serveur : son nom, s'il est serveur de domaine, etc... Il contient aussi les réglages par défaut des partages (ici guest ok=no). Ces réglages par défaut pourront être outrepassés dans chacun des partages (voir [test] qui, lui, force guest ok=yes)

Chaque partage est ensuite défini, avec ses options (qui sont d'un nombre très important !!!). Par exemple read only (yes ou no) pour lecture seule, browsable (yes or no) pour indiquer si la ressource est annoncée dans la browse list, guest ok (yes or no) indique s'il faut s'identifier ou pas, guest only (yes or no) si c'est seulement comme le guest, guest account = 'un_nom_de_user' pour qu'Unix retrouve les droits à appliquer (ce sera ceux de cet utilisateur!!!) lors d'une connexion guest... Le fichier exemple ci-dessus donne donc trois partages : [homes] (les homes directory des utilisateurs, protégés par mot de passe), [printers] permet un accès aux imprimantes, et [test] est un partage fixe... Remarquez la commande path qui indique à quel répertoire ce partage correspond sur la machine Unix...

NOTA : Si vous n'êtes pas en ***security = share***, alors pensez à créer l'utilisateur (du guest account)... Celui ci doit exister dans votre sécurité Linux, et être ajouté à la sécurité samba (/etc/samb/smbpasswd)... Pour ce faire, utilisez ***smbpasswd -a user***

17.b - Un peu plus loin avec Samba

Voyons un peu maintenant des options plus intéressantes. Dans la section [globals], voici d'autres variables à documenter....

```
[global]
      #  Server configuration parameters
      netbios name = HYDRA
      server string = Samba %v on (%L)
      workgroup = SIMPLE
```

Les noms parlent d'eux même, remarquons quand même le server string (commentaire apparaissant dans Voisinage Réseau, à coté du nom Netbios) qui est constitué des variables %v (version) et%L (nom NETBIOS). Le workgourp est défini ici.

```
 #  Networking configuration options
      hosts allow = 192.168.220. 134.213. localhost
      hosts deny = 192.168.220.102
      interfaces = 192.168.220.100/255.255.255.0 \
                        134.213.233.110/255.255.255.0
      bind interfaces only = yes
```

Ici, divers réglages réseaux permettant d'interdire ou de permettre à des machines l'utilisation du

serveur (192.168.220.* et 134.213.*.* ont des accès). Interfaces permet de définir, en cas de doute, la liste des cartes réseaux que samba pourra utiliser... et surtout, d'interdire d'autres entrées avec l'option bind interface only.

```
# Debug logging information
     log level = 2
     log file = /var/log/samba.log.%m
     max log size = 50
     debug timestamp = yes
```

Ici, nous réglons le niveau et le type de log, c'est à dire les traces d'événements sur le serveur... log level = 2 est assez élevé (le normal est 1, et pas de log est 0). Le fichier de log est défini en dessous. Il est possible de logguer tout dans un seul fichier, ou de faire des fichiers différents. Ici, c'est l'option par machine (%m) qui a été choisi. On aura donc un fichier de log par machine connectée. On aurait pu jongler avec diverses variables %u (utilisateur), %I (adresse ip), voire même %a (par type d'OS).

```
#  Browsing election options
     os level = 34
     local master = yes
```

Si vous connaissez le principe des élections sur un réseau de type Netbios, vous comprenez pourquoi il est possible ici de contrôler nos nombre de voix... 33 Correspondant aux nombres de voix d'un serveur NT (qui vote pour lui), nous nous arrangeons pour le battre et devenir le browser master. De même Local master indique que nous voulons gagner ces élections (nous votons pour nous).

Observons maintenant les partages, qui sont tous les noms mis entre crochets, sauf [global]

```
[homes]
     browseable = no
     map archive = yes
[printers]
     path = /usr/tmp
     guest ok = yes
     printable = yes
     min print space = 2000
[test]
     browseable = yes
     read only = yes
     guest ok = yes
     path = /export/samba/test
```

En général, on trouve le nom du partage, et le répertoire Unix qui y correspond, grâce à la variable **path**... On voit ici que test est le nom de partage de */export/samba/test*. On notera que [**homes**] n'a pas de path, puisqu'il s'agit du home directory de l'utilisateur connecté, que la machine obtient en consultant le fichier */etc/passwd*. D'autres options sont intéressantes. **Guest ok =yes** indique que l'on peut se connecter à la ressource sans mot de passe, et ce avec les droits d'un utilisateur désigné, par la commande **guest account = xxx.** Il faudra bien vérifier les droits de cet utilisateur (et penser à le créer avec **smbpasswd -a user**, pour qu'il existe dans votre **/etc/samba/smbpasswd.** Si vous oubliez cette étape, vous ne pourrez pas accéder aux IPC). Browseable permet la diffusion du partage, c'est à dire que la browse list est avisée... (voisinage réseau, et net view). Il est donc possible de cacher une ressource, sans utiliser de $...
read only parle de lui même, et fait partie d'un ensemble de commandes comme **read list** qui permet de définir une liste d'utilisateurs qui auront accès, **write list** (même chose en écriture), **writeable** (en lecture seule ou non), **host allow** (liste de machines qui peuvent accèder au partage), **valid users**,

invalid users, etc...

17.c - A la mode NT

Il est possible de régler samba afin qu'il se comporte de plus en plus comme un serveur NT (c'est a dire qui centralise la sécurité des accès...)
Commençons par basculer le mode de sécurité : il en existe 4 :

security = share	La sécurité est vérifiée à chaque partage... Il s'agit d'un mode poste à poste...
Security = user	Ici, l'utilisateur doit être reconnu. Il doit donc exister au niveau Unix, et au niveau samba. La commande **smbpasswd -a username** permet d'ajouter cet utilisateur à la sécurité samba, en le liant à son compte unix. Les mot de passe peuvent être différent. Le programme smbd interrogera le fichier /etc/samba/smbpasswd, qui contient le nom, l'UID et le mot de passe de l'utilisateur
Security=server	On délègue à un autre serveur (NT) le rôle de validation de l'utilisateur...
Security = domain	On délègue à un domaine NT (donc un PDC ou des BDC) la sécurité...

L'option security=user est intéressante car elle permet une véritable gestion des users. Il existe ensuite un tas de nuance d'utilisation.
Personnellement, j'utilise samba de la façon suivante :
Il gagne les élections, devient le master browser, contrôle les partages, gère les users, et les valide.
Mes réglages sont les suivant (smb.conf)

```
[global]
      workgroup = TEST
      netbios name = SERVEUR
      server string = le serveur SaMBa
      os level = 65
      preferred master = yes
      domain master = yes

      domain logons = Yes
      encrypt passwords = Yes
      wins support = Yes
```

On voit
- la définition **netbios** (nom du WG, nom de la machine)
- le **niveau de l'OS** (65) et domain master et preferred master activé pour qu'il remporte les élections,
- **domain logons** oblige samba à valider la connexion de l'utilisateur à sa machine (il faut pour cela avoir activé '**Validation Domaine NT**" dans Client pour réseau microsoft)
- **encrypt passwords** pour les machines clientes à partir de Win95 OSR2 (qui s'est mis à crypter les mots de passe)
- **wins support** qui remplacera un serveur DNS, pour peu que vous pensiez aussi à déclarer votre machine Unix en tant que serveur Wins dans la config TCP du client Microsoft.

Dans ce type de configuration, vous êtes capable d'offrir un système équivalent à celui d'un serveur NT. Chaque ordinateur client affichera un écran de connexion, qui permettra de saisir le nom d'utilisateur, le mot de passe, et affichera le nom du domaine NT (en fait SaMBa) qui validera ce

compte. Un message est envoyé sur le réseau, et revient OK ou pas. Si tout est OK, on peut alors, si besoin est, exécuter un "logon script" (un petit script DOS). Ce script est celui indiqué par **logon script =** et peut utiliser des astuces telles que %U (pour le nom d'utilisateur, afin de le personnaliser !!!)

```
NET   TIME   \\SERVEUR   /SET   /YES
NET   USE   H:   /HOME
NET   USE   L:    \\SERVER\LOGICIELS
```

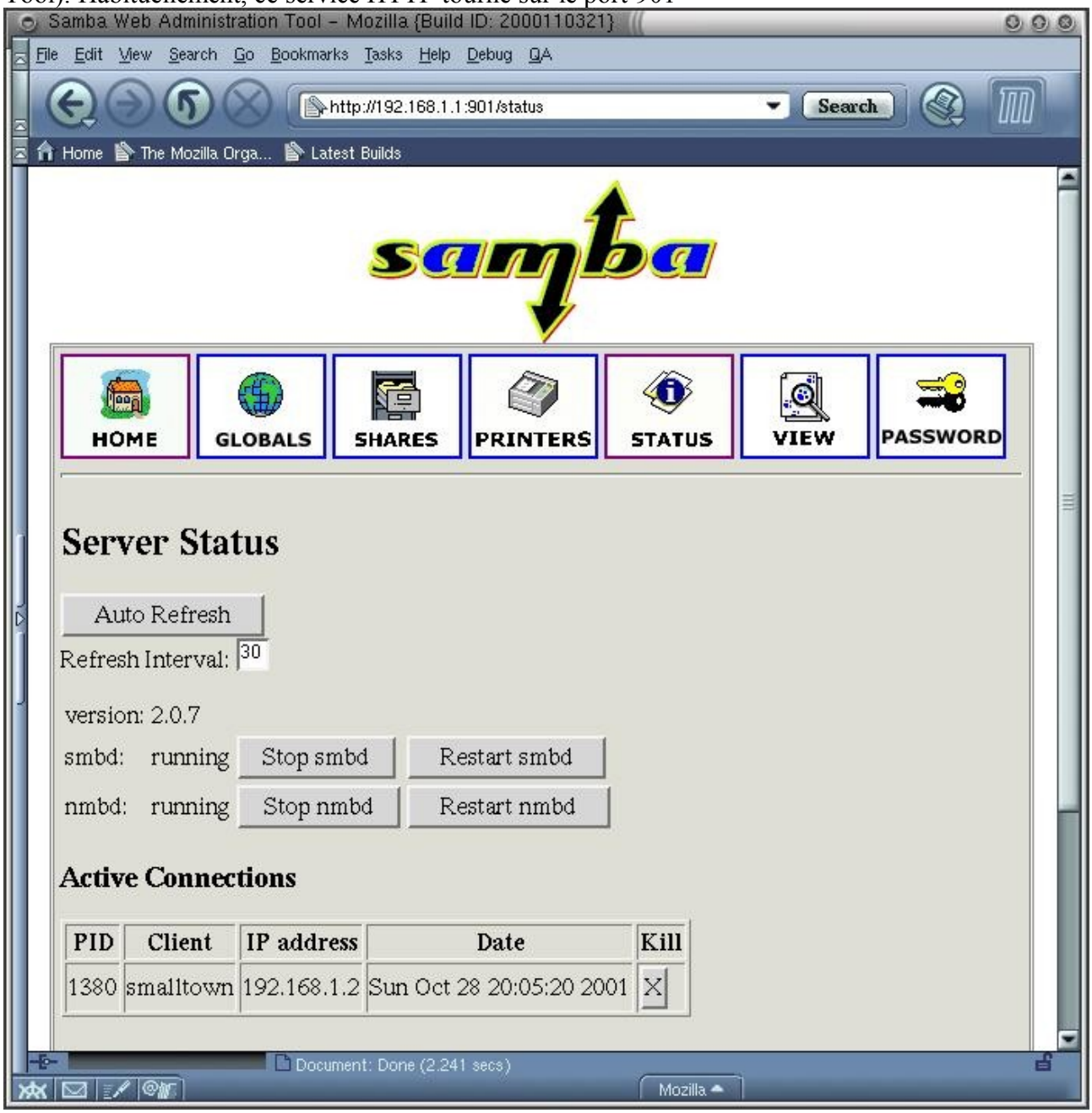
un exemple de script de logon, qui sera exécuté par chaque client, à chaque connexion.

NOTA : après quelques tests, on s'aperçoit que le home de chaque user est lisible en forçant la connexion (non browsable, mais on peut forcer la main avec un NET USE G: //nostromo/root). Afin d'éviter cela vous pouvez utiliser la variable globale %S qui symbolise le nom du user... Et entre les directives suivantes :

```
[homes]
        com[homes]
        comment = Home Directoy de %S
        valid users = %S
        writeable = Yes
        browseable = No
```

17.d - SWAT

Pour régler à distance et avec une interface graphique votre serveur, vous pouvez vous connecter avec n'importe quel navigateur, de n'importe quel OS et n'importe machine (du moment que la sécurité le permet) sur le serveur, pour peu que celui ci exécute swat (Samba Web Administration Tool). Habituellement, ce service HTTP tourne sur le port 901



18) Exercices

18.a - Connexion, premiers contacts

> *Recommandation : Pour toute commande, vérifiez le fonctionnement avec **man** (la documentation en ligne) : par exemple **man ls***

Connectez vous sur la machine.

Changer votre mot de passe. (si vous avez le droit :-( ). Vérifiez le bon fonctionnement de votre mot de passe.

Affichez le nom de votre répertoire courant

Affichez le contenu de ce répertoire.

En utilisant Alt + les touches de fonctions (F1 à F6), connectez vous sur les différents terminaux. Avec la commande **tty**, affichez le nom du terminal. Promenez vous de terminal en terminal en lançant différentes commandes (**ls**, **tty**, **pwd**, **cd**...) et appréciez le coté multiutilisateur de Linux.

Grâce à la commande **who**, affichez tous les utilisateurs loggués sur votre machine

Comment compter le nombre d'utilisateurs connectés ? Comment afficher le temps de connexion ?

(si vous avez le droit) Lancez un arrêt de la machine (commande **shutdown**). Remarquez l'arrêt de tous les processus, et le message indiquant que tout est ok.

Pour les salles équipées de multi-boot Éteignez la machine, puis redémarrez... Au démarrage, vérifiez qu'il est possible de choisir l'un ou l'autre des OS. Retournez sous Linux.

En vous promenant dans l'arborescence (la racine sous Linux est /), notez sur papier la structure des répertoires de votre machine (sur deux niveaux uniquement). La structure des répertoires est peu ou prou identique sur les Un*x (Voir http://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)

Testez le résultat de la commande **cd -** en la lançant plusieurs fois.

Le bash (l'interpréteur de commande) offre une caractéristique intéressante : la complétion (terme pas vraiment français) qui complète automatiquement ce que vous tapez, par l'appui de la touche **TAB** :

- Si la machine beep, c'est qu'il y a impossibilité (rien ne correspond)
- Si elle ne beep pas, mais que rien ne se passe, c'est qu'elle hésite. Réappuyez sur **TAB** pour qu'elle vous liste l'ensemble de possibilités, et complétez un peu plus...
- Si elle trouve sans ambiguïté, elle complète, que ce soit le nom d'un fichier ou d'un exécutable...

Pour tester (sans lancer l'exécutable), chercher si il existe un programme qui commence par ***Xc***.

Listez l'ensemble des commande accessibles qui commence par ***add***.

Meme chose par ***user***....

Utilisez le programme **cal** qui donne le calendrier pour l'année 1752. Qu'a t'elle de particulier ? (consulter le **man** à ce sujet).

A quoi sert la commande **touch** ?

A quoi sert la commande **free** ? Testez la.

A quoi sert al commande **top** ? Testez la (q pour quitter).

Affichez le contenu de répertoire /**etc**. Affichez les tailles des fichiers, puis affichez-les par date de dernière modification.

A quoi sert la commande **finger** ? Testez la.

A quoi sert la commande **write** ? Testez la.

A quoi sert **mesg** ?

18.b - Unix Utilisateur

Pour ces exercices, point n'est besoin de commandes complexes... Utilisez vos capacités de déduction, le Bash (et sa complétion magique!!!), et deux trois commandes... Prenez l'habitude de farfouiller... Bon courage....

- Créez dans votre répertoire trois sous-répertoires (**Programmes, Documents, Personnel)**.

- **Programmes** et **Documents** doivent contenir deux répertoires (nommés **anglais** et **français**)

- Supprimez le répertoire **Personnel**

- La sécurité des utilisateurs sous Unix est stockée dans 3 fichiers de configuration - **passwd**, **shadow** et **group.** Copiez **passwd** dans votre home.

- **Cherchez** à l'intérieur les informations vous concernant, et verifiez-les. Notez vos **numéros** (le premier est votre ID, et le second le ID de votre groupe).

- Copiez maintenant le fichier **group** chez vous.

- **Renommez** le en TEST.group. **Recherchez** y le numéro, et **vérifiez** le nom du groupe correspondant.

- Créez un fichier vide nomme **essai** dans votre home directory.

- Faites en deux copies, nommées **essai.1** et **essai**.**1.copie**.

- **Renommez** essai en essai.fichier_original. **Supprimez** ensuite ces trois fichiers.

- Parmi les fichiers de configuration, il en existe un qui liste l'ensemble des **protocoles** connus, et donne le port qui leur sont affectés. Il s'agit du fichier **services**, stocké a l'endroit habituel des fichiers de configuration. Faites en une copie dans votre home, que vous appelerez **fichier_des_ports**. (Nota : Si vous ne vous souvenez pas où trouver ces fichiers de configuration, consulter http://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)

- **Cherchez** y le numero du port *www* (le web), celui du courrier sortant **smtp**, et celui du web securise (***https***).

- Quelle commande permet en une fois de **copier** le fichier /etc/group et /etc/passwd dans le sous répertoire **Documents** de votre home directory ?

- Que remarquez vous concernant les **extensions** dans un système de fichiers Unix ?

- Testez la commande **file**. A quoi sert-elle ? Qu'en pensez vous ?

- Testez la commande suivante *touch test{1,2,3,4,5,6}*. Puis t*ouch test[a,b,c,d}{1,2,3,4}*. Regardez le contenu de votre répertoire. A quoi sert {} ?

- **Les JOKERS : * et ? Listez** tous les fichiers dont le nom se termine par 1.

- Puis **ceux** qui contiennent un a.

- **Supprimez** tous les fichiers de 6 caractères, et qui ont un a en 5eme.

- **Supprimez** tous les fichiers test1 test2 test3, mais pas les testa1, testa2,testb1 etc

- **Supprimez** tous les fichiers dont le nom commence par test

- Nous allons continuer à prendre en mains le système, grâce à des fichiers de personnalisation. Lorsque vous vous connectez, le bash (si c'est votre shell par défaut) exécute un certain nombre de tâches, et notamment la lecture de fichiers de configuration. Par défaut, il lit le fichier /etc/profile (général à tous les utilisateurs), puis cherche dans votre home directory la présence des fichiers suivant (dans l'ordre) :
  .bash_login
  .bash_profile
  .profile
  Le point en entête du nom masque automatiquement ce fichier lors d'un ls (seul l'option a permet son affichage). Ces fichiers peuvent contenir des commandes qui seront alors exécutées avant de vous donner l'accès au clavier. On peut aussi y définir des variables d'environnement. Par exemple, on peut lancer la commande umask (qui définit les droits par défaut sur vos fichiers et

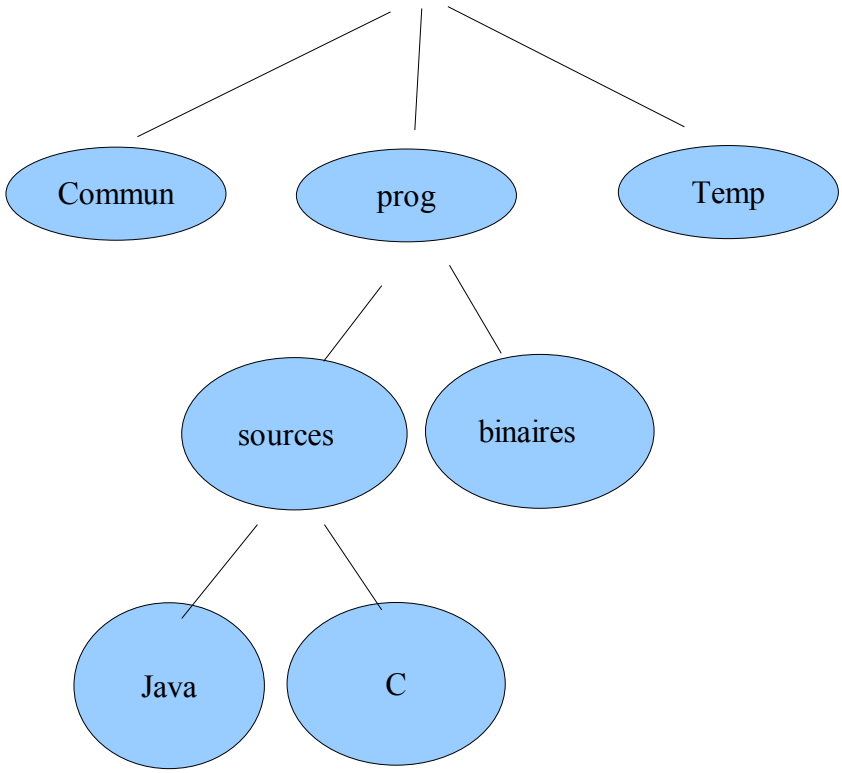
répertoires), ulimit (qui limite vos processus), des alias (pour renommer une commande, afin de la rendre plus pratique, comme ll pour ls-al), et définir PS1 et PS2 (qui sont les prompts du shell, c'est à dire le texte indiqué sur la ligne de commande.) Si votre PS1 vaut '\d ->A vous', alors le pc affichera la date suivi de la flèche et A vous à chaque commande. (Vous trouverez une documentation dans info bash, mais voici quelques raccourcis courant :
\d la date, \h le nom de la machine , \H le nom complet, \l le nom du terminal , \n un saut de ligne, \t l'heure , \u le nom de l'utilisateur, \w le répertoire courant... Vous pouvez bien sûr mélanger ces commandes avec des lettres alphabétiques...).
Testez **export PS1='--\t--$'** dans votre .profile. Puis améliorez votre .profile, en créant quelques alias de commandes (man alias), et en personnalisant votre PS1 et PS2.

18.c - File System

Il s'agit de tester les possibilités du file system. Essayez de réaliser les exercices, en réglant les droits pour l'utilisateur guest (Il s'agit de régler les propriétés pour se protéger de cet utilisateur : vérifiez ses caractéristiques avant).

Ensuite, faites un telnet sur des postes voisins (en tant que guest !!!), et testez les droits sur ses répertoires...(Si le telnet ne marche pas, passez simplement dans /home de votre machine...)

créez une arborescence pour vos programmes dans votre home directory:



Tous ces répertoires doivent vous appartenir, (pas à guest, mais à vous!!!) et il sont en lecture seule, et les fichiers contenus (créez en deux ou trois, même si vous ne connaissant le langage indiqué) doivent être protégé. Seul le répertoire binaire accepte qu'un guest puisse y écrire des fichiers.

Créez ensuite un répertoire /ftp. Nous l'utiliserons pour simuler le comportement d'un serveur ftp. Dans celui ci, il y a deux répertoires : pub et incoming. Dans pub, vous pouvez voir et récupérer tout ce qui s'y trouve. Par contre, vous n'avez aucun droit de modifier, d'ajouter, ni d'effacer des fichiers. Dans incoming, vous pouvez entrer et déposer des fichiers. Par contre, vous n'avez même pas le droit de lister le contenu (un serveur ftp dépend d'un administrateur, qui jugera s'il est bon de publier votre fichier. Il le déplacera alors vers pub. Vous n'avez en aucun cas le droit de lister les fichiers que d'autres ou vous même avez déjà déposé en incoming). Essayez de créer ces deux répertoires, et d'y affecter les bons droits.

Essayez de créer un répertoire aveugle, dont seul le propriétaire peut voir le contenu. Cacher à l'intérieur un répertoire secret, lui en accès total à tout le monde. L'objectif est de ne pas pouvoir trouver ce répertoire, mais pourtant d'y avoir un accès complet...

Connectez vous en tant que guest sur votre machine, et vérifiez (utilisez la commande touch par exemple).

Ceci terminé, testez les machines des autres. telnet est un programme qui simule un terminal sur la machine demandée : tapez telnet pc412, et vous serez exactement comme si étiez physiquement sur cette machine. Sur l'écran virtuel, vous verrez les signes habituels d'une connexion (login, puis password...) Utilisez le compte guest, dont le password est soit rien, soit guest.... Une fois connecté, vérifiez les répertoires crées et les droits correspondants. (En fait, vous pouvez tout a fait tester le repertoire des autres directement sur votre machine. Mais faire un telnet est plus amusant, même s'il n'apporte rien....)

18.d - Commande Find

trouvez les fichiers dont la taille dépasse deux mégas.

Ceux qui vous appartiennent.

Ceux accédés dans la journée

Affichez l'arborescence de /etc

Ceux dont les permissions 4000 sont actives (le SUID, véritable bête noire de l'administrateur...)

Ceux supérieurs à 1 Mo, et pas accédé depuis 1 mois (utile pour l'administrateur)...

Quelle commande utiliseriez vous pour compresser tous ces fichiers (gzip...)

comment faire pour supprimer partout tous les fichiers créés par toto ?

18.e - Liens

Copier le fichier /etc/passwd dans votre directory.

Créez deux liens sur ce fichier passwd1 (lien en dur) et passwd2 (symbolique)

vérifiez par ls -l les différences. Comparez avec l'outil stat.

Modifiez les droits de passwd. Vérifiez les droits des trois fichiers?

Ajoutez une ligne dans passwd2. Vérifier le contenu de passwd1 et passwd.

Supprimez le fichier passwd.

Vérifiez le détail des fichiers pass*. Faites un cat de passwd1 et de passwd2. Que constatez vous ?

Créez un lien sur un répertoire. Quel peut être l'intérêt de cette action ? Consultez le répertoire /usr/src. Que constatez vous ?

Cherchez comment faire apparaître les **inodes** avec *ls*.

Est ce que plusieurs noms dans le **file system** peuvent pointer sur la meme **inode** ?

Créer un répertoire test. Regardez **l'inode** de ce répertoire. Créez des fichiers dans ce répertoire. Regardez toutes les **inodes** de ce répertoire test.

Consultez le numéro d'inode de /usr. Remarquez le nombre de liens qui pointe sur ce répertoire !!! Cherchez quels peuvent bien être tous ces liens... Cette recherche devrait vous indiquer comment on construit un système prétendument hiérarchisé (une arborescence) sur un support plat (un disque)!!!

Au démarrage, certaines distributions Linux utilise un système hérité du System V. Il s'agit d'un répertoire */etc/rc.d* qui contient des sous répertoires correspondant aux différents niveaux de démarrage de la machine (configurés dans */etc/inittab*). Les modes de démarrage habituels sont 3 ou 5. Visitez tous ces répertoires et essayez de comprendre le mode de fonctionnement du démarrage System V, et ce qui se passe dans ces répertoires.

Dans le même ordre d'idées, listez le contenu du répertoire */etc/rc.d/init.d.*

Vous y trouverez un script de lancement du service syslog (journal des évènements, qui stocke tout ce qui se passe dans un ou plusieurs fichiers, dont /var/log/syslog, ou /var/log/message, selon les réglages de /etc/syslog.conf bien sûr). Trouvez les liens dans les répertoires /etc/rc.d qui pointent sur ce script (Selon le type de liens utilisé, il pourrait y avoir deux solutions. Trouvez les, et expliquez pourquoi c'est ce choix qui a été fait!!!).

18.f - Redirections et pipes

Exercices préparatoires :

Décrire le rôle des commandes suivantes :

cat	.....
echo	.....
less	.....
more	.....
wc	.....
grep	.....
sort	.....
cut	.....
uniq	.....

A quoi sert le fichier spécial /dev/null ?

Les REDIRECTIONS

1. Grâce à **echo** qui permet d'afficher un texte, affichez le texte "ma date de naissance est le ..."

2. Créez un fichier qui contient cette phrase, suivie du calendrier du mois de cette année là..

3. allez dans la racine, et écrivez une commande qui liste l'ensemble des répertoires et fichiers qui vous sont interdits (sous entendu : comment savoir tout ce qui vous est interdit d'accès ?). *Ici, réfléchissez.. En fait, Il faut d'abord trouver comment lister tout, et tout ce qui est dans tout ,et tout ce qui est dans tout qui est dans tout, etc (man est votre ami). Ensuite, dans le chaos obtenu, ce qui nous intéresse, finalement, ce ne sont pas les bonnes réponses.. Uniquement les mauvaises. (mais ne passez pas trop de temps là dessus, ce n'est pas facile à trouver, bien que ce soit très simple... Vous pouvez passer à la suite, en laissant votre esprit réfléchir sur le problème)*

4. Créez un fichier qui contient la liste des fichiers présents dans votre répertoire.

5. Ajoutez à ce fichier la liste des utilisateurs connectés.

6. Allez dans /tmp, et afficher le contenu de l'ensemble des fichiers.. (pas le nom des fichiers, mais les contenus de chaque fichier). Récupérez les messages d'erreurs dans un fichier Error

7. Même chose, en jetant les messages d'erreurs à la poubelle.

8. La commande **echo** permet d'envoyer un texte vers la sortie standard. En utilisant cette commande, ajoutez un texte en fin du fichier de l'exercice précédent. Ajoutez une ligne de séparation dans le fichier (une suite de – par exemple)

9. La commande **df** affiche un bilan des partitions, la commande **mount** affiche un bilan des points de montage et des partitions montées dessus, la commande **uptime** affiche le temps depuis lequel le SE fonctionne : Grâce ces commandes, créez un fichier BILAN qui contient toutes ces informations, les unes après les autres.

10. En combinant avec des **echo**, et en exploitant les possibilités de chacune des commandes, essayez d'obtenir le fichier le plus clair et le plus complet possible. (nota : vous pouvez même tenter une version en html, puis l'ouvrir avec un navigateur

11. **Bc** est une calculatrice en mode texte. Lancez là, et tapez des opérations (5+2 (validez) 5*4 (validez) quit (pour arrêter)). Grâce à un éditeur de texte (**nano**, **pico, joe,** voire **mc)**, créez un fichier qui contient quelques opérations (une par ligne).

12. Affichez le contenu de ce fichier.

13. Débrouillez vous pour obtenir les résultats des opérations décrites dans ce fichier.

14. Même chose, en récupérant les réponses dans un fichier nommé **reponses**.

15. sachant que la commande **mail user** permet de lui envoyer un mail, essayer de lui envoyer directement ce fichier **reponses**

Un peu de réflexion :

que fait **cat** tout seul ? (un peu d'aide si vous êtes bloqué, CTRL+D simule la fin de fichier au clavier) Pourquoi ?

Expliquez comment marche **cat fichier** ?

A quoi sert **cat > fichier** ?

A quoi sert **cat >> fichier** ?

Cherchez le rôle de la commande **sort**.

Écrivez plusieurs mots (1 par ligne) dans un fichier **test.mots.**

Affichez cette liste triée, grâce à la commande **sort**.

Que fait **sort** tout seul ?

Et **sort > fichier** ?

Les PIPES

Utilisateurs connectés : commande who

Utilisateurs connus : getent passwd

1. listez les utilisateurs connectés, et envoyez le tout dans un fichier
2. listez les utilisateurs connectés, et triez les par nom
3. listez les utilisateurs connus, triez les par nom et envoyez le tout dans un fichier
4. listez tous les utilisateurs connectés, uniquement leurs noms, et triez le résultat
5. listez tous les utilisateurs connus, uniquement leurs noms, et triez le résultat
6. Un fichier **access.log** vous est fourni sur ma page. *Ce sont les informations que donne un serveur web à propos de toutes les visites qu'il a reçues.* Affichez son contenu. Comptez le nombre d'accès total.
7. Affichez tous les accès à la page w00tw00t
8. comptez le nombre d'accès à la page w00tw00t
9. comptez le nombre d'accès à cette page pour deux ou trois jours précis
10. Cherchez les accès en error (le code 404 est le code qui permet de reconnaître une page en erreur). Attention, 404 peut apparaître aussi dans une taille de fichier (14042 par exemple, qui pourrait être la taille d'une image)
11. Combien y 'a t'il de pages en erreur ?
12. Les pages w00tw00t sont des tentatives d'attaques. Récupérez les adresses IP des machines qui vous attaquent.

13. Eliminez les doublons. Combien y 'a t'il de machines différentes qui vous ont attaqué ?

14. Lister les différents navigateurs qui ont visité votre site (dernier champ)

15. la commande cat permet d'afficher le contenu d'un fichier, et * est un caractère joker qui veut dire n'importe quel ensemble de caractères. Donnez la commande qui permet d'afficher le contenu d'un ensemble de fichiers

16. affichez toutes les lignes des vos pages web

17. comptez le nombre de lignes de vos pages web

18. affichez la liste de tous les processus

19. recherchez tous les processus bash qui tournent actuellement

20. recherchez tous les processus bash triés par tty et croissant

21. Revenons aux utilisateurs connus. Affichez le nom de tous les étudiants (juste leur nom et prénom)

22. Triez tout par nom. Comptez le nombre de personnes qui peuvent se connecter.

23. Ne sélectionnez que les noms de familles. Combien y a t'il de noms de familles différent ?

24. Quels sont les noms de famille qui apparaissent plusieurs fois ? (faites un man uniq)

25. la commande tr permet de transformer les caractères. La mise en majuscules se fait de la façon suivante : tr [a-z] [A-Z] : Expliquez pourquoi.

26. Utilisez la commande tr pour afficher tous les noms des utilisateurs connus en majuscules.

27. Recherchez tous les utilisateurs dont le prénom est pauline (en ignorant la casse) : (2 façons de faire.)

18.g - Processus.

Affichez les processus actifs grâce à la commande ps

lancez la commande plusieurs fois. Quels sont les processus affichés ? Que remarquez vous concernant les pids ?

Testez l'option x de ps. Qu'obtenez vous ?

Obtenez tous les processus avec l'option a.

Testez maintenant avec x et a en même temps, puis ajoutez l'option l.

top permet de savoir l'activité de votre machine, à intervalle réguliers. Lancez cette commande. Top lit votre clavier, et réagit à votre frappe. Tapez q pour quitter.

Relancez top, et observez les processus présents. Il est possible de modifier les champs affichés (en ajouter, en supprimer) grâce à la commande field (f). Ajoutez le PPID, l'UID.

Revenez maintenant sur votre machine, et consultez la documentation de top, notamment le sens de TIME, de load average, de %CPU, de STAT.

Grâce à la commande k, essayez d'arrêter un processus.

Lancer une recherche du fichier xf86config en tache de fond (sur un autre terminal, de façon à laisser top tourner).

Lançons un programme bien plus long. Recherchez en tache de fond tous les fichiers dans toutes l'arborescence qui contienne le mot nameserver. Observez ce qui se passe dans top.

Vérifiez si le programme inetd tourne en mémoire.

Faites apparaître l'arborescence de vos taches (essayez sans X, puis X c'est à dire connecté en mode graphique).

listez les processus. Choisissez en un, et essayez de l'arrêter.

Repérez votre processus bash. Tuez le. Que se passe t il ?

Essayez de tuer le processus init.

Recherchez le processus login qui vous concerne. Tuez le.

Imaginez ce que donnerait un kill -1 du programme init.

Écrivez le fichier suivant dans votre home.

```
#!/bin/sh
while :
do
        date '+il est %T'
```

```
        sleep 60
done
```

Ce fichier, que vous appellerez comme vous voudrez, est un petit script shell. Il exécute sans fin l'affichage de l'heure, toutes les minutes. Rendez ce script exécutable (chmod +x NomScript), ou click sur la propriété correspondante en mode graphique, et lancez le en tache de fond.

Vérifiez que vous récupérez la main, et que l'heure s'affiche régulièrement. Vérifiez avec ps les différentes taches actives (option -l). Tuez ensuite cette tache.

18.h - Files System, mount, et formatage

1. Formatez une disquette

2. mettez un file system dos sur cette disquette, montez la dans un répertoire /test, puis copiez un fichier dessus... Démontez la disquette.

3. Installez un file system Linux sur la disquette, montez la, et vérifiez les points de montage de votre machine...

4. créez une disquette de boot (consulter le fichier /etc/lilo.conf pour connaître le nom et l'emplacement du noyau utilisé sur votre machine). Bootez avec cette disquette. Doit-on créer un file system dessus ?

5. Dans /home/divers, vous trouverez deux images de disquettes boot et root (démarrage de Linux). Générez les disquettes correspondantes, et testez-le...

6. Montez un cdrom dans votre arborescence, puis copiez un des fichiers du cd sur la disquette... Après vérification, démontez tout...

7. Montez un cdrom d'installation de Linux (distrib de votre choix), et consultez l'arborescence. Y a t-il des images pour démarrer l'installation de linux. Dans quel répertoire ? Y a t-il une aide ? Quels sont les images à choisir ?

8. Le serveur apollo offre un répertoire /test en nfs. Montez ce partage dans le /mnt de votre machine, et consultez le contenu de ce répertoire. Lisez le contenu du message qui vous y attend...

9. La machine PC517 est une machine windows qui offre des partages réseaux en Netbios... Après vérification que votre machine le supporte (désolé si ce n'est pas le cas), essayez de voir le contenu du répertoire echange de la machine pc517.

10. Votre disque dur contient un répertoire de swap linux. Il existe une astuce à ce propos. Imaginez que vous deviez faire une nouvelle installation de linux sur votre machine, or vous ne possédez pas de solution de sauvegarde. Il est possible d'utiliser temporairement cette partition pour y sauvegarder votre home directory, d'installer la nouvelle version de linux, de restaurer le home, puis de reactiver le swap. Essayez de faire cette sauvegarde. Quand elle est faite, testez la, puis réactivez le swap...

PS : A partir de l'exercice 5, lancez vi sur un terminal, et consignez toutes les solutions dans un fichier exoFS

Commandes à connaître, et à consulter...
cpio dd df free mount umount mkfs et ses dérivés mkswap smbclient swapon swapoff tar

18.i - Shell Scripts

Faire un script qui affiche la date, le répertoire courant, et le nombre de fichiers de ce répertoire

Modifiez le script précédent pour qu'il fasse la même chose pour un répertoire passé en argument

Modifiez le script précédent pour qu'il vérifie qu'on lui donne bien un argument, et que cet argument soit bien un répertoire.

Écrire un script qui rend exécutable le fichier passé en argument (très pratique, n'est ce pas ??). Introduire des tests pour vérifier sa validité (ne rendre exécutable que des scripts !!!)

Écrire un script qui vérifie si la personne dont le nom est passé en argument est loggé sur la machine.

Écrire un script qui tue un processus par son nom.

Écrire un shell qui recherche tous les liens (symboliques ou non) sur un fichier, et les affiche.

Écrire un script qui recherche un fichier à partir d'un certain point de l'arborescence, et élimine les messages d'erreur (l'objectif est de simplifier l'invocation de find).

Écrire un script loop qui permet la création d'une suite de nombres (par exemple loop 1 20 sort tous les nombres de 1 à 20). Se servir de cette commande pour pinger toutes les machines de la salle...

écrire un script qui prend en argument le nom d'un utilisateur, et sort les informations le concernant sous la forme :
login :
Nom Complet :
Répertoire :
Shell :

Écrire un script procuser qui donne l'ensemble des processus appartenant à un utilisateur donné en argument. Votre script doit vérifier sa bonne invocation, et sinon affichez le message suivant:
Usage : procuser NomUser

Écrire un script d'administration de la machine : sachant que la commande df permet de connaître l'occupation des disques durs, et que le journal des évènements de votre machine s'appelle syslogsyslog, (faire un man df, et contrôlez le contenu de /etc/syslog.conf, qui est le fichier de configuration du syslog), construisez un fichier bilan affichant tous les logins de root, les points de montage, l'occupation des partitions. Envoyer ce fichier par mail à root.

Modifiez le script qui rend actif un script (exercice 4) en lui faisant ajouter des commentaires sur la date de création, l'auteur, et une petite description du script (par exemple, je lance en passant en argument le fichier à activer, mon nom est demandé, ainsi que l'objectif du script : le tout est ajouté et bien présenté en début du script). Cela permet de bien documenter et de standardiser...

Écrire un script menu, chargé de lister des options en les numérotant, et de demander à l'utilisateur de faire son choix. Si ce choix est valide, on quitte le script en rendant le chiffre choisi par l'utilisateur. Les différentes options du menu sont passées en argument lors de l'appel à menu (Attention à l'IFS : il y a deux solutions envisageables...). Tachez de bien réaliser ce travail, il vous sera utile...

18.j - Exercices compilation et patch du noyau.

Téléchargez ou récupérez les sources complètes de l'avant dernière version de votre noyau (en format tgz, ou tar.bz2)

En ayant soin de bien pointer les conséquences, décompactez les sources AU BON ENDROIT.

Créez le lien symbolique sur le nouveau noyau.

Lancez la configuration du noyau.

Compilez votre noyau, et les modules....

Dans un but de test, prenez une disquette, et formatez la.

Dupliquez le noyau sur la disquette.

Bootez ensuite avec cette disquette, et vérifiez le bon fonctionnement du noyau...

Si votre noyau est "bancal", reconfigurez le, et recompilez le.

Lorsque vous disposez d'un noyau opérationnel, modifiez le lilo.conf, afin de proposer le démarrage de ce nouveau noyau, et gardez un accès à l'ancien !!

Vérifiez le bon fonctionnement du démarrage (ancien et nouveau).

Récupérez maintenant le patch du dernier noyau.

Patchez votre noyau.

N'oubliez pas de renommer le répertoire, et de vérifier la validité du lien symbolique.

Recompilez le noyau, installez le, et testez le (laissez toujours une possibilité de démarrer l'ancier).

Modules : Compilez maintenant votre noyau en activant les modules, et en désactivant le « kernel module loader » (le chargeur automatique de module, kmod). Choisissez les options correspondantes à votre machine (par exemple pas de SCSI, IDE, chipset VIA, carte mère PII, processeur Athlon, carte réseau compilée en dur, etc). Dans la partie File system, vous compilerez MSDOS, FAT et VFAT en modules...
Compilez les modules, installez les, et modifiez votre loader...
Testez votre noyau...
Vérifiez les messages du noyau (dmesg | less)
Essayez de monter une partition ou une disquette Windows. Que se passe t'il ?
Utilisez insmod pour insérer le module vfat dans le noyau (attention, vous vous apercevrez que vfat dépend de fat, qui dépend de msdos)
Vérifiez que le montage fonctionne...
Listez les modules du noyau, puis supprimez les...
Testez maintenant la commande modprobe vfat. Que se passe t il ?
Recompilez votre noyau, en ajoutant le « kernel module loader ».
Testez votre noyau. Montez un support vfat... Que se passe t il ?

Malheureusement, tout n'est pas toujours aussi simple... Les modules sont passionnants, car il permettent l'ajout de fonctionnalité à la volée... En effet, vous pouvez toujours compiler des modules (par make menuconfig; make dep; make modules;make modules_install, puis un modprobe et le tour est joué, sans arrêter le noyau courant !!!!)
Recompilez votre noyau, avec le kernel module loader, et votre carte réseau en module...
Redémarrez avec ce nouveau noyau...
Essayez de configurer votre carte réseau (ifconfig eth0 192.xxx.xxx.xxx)
Pourquoi cela ne fonctionne pas ? (qu'est ce que le kmod essaye de résoudre ?)

RedHat,Mandrake...	***Debian***
Ajoutez dans le fichier /etc/modules.conf la ligne suivante **alias eth0** ***driver*** (par exemple alias eth0 3c59x)	Ajoutez dans le fichier /Etc/modutils/aliases la ligne **alias eth0** ***driver*** (par exemple alias eth0 3c59x) puis lancez la commande : update-modules

Relancez maintenant la commande ifconfig... Ca marche !!!

18.k - Sauvegardes, package, et démarrage de la machine...

Formatez une disquette, et sauvegardez votre home, sans faire de file system...

Récupérez ensuite le contenu de la sauvegarde dans un sous répertoire de tmp...

Reformatez la disquette, et transformez la en disquette bootable

démarrez votre machine avec.

Redémarrez normalement.

Pour la suite de l'exercice, nous allons installer un serveur de news sur vos machines. Vous devrez créer des forums, et vérifier le bon fonctionnement à partir d'une machine distante. Enfin, vous devrez automatiser le démarrage du service...

Le serveur de courrier s'appelle **InterNet News server (inn).** Vous devriez trouver package d'installation sur les cd d'origine de votre distributions. (Essayez de tout faire en interface texte !! Attention aux dépendances !!!)

.....

.....

.....

Une fois le package installé, vous allez devoir configurer le serveur : Rendez vous dans ***/etc/news.*** Le fichier inn.conf contient des informations sur votre serveur, par exemple organization ou fromhost, qui apparaîtront dans vos posts (jetez un coup d'oeil à ce fichier..). Le fichier ***nnrp.access*** contient la liste des machines qui peuvent consulter votre serveur de news, et y déposer des posts. De base, tout est fermé : la syntaxe des lignes de ce fichier est :
machine:permissions(ReadPost):nomutilisateur:motdepasse:group

```
# Default to no access
*:: -no- : -no- :!*
# Allow access from localhost
localhost:Read Post:::*
```

La lecture de ces réglages qui interdisent tout accès à tout le monde, sauf à la machine localhost, devrait vous permettre de deviner comment faire exactement l'inverse (accès à tout le monde !!!)

Ensuite, dans une utilisation réelle, on devrait indiquer les sites distants, avec qui vous échangerez tous les posts. Ici, nous n'installerons pas cette possibilité (sinon, modifiez le fichier ***/etc/news/newsfeed***)

Enfin, pour indiquer la liste des forums que nous relaierons (ou éventuellement pas !!!), vous devez configurer le fichier ***/var/lib/news/active***.

```
control 0000000000 0000000001 y
control.cancel 0000000000 0000000001 y
junk 0000000000 0000000001 y
test 0000000000 0000000001 y
to 0000000000 0000000001 y
espacefoch 0000000001 0000000001 y
administratif 0000000001 0000000001 y
```

Les 3 ou 4 premières sont plus ou moins standards, et vous pouvez créer vos propres newsgroups ensuite (ici espacefoch, et administratif). Les deux nombres suivants sont le himark et le low mark (voir ***man active***), et la lettre (y) indique que l'envoi de posts est possible (encore ***man active***).

Créez vous quelques newsgroups, puis lancez le serveur innd (Utilisez si possible les consoles virtuelles : sur une vue, faites un ***tail-f /var/log/messages*** afin de voir les messages d'erreurs, sur une autre, lancez le serveur (à vous de trouver comment).

.....

.....

.....

Sous un lecteur de news (éventuellement en mode graphique) , essayez de récupérer, puis d'écrire des posts !!!).

.....

.....

.....

Quand tout fonctionne, modifiez les scripts de démarrage pour que le serveur ne démarre automatiquement qu'en **init 5** et pas en **init 3**.

.....

.....

Désinstallez ensuite entièrement le logiciel....

.....

.....

Sachant que vous disposez de 6 consoles texte, et d'une septième graphique, essayez de rajouter 2 consoles (la 8 et la 9). Redémarrez la machine pour vérifier qu'elles existent.

.....

.....

Votre système d'exploitation consigne tous les événements dans différents fichiers dits "de log". L'un de ses fichiers est ***/var/log/messages***. La commande ***/usr/bin/tail -f*** permet de visualiser en continu le contenu de ce fichier (testez cette commande sur une vue, puis logguer vous en root sur une autre afin de tester). Imaginons que vous voulez, dans le cadre de l'exploitation d'un serveur, avoir toujours une vue affichant le contenu de ce fichier : Ajouter une dixième vue, sur laquelle automatiquement cet affichage aura lieu (au lieu d'un login). PS : il y a une petite difficulté... N'hésitez pas à tester !!!

.....

.....

18.I - Serveurs et démons...

Créez un répertoire public sous la racine

exportez le en NFS en lecture écriture. Vérifiez avec un client que l'on peut se connecter. Créez un fichier avec la commande touch.

Ajoutez une nouvelle exportation : /essai

Connectez un client.

Mettez en place un serveur ftp anonyme sur votre machine. Attention, lisez bien la documentation : Il faut installer les commandes dans l'arborescence du serveur... Créez un petit texte de bienvenue, et un second texte que vous nommerez avis.

Lancez votre serveur, et vérifiez le bon fonctionnement avec un client.

Certains services passent par inetd. Vérifiez le bon fonctionnement de celui ci, en coupant l'accès à telnet (il suffit de commenter la ligne concernant ce service dans le fichier de configuration).
Redémarrez ensuite le super serveur inetd pour vérification.

Installons maintenant un serveur SaMBa. Ce serveur permet d'offrir un accès transparent aux clients NetBios (c'est à dire aux machines Windows).

Créez un répertoire /samba.

Partagez ce répertoire en lecture seule pour les clients Windows (testez la configuration avec testparm).

Lancez le serveur, et vérifiez le fonctionnement avec un ou plusieurs clients Windows.

Il est possible de configurer Samba à distance, en utilisant un navigateur WEB. En effet, il existe un serveur SWAT qui offre un écran de contrôle en HTTPD, sur le port 901.

Essayez d'activer ce service sur votre machine, et connectez vous à partir de n'importe quel type de client (Win ou Linux...)

Grâce à SWAT, modifiez la configuration de Samba (OS Level par exemple...)

19) Exercices récapitulatifs

1. connectez vous
2. changez de password
3. inspectez votre home directory
4. remontez à la racine
5. affichez le contenu
6. allez dans etc et listez son contenu
7. affichez le contenu de passwd, et de inittab
8. testez les différentes commandes cat, more et less
9. retournez dans votre home directory
10. listez les utilisateurs connectés, et envoyez le tout dans un fichier
11. listez les utilisateurs connectés, et triez les par nom
12. listez les utilisateurs connus, triez les par nom et envoyez le tout dans un fichier
13. listez tous les utilisateurs connectés, uniquement leurs noms, et triez le résultat
14. listez tous les utilisateurs connus, uniquement leurs noms, et triez le résultat
15. écrivez une commande qui donne le nombre d'entrées des répertoires suivants: /usr /bin /var /etc
16. écrivez une commande qui liste la taille et le nom des fichiers, dans l'ordre croissant (pipes assez complexe, essayer d'abord de lister les fichiers, puis de couper comme il faut (il y a un problème d'espace, voir tr), puis de ne récupérer que les bons champs, et enfin de trier...
17. VI Editor :
    En tant que simple utilisateur, créez un répertoire travaux dans votre home directory
    copiez /etc/fstab
    renommez le en fstab.test
    Ce fichier contient les montages standard sur votre machine (c'est à dire dans quel répertoire verra-ton tel périphérique...)
    En imaginant que vous disposez d'un nouveau disque dur (hdb), coupé en deux partitions (hdb1 contenant un file systèm dos, et hdb2 contenant un file system Linux qui correspondra à /home), modifiez le fichier fstab.test.
    On imagine maintenant que vous changez de hard (plus de IDE, tout en SCSI). Le nouveau nom des périphériques est sda et sdb...
    Modifiez le fstab en conséquence
18. *Vous savez gérer les adresses IP des machines. Vous savez que vous pouvez pinger une machine dont vous connaissez l'adresse. Si vous voulez pinger une machine par son nom, vous devez : Soit disposez d'un DNS dont vous serez client, soit renseigner le fichier /etc/hosts (correspondance nom et adresse IP)*

pinger des machines voisines
notez leurs adresses, et donnez leur un nom
remplissez le fichier hosts
ping le nom de la machine
connectez vous au serveur de cette machine (http://nom/)

19. Modifiez le nom de votre machine (fichier HOSTNAME)

20. *LILO est le loader (le lanceur) d'OS de Linux. Il vous permet de choisir parmi plusieurs versions de Linux, et de lancer d'autres OS. Il est commandé par le fichier lilo.conf (dans /etc), et le petit programme loader est fabriqué par la commande lilo. Ce petit programme sera installé à l'endroit indiqué par la commande boot. Pour fabriquer de nouvelles entrées, il suffit de récréer une image (le nom du fichier noyau) et d'y associer un label (le nom que lilo vous proposera...). Lorsque l'on met à jour son noyau Linux (recompil, mise à jour), il est conseillé de recréer un lilo, pour pouvoir rebooter sur l'ancien image en cas de plantage de la nouvelle.*
Créez une nouvelle entrée dans le lilo, de façon à démarrer la même image avec le nom monlinux (il y aura trois choix : monlinux, linux et dos)

Table des matières