Support formation de base à linux

Première partie : généralités

Linux est un système d'exploitation pour de multiples architectures matérielles, basé dans son concept sur unix.

La quasi totalité des commandes unix se retrouve sous linux, parfois avec quelques spécificités.

Unix

Système d'exploitation né aux Bells Labs (laboratoire de recherche de ATT) en 1969. Evolution du système, notamment réécriture en langage C en 1975.

A partir de 1977, l'équipe de l'université de Berkeley travaille en parallèle sur unix et sort ses propres versions (BSD)

En 1984, Berkeley sort la version 4.2 BSD et ATT le système V

A partir de 1980 sortent des systèmes d'exploitation dérivés d'unix.

On trouve donc sur le marché plusieurs implémentations d'unix possédant chacune ses spécificités.

Points clés d'unix

Système multi-tâches et multi-utilisateurs, orienté réseau, support natif de tcp-ip.

Chaque tâche s'exécute dans un environnement protégé (un plantage de la tâche n'entraîne pas un plantage du système).

Séparation entre le niveau administrateur et le niveau utilisateur.

Possibilité de définir des groupes d'utilisateurs ayant des droits spécifiques. Un même utilisateur peut appartenir à plusieurs groupes.

L'administrateur s'appelle « root » (à ne pas confondre avec le nom de la racine du système de fichiers).

Ecriture en langage C (portabilité)

Trois niveaux : noyau, interface homme-machine (shell), outils divers

Système de fichiers : trois types de fichiers : fichiers « ordinaires », répertoires, fichiers spéciaux (périphériques, tubes)

Sous unix, tout est fichier (fichiers et répertoires, mémoire, périphériques, processus…)

Les commandes sont donc les mêmes pour toutes les opérations.

La structure du système de fichiers est arborescente, à partir d'un noeud initial appelé « root », et symbolisé par le caractère « / » (slash) (c'est donc différent des systèmes windows où le caractère utilisé est « \ » (backslash).

Le caractère de séparation de niveau dans le chemin d'un fichier est également le slash (« / »).

A chaque fichier unix sont attachés différents attributs :

Le propriétaire (créateur du fichier)

Le groupe (ensemble d'utilisateurs)

Les droits d'accès (lecture, écriture, exécution).

Les droits d'accès sont définis individuellement pour le propriétaire, le groupe et les autres (c'est-à-dire qui ne sont pas le propriétaire et qui n'appartiennent pas au groupe du propriétaire).

En plus de ces droits, il existe également d'autres attributs : le bit SUID (le fichier s'exécute avec les droits de son propriétaire), le bit SGID (idem mais pour le groupe), le sticky bit (à l'origine, permet le maintien en mémoire du fichier, mais son utilisation a évolué).

Tous les programmes tournant en mémoire sont appelés des processus. Comme indiqué plus haut, les processus apparaissent dans l'arborescence du système de fichiers.

Tout processus a un processus père, dont il hérite de l'environnement (à l'exception du processus *init*, qui est le père de tous les autres et qui est donc toujours en mémoire).

Chaque processus a une priorité d'exécution (qui peut être changée).

Arborescence unix

L'arborescence unix est unique, quel que soit le nombre des disques, partitions…

La racine de l'arborescence est « / ».

Au dessous de cette racine se trouvent différents répertoires. On trouve sous unix en général les répertoires suivants : /bin (exécutables), /tmp (fichiers temporaires), /dev (périphériques), /etc (paramétrages généraux du système), /usr (ressources du système), /users (données utilisateurs).

Cette arborescence est légèrement différente suivant les systèmes (et notamment sous linux) pour lequel on trouve cette structure générale :

/ : racine du système de fichiers (appelée aussi root)

/boot : contient le noyau linux

/root : répertoire des données de l'administrateur (root)

/home : répertoire des données des utilisateurs (contient autant de sous-répertoires que d'utilisateurs)

/etc (**e**diting **t**ext **c**onfig) : répertoire contenant les fichiers de configuration généraux du système (fichiers de type texte) – correspond approximativement à la branche HKLM du registre windows

/bin (**bin**aries) : contient les binaires de base liés à une librairie

/lib (**lib**raries) : contient les librairies nécessaires aux binaires de /bin

- /lib/modules : contient les modules du noyau

/sbin (**s**tatic **bin**aries) : contient les binaires statiques (non liées à une librairie)

/usr (**u**nix **s**ystem **r**essources) : contient toutes les ressources du système

- /usr/bin : contient les binaires liés à une librairie
- /usr/lib : contient les librairies nécessaires aux binaires de /usr/bin
- /usr/sbin : contient les binaires statiques (non liées à une librairie)
- /usr/share : contient les ressources partagées par les logiciels de /usr/bin
- /usr/local : contient les programmes installés par une compilation locale ou un script d'installation

/tmp : contient les fichiers temporaires généraux du système

/dev (**dev**ice) : contient les périphériques du système

/proc : contient les processus en cours d'exécution, les paramètres courants (dynamiques) du système

/var : répertoire de données (courrier, logs, spool…)

/mnt : répertoire servant au montage des système de fichiers « externes » (disquette, cd-rom, réseau…)

On ne trouve pas sous unix des lettres de lecteurs (c :, d :, a : …..) comme sous dos/windows. Tous les lecteurs de disques et partitions sont rattachés à l'arborescence principale par une opération appelée montage.

Linux et unix

Linux constitue une implémentation libre d'unix. Son origine remonte aux années 90 où un étudiant finlandais, linus torwald, limité par msdos et n'ayant pas les moyens d'acquérir une licence unix, décide de créer son propre unix et lance un appel au peuple par internet pour

l'aider dans cette tâche. Des milliers d'informaticiens adhèrent au projet, qui voit bientôt le jour, puis évolue en continu depuis.

L'architecture de linux est la même que celle d'unix, tout ce qui précède s'applique donc à linux.

Cependant (et comme pour unix), on peut trouver des variantes notamment en ce qui concerne l'arborescence.

Le système linux étant composé d'une multitude de programmes, risquant d'introduire une certaine incohérence, ils sont regroupés (par un éditeur ou une communauté) en un ensemble cohérent appelé « distribution ». Par exemple, RedHat, Mandrake, Suse, Debian, Caldera…

Chaque distribution a ses spécificités. Dans le but d'unifier un peu tout ça, il existe le projet LSB (linux standard base) qui propose une arborescence unifiée.

Peu de distributions respectent pour le moment cette arborescence.

Néanmoins, pour les distributions basées sur la RedHat, on retrouve toujours à peu près la même arborescence (voir copies d'écran).

On notera le remplacement de /users par /home, la création de /sbin (exécutables), /var (données variables), etc…

Compléments sur le système linux

Les niveaux d'exécution (runlevels)

Le démarrage du système est pris en charge par le processus init.

Plusieurs niveaux d'exécutions sont définis, permettant le démarrage et l'arrêt spécifique de différents services. Par exemple, démarrage en mode simple utilisateur, en mode multi-utilisateurs, avec support réseau, en mode texte ou en mode graphique.

Ce mode de démarrage est issu du système V de ATT.

Chaque niveau d'exécution permet de définir les services à démarrer lors du lancement et à arrêter lors de l'arrêt ou du changement du niveau d'exécution).

Ces services sont appelés sous unix des démons (daemon, de Disk And Execution MONitor). Ce sont des processus tournant en arrière plan offrant des ressources au système ou aux utilisateurs. La notion du démon unix est la même que celle du service windows.

Les processus

Tout programme tournant en mémoire s'appelle un processus. Chaque processus possède un numéro d'identification unique, des ressources propres (espace mémoire, espace d'entrées-sorties…). Un processus hérite de l'environnement du processus qui l'a lancé (variables).

Chaque processus peut être supprimé (tué) par la commande kill. Les ressources utilisées par le processus sont alors restituées au système.

Chaque processus est à l'écoute de signaux. Ces signaux peuvent être envoyés à un processus par un autre, mais aussi avec la commande kill. Le signal le plus courant est le signal SIGTERM, qui demande au programme de s'arrêter. Lors de l'arrêt du système, un signal SIGTERM est envoyé en broadcast à tous les processus pour qu'ils s'arrêtent correctement.

Les attributs des fichiers, les droits, notation octale et symbolique

Attributs de propriété

Chaque fichier a un propriétaire (l'utilisateur qui l'a créé) et appartient à un groupe.

Le propriétaire peut être changé avec la commande chown (CHange OWNer)

Le groupe peut être changé avec la commande chgrp (Change GrouP)

Seul le propriétaire (ou root) peut changer les droits de propriété.

Droits

Chaque fichier bénéficie de droits, définis en trois catégories : les droits du propriétaire (u : User), les droits du groupe (g : Group) et les droits des autres (o : Other).

Ces droits peuvent être la lecture, l'écriture et l'exécution.

Pour chaque fichier, on a donc trois groupes de trois attributs. On désigne symboliquement les droits par r (Read : lecture), w (Write : ecriture), x (eXecute : exécution). Ces droits étant définis pour trois catégories : propriétaire, groupe et autres, on note symboliquement les droits sur les fichiers par trois groupes de trois lettres. Aucun droit est symbolisé par un tiret -

Par exemple, un fichier ayant les droits de lecture, d'écriture et d'exécution pour le propriétaire, le groupe et les autres aura les attributs rwxrwxrwx. Le même fichier sans aucun droit d'exécution sera noté rw-rw-rw-. Si le propriétaire a seul le droit d'exécution, on aura rwxrw-rw-. Si le propriétaire a le droit de lecture/écriture, le groupe le droit de lecture seul et les autres aucun droit, on aura rw-r-----

Cette notation est la notation symbolique. Cette notation a le mérite d'être explicite et simplement compréhensible, mais elle est plus longue à mettre en œuvre que la notation octale.

Celle-ci utilise la conversion en base 8 (sur trois bits) des trois attributs de droits. Le premier bit (en partant de la gauche) correspond au droit de lecture. Le second au droit d'écriture, le troisième au droit d'exécution.

Comment convertir : le bit de gauche vaut 4, celui du milieu vaut 2, celui de droite vaut 1. Ajouter simplement ces valeurs : vous aurez un nombre octal (de 0 à 7) correspondant aux droits. Par exemple

7 = 4 + 2 + 1 -> droits de lecture, écriture et exécution

4 = 4 + 0 + 0 -> droits de lecture

5 = 4 + 0 + 1 -> droits de lecture et d'exécution

Ceci pour chaque catégorie d'utilisateurs (propriétaire, groupe et autres).

Exemples :

rwxrwxrwx correspond à 777

rw-r--r-- correspond à 644

rw-r----- correspond à 640

Ces droits peuvent être changés avec la commande chmod

Par exemple chmod 640 nom_fichier donnera les droits de lecture/écriture au propriétaire, lecture seule au groupe et aucun droit pour les autres.

La commande chmod permet également le changement des droits sous forme symbolique.

Bien entendu, il est possible de changer les attributs avec un programme graphique !

Les droits sont en fait notés sur 12 bits. Les neufs bits de droite correspondent aux droits ci-dessus, les trois bits de gauche correspondent aux attributs spéciaux décrits ci-dessous. Si ces trois bits sont omis, ils sont considérés comme égaux à 0 (de manière générale, les chiffres omis sur la gauche sont considérés comme nuls : 7 équivaut à 0007, 45 à 0045, 644 à 0644).

Il existe d'autres type de droits :

Le bit SUID, le bit SGID et le « sticky bit ».

Le bit SUID permet l'exécution du fichier avec les droits de son propriétaire, outrepassant le comportement par défaut qui est qu'un fichier est exécuté avec les droits de l'utilisateur qui le lance.

Une explication : soit un fichier système exécutable pour tous mais qui appartient à root : ses droits sont donc rwx - -x - -x ou encore 711.

Si l'utilisateur « machin » exécute ce fichier, celui-ci s'exécutera avec les droits de « machin ». Si ce programme écrit sur le disque, il ne pourra écrire que dans des zones où l'utilisateur « machin » a le droit d'écriture (il ne pourra pas par exemple écrire dans les répertoires système).

Si le bit SUID est positionné, le programme sera alors lancé avec les droits « root » (administrateur) et pourra donc écrire partout sur le disque, y compris dans les répertoires système. Il est évident que c'est très dangereux pour le système !

Le bit SGID joue le même rôle, mais pour le groupe.

Quant au « sticky » bit, il servait historiquement à permettre le maintien du fichier en mémoire. Il est actuellement utilisé pour les répertoires. S'il est positionné, seul le propriétaire d'un répertoire ou du fichier situé dans ce répertoire peut supprimer le fichier.

Ces trois bits se positionnent de la même manière que les neuf autres : avec la commande chmod.

Pour donner les droits de lecture/exécution à tous et positionner le bit SUID, on tapera :

Chmod 4555 correspondant en notation symbolique à r-s r-x r-x

On voit que le x du premier groupe de droits (droits du propriétaire) a été remplacé par un s : son exécution s'effectue maintenant avec les droits du propriétaire du fichier.

Type de fichier

Chaque fichier contient également l'information de son type :

d pour un répertoire, - pour un fichier normal, b pour un périphérique en mode bloc (exemple : disque dur), c pour un périphérique en mode caractère (exemple : imprimante), l pour un lien symbolique (voir plus loin), p pour un tube nommé (fifo), s pour une socket.

Ces informations apparaissent avec la commande ls –l

Exemple pour un répertoire : ls -dl test (le d signifie que l'on veut l'info sur le répertoire et non sur son contenu)

drwxr-xr-x ……

Le premier d signifie qu'il s'agit d'un répertoire. On note le x (exécution) : pour un répertoire, cela signifie que l'on peut entrer dans le répertoire ou lister son contenu.

L'umask

Qu'est-ce donc que cette chose bizarre ?

L'umask correspond aux droits définis par défaut lors de la création de fichiers ou de répertoires.

On aurait pu définir un masque de la même manière que les droits, mais ça n'aurait plus été un masque ! Le masque correspond en fait au complément à 777 des droits désirés.

Exemple : umask 022 -> droits 644

Ca ne marche pas ! Pourquoi : parce que, par défaut, un fichier n'a pas le droit d'exécution !

L'umask n'affecte donc pas le bit « exécutable ». En fait, il faut enlever 1 à chaque chiffre.

Umask 022 -> droits théoriques 755. On enlève 1 (droit d'exécution) à chaque chiffre. On trouve donc les droits 644.

Par contre, pour un répertoire, on constate bien que le droit d'exécution est positionné (droit d'entrée et de parcours du répertoire).

Pourquoi faire simple quand on peut faire compliqué !

Les liens

Sous unix/linux, un fichier n'est pas caractérisé par son nom, mais par un numéro d'i-node.

On associe ensuite à ce numéro d'i-node un ou plusieurs noms de fichiers. Un fichier physique peut avoir plusieurs noms, qui font référence au même fichier. Ce sont les liens physiques. Ils sont crées avec la commande ln.

Le fichier n'est effectivement détruit qu'une fois que toutes ses références sont détruites. En clair, seule la destruction du dernier lien physique provoquera la suppression effective du fichier. A utiliser avec prudence.

Un autre type de liens beaucoup plus utilisé : le lien symbolique. C'est en quelque sorte un « raccourci » vers le fichier réel. On le crée également avec la commande ln, en y rajoutant l'option –s (symbolic). Contrairement aux liens physiques, les liens symboliques ont une existence propres : il constituent effectivement un fichier sur le disque, mais ce fichier pointe vers un autre fichier. Ils peuvent être détruits sans affecter le fichier physique. De même, la destruction du fichier physique n'entraîne pas la destruction du lien symbolique. Si le fichier pointé par le lien disparaît, celui-ci pointe sur rien.

Les fichiers cachés

Sous linux, les fichiers cachés sont ceux dont le nom commence par un point.

Il ne sont pas affichés par défaut, mais on peut les visualiser avec l'option qui va bien, par exemple ls -l

Les autres attributs de fichiers

Le système de fichiers ext2 possède d'autres attributs (pas très utilisés).

Les trois attributs utilisables sont l'attribut A (ne met pas à jour la dernière date d'accès : minimise les accès disque), i (le fichier ne peut pas être modifié ni effacé – ne peut être défini que par root) et S (écriture synchrone sur le disque).

Ces attributs se positionnent avec la commande chattr. Ils se listent avec la commande lsattr.

Par défaut, aucun de ces attributs n'est positionné.

Le montage des systèmes de fichiers, l'automontage

Comme je l'ai écrit plus haut, l'arborescence unix est unique à partir d'une racine / (root).

Cette arborescence comporte différents répertoires. Ceux-ci sont créés avec la commande qui va bien mkdir (on trouve souvent un alias nommé md, pour les nostalgiques de ms-dos ou les gens pressés ou flémards).

Ce qui est intéressant, c'est qu'on peut associer à n'importe quel répertoire de l'arborescence un périphérique ou une partition quelconque.

Un exemple : vous avez trois disques durs avec des partitions sur chacun des disques, un lecteur de disquette et un lecteur de cdrom. Vous n'aurez pas comme dans windows des lettres a :, c :, d : etc… (limité à z :)

Vous allez simplement associer à un répertoire le disque ou la partition que vous voulez (sans limite de nombre).

Cette opération s'appelle le montage et se réalise avec la commande mount

On associe en fait un périphérique avec un répertoire.

Exemble : mon deuxième disque sur la première nappe ide (hdb) contient une première partition primaire (partition 1) que je veux associer au répertoire /home.

Je tape la commande mount /dev/hdb1 /home

Bon, j'ai un peu abrégé, il y a des tas d'option que l'on peut mettre, notamment le type de système de fichiers (fat, vfat, ntfs, ext2, iso9660, ou même des systèmes de fichiers réseau : samba ou nfs), si un utilisateur peut monter ou pas le système de fichiers, si le système de fichiers doit être monté au démarrage…

Bien que cela soit simple, ça peut devenir pénible de monter une disquette ou un cdrom chaque fois que l'on en change. C'est pourquoi il existe l'automontage (comme sous windows). Il suffit de se placer dans le dossier correspondant et le périphérique est monté automatiquement. Ca ne fonctionne pas superbien, le supermount, et les puristes préfèrent monter « à la main » (cela peut être d'un simple clic). C'est plus fiable.

Les périphériques

On est obligé d'en parler, puisqu'il faut bien connaître les périphériques pour pouvoir les monter. Et là, ça se complique un peu. Pourquoi : parce que le nommage des périphériques

est un cours de mutation. Il y a l'ancienne mode, et la nouvelle mode. Il faut savoir que si l'on utilise la nouvelle mode, des liens sont créés vers les anciens noms.

On commence donc par les anciens noms, qui devraient fonctionner sur à peu près tous les linux de la terre.

Pour faire simple, je ne parlerai que des disques ide et des lecteurs de disquettes, sachant que le schéma est transposable pour n'importe lequel des périphériques d'entrée/sortie, y compris les sorties série, les imprimantes, les disques SCSI…

Les disques durs IDE s'appellent hd (Hard Disk) suivi d'une lettre (sd en SCSI)

a et b sont les disques situés sur la première nappe IDE respectivement en maître et en esclave

c et d sont les disques situés sur la deuxième nappe IDE respectivement en maître et en esclave

Cet ensemble de trois lettres désigne donc un disque physique (de hda à hdh)

Si ces disques sont des disques durs contenant des partitions, les partitions primaires et étendue sont numérotées de 1 à 4, les partitions logiques de 5 à n (sur un disque IDE, le nombre de partitions est limité à 63 – 15 en SCSI)

Rappelons en IDE, il ne peut y avoir au maximum que 4 partitions primaires, ou trois primaires et une étendue, la partition étendue contenant les partitions logiques).

La partition étendue est nécessairement la dernière des quatre (primaires ou étendue). On ne peut pas avoir une partition primaire, puis une étendue, puis une primaire.

En résumé : hda1 à hda4 : partitions primaires ou étendue du premier disque dur de la première nappe IDE – hda5 à hda*n* : partitions logiques du premier disque dur de la première nappe IDE.

CDROM IDE : il n'y a pas de table de partition sur un CDROM. Le périphérique est désigné par le nom du disque. Exemple : disque esclave de la deuxième nappe IDE : hdd

Ces noms sont tous préfixés par /dev, qui est l'emplacement de stockage des périphériques.

Le nom complet pour hda1 par exemple est /dev/hda1

Nouveaux noms :

On trouve maintenant une structure arborescente (voir copie d'écran)

/dev/ide/host0/bus0/target0/lun0/disk correspond à hda

/dev/ide/host0/bus0/target0/lun0/part1 correspond à hda1

/dev/ide/host0/bus0/target1/lun0/disk correspond à hdb

/dev/ide/host0/bus1/target0/lun0/cd correspond à hdc (si c'est un lecteur de cd)

C'est beaucoup plus simple…

Fin de la première partie

Prochaine étape : mode console, consoles virtuelles, le shell et les commandes principales, variables, path, redirections, la complétion, les scripts, les fichiers de configuration système et utilisateur

Etape suivante : l'environnement graphique, le serveur X, les windows managers, les services, l'accès distant, l'installation des programmes

Etape suivante : partages linux/windows, samba

Support formation de base à linux

Deuxième partie : commandes système, configuration

mode console, consoles virtuelles, le shell et les commandes principales, variables, path, redirections, la complétion, les scripts, les fichiers de configuration système et utilisateur

Mode console

Linux peut être utilisé en mode console ou en mode graphique. Le système linux lui-même est en mode console. Une couche graphique est rajoutée au dessus du mode console (de la même façon que sous windows 3.1). Cela permet de ne pas utiliser le mode graphique pour les serveurs (qui n'en ont pas besoin en fonctionnement normal) et donc d'économiser les ressources du système (le mode graphique est un gros consommateur de ressources système), tout en pouvant utiliser le mode graphique pour une configuration plus aisée ou pour l'usage en station de travail.

Cette faculté de travailler en mode console (mode texte) permet l'utilisation de machines peu puissantes en tant que serveur.

Mode graphique

Le mode graphique est constitué par une couche supplémentaire ajoutée au système d'exploitation, totalement indépendante de celui-ci. Cela signifie qu'il n'est pas nécessaire d'installer le mode graphique pour le fonctionnement d'une machine linux (contrairement à windows depuis W95 ou d'autres systèmes tels QNX où l'interface graphique (photon) fait partie du système).

Le mode graphique est le plus souvent implémenté à l'aide d'un serveur graphique (serveur X) qui offre une API de bas niveau pour les fonctions graphiques. Le serveur X est en fait une interface entre l'API et le matériel.

Une couche supplémentaire est ajoutée au serveur X pour offrir des fonctions de plus haut niveau : c'est le Window Manager (KDE, Gnome, IceWm…). C'est cette couche supplémentaire qui offre l'aspect de l'interface graphique utilisateur. La multiplicité des window manager fait qu'un système linux peut avoir des aspects très différents. Chaque WM s'appuie sur une bibliothèque graphique (Qt pour KDE, GTK pour Gnome). Ces bibliothèques sont souvent installées de manière à pourvoir utiliser les applications qui y font appel. Il y en a d'autres (fox par exemple, qui offre un aspect visuel très proche de windows). L'aspect des applications graphiques dépend pour une bonne part des bibliothèques utilisées et n'est donc pas uniforme, contrairement à windows où la plupart des applications utilise l'API graphique standard de Windows (GDI). On trouve cependant également sous windows des API graphiques différentes (GTX et Qt existent également sous Windows).

Consoles virtuelles

Il est possible en mode texte d'ouvrir plusieurs consoles virtuelles (dans une installation standard, il y en 6). Chaque console est indépendante des autres. On passe de l'une à l'autre par une combinaison de touches. L'affichage graphique tourne sur une console virtuelle au delà des 6 consoles texte. Il est possible également de lancer plusieurs serveurs X qui occuperont également une console virtuelle.

En cas de problème sur l'une des consoles (plantage programme par exemple), il est le plus souvent possible d'ouvrir une autre console pour pouvoir régler le problème.

Le shell

Le shell est l'interface homme/machine. Il permet de taper des commandes transmises au système (noyau et processus).

Il y a en fait plusieurs shells. Historiquement, c'est le shell sh qui était utilisé sous unix. Plusieurs version sont ensuite apparues, apportant une évolution des fonctionnalités (par

exemple shell bourne : bash – shell korn : zsh et d'autres encore). Ce document se réfère au shell bourne (bash).

Le shell est un interpréteur de commandes de la même manière que command.com ou cmd sous msdos/windows. Ceux qui ont connu msdos ne seront pas trop dépaysés. Ils apprécieront certainement les fonctionnalités avancées du shell, qui en fait une interface souvent plus rapide et efficace qu'un environnement graphique, mais qui nécessite cependant de connaître les commandes du shell.

Tel est le but du présent paragraphe.

Rappel important : sous unix/linux, la casse des commandes et des noms de fichiers est importante. En règle générale, les commandes sont en minuscules, les noms de fichiers le sont souvent, les variables sont souvent en majuscules. Ce ne sont que des habitudes, pas des obligations. Fichier est différent de fichier ou FICHIER. Il faut s'en souvenir !

Les options des commandes sont spécifiées par un tiret suivi d'une lettre, ou un double tiret suivi de nom de l'option. Le tiret doit être séparé de la commande par un espace.

Les commandes principales

Les commandes principales dont je vais parler ici ne sont pas nécessairement les commandes principales du linuxien pur et dur ou de l'administrateur système féru de la ligne de commande.

La sélection présentée correspond aux commandes que je pense nécessaires un jour ou l'autre pour utiliser ou administrer un système linux.

Dans le même ordre d'idée, les commandes présentent souvent de nombreuses options dont je n'évoquerai que les principales.

Il suffit ensuite d'employer la commande magique « man nom_de_commande » pour avoir tous les détails.

N'oubliez pas avant toute question : RTFM (Read The Fucking Manual), comme les experts aiment à le rappeler dans les forums…

Commandes internes

Les commandes internes sont, comme sous msdos, celles interprétées directement pas le shell et qui en font partie intégrante (built-in). Elles n'apparaissent dont pas en tant que fichiers exécutables. Il y en a peu. La commande magique man ne s'applique pas aux commandes internes. Pour avoir une aide (succinte) sur une commande interne, taper « help nom_de_la_commande_interne ».

Les commandes qui vont être vues dans ce paragraphe sont les suivantes :

Cd, echo, exit, kill, ulimit, export, pwd, test, umask

Les commandes de conditions et de boucles (if, for, until, while) sont également des commandes internes. Elles seront vues plus loin (section scripts shell).

Il y en a d'autres, vous pouvez en avoir la liste avec la commande « help »

cd : Change Directory

Permet de changer de répertoire courant

Syntaxe : cd chemin_du_répertoire

Le chemin peut être absolu (il commence par le caractère /) ou relatif : il s'applique alors à partir du répertoire courant. Si aucun chemin n'est fourni, se positionne dans le répertoire de travail de l'utilisateur (voir pwd ci-dessous, voir variable $HOME)

Exemple : se positionner dans le répertoire /usr

cd /usr

echo

affiche sur l'écran

syntaxe : echo chose_à_afficher

chose_à_afficher peut être une variable, une chaine, le résultat d'une commande…

exemple : afficher « hello word » sur l'écran :

echo ''hello word''

exit

quitte le shell courant. Raccourci : Ctrl-D

syntaxe : exit

kill

permet de « tuer » un processus (se fait en général pour les processus qui ne répondent plus) ou d'envoyer un signal à un processus. Par défaut, le signal TERM est envoyé, demandé au processus de se terminer.

Syntaxe : kill numéro_de_processus, kill –s <signal> numéro_de_processus

Exemple : tuer un processus récalcitrant :

kill –s 9 1590 (envoie le signal KILL au processus 1590)

ulimit

fixe les limites de ressources du système attribuées à l'utilisateur : mémoire, fichiers…

syntaxe : ulimit -<option> nombre

exemple : limiter la taille des fichiers core (dump mémoire en cas de plantage) à 0 :

ulimit –c 0

export

exporte une variable dans l'environnement courant et permet de la définir

syntaxe : export VARIABLE ou export VARIABLE=valeur

exemple : rajouter au path courant un path vers /usr/local/bin

export PATH=$PATH : /usr/local/bin

pwd (Print Working Directory)

affiche le répertoire de travail de l'utilisateur courant

syntaxe : pwd

exemple 1 : affiche le répertoire de travail (pour le superutilisateur)

pwd (renvoie /root)

exemple 2 : définit la variable REPUTIL avec le contenu du répertoire de travail courant :

export REPUTIL=` pwd` (utilisation de l'antiquote pour renvoyer le résultat de la commande)

test

permet de tester une condition (existence ou type d'un fichier, comparaison de chaînes, comparaison de nombres. Renvoie un statut égal à 0 si l'expression est vraie, 1 si elle est fausse. Est utilisé dans les expressions conditionnelles, le plus souvent dans les scripts. Il est souvent remplacé par une autre syntaxe [<expression>]

Syntaxe : test <expressions>

Exemple : efface un fichier <fichier> s'il existe

if test –e <fichier> ; then rm <fichier> ; fi

Avec la seconde syntaxe, on aurait

If [-e <fichier>] ; then rm <fichier> ; fi

umask

définit l'umask (masque des droits par défaut lors de la création de fichier). Ne positionne jamais le bit de fichier exécutable sur les fichiers normaux.

Syntaxe : umask <masque>

Exemple : positionne le mode de création par défaut à 644

umask 022

Commandes externes

Les commandes externes sont en fait des fichiers exécutables indépendants du programme shell (sh, bash, ksh ou autres) et n'est font donc pas partie à proprement parler. Par abus de langage, on désigne souvent par shell la fenêtre de commandes en mode texte (comme une fenêtre msdos).

Les commandes qui vont être vues dans ce paragraphe sont les suivantes :

Cp, mkdir, rmdir, rm, ls, ps, mv, chown, chmod, chgrp, mount, umount, ln, tar, rpm, grep, find, nice, more, less, man, ifconfig, cat

Cp : copie de fichiers

Syntaxe : cp <source> <destination>

Si <destination> existe et est un répertoire, <source> est copié dans ce répertoire.

Pour plus d'information : man cp

Mkdir : création de répertoire (alias : md)

Syntaxe : mkdir <répertoire>

Rmdir : suppression de répertoire

Syntaxe : rmdir <répertoire>

Le répertoire doit être vide.

Rm : effacement de fichiers

Syntaxe : rm fichier

Ls : listage du contenu d'un répertoire

Syntaxe : ls –option motif

Options courantes : -a affiche tous les fichiers, y compris les fichiers cachés. –d affiche les répertoires comme les fichiers, et pas leur contenu. –l affiche au format long (droits, propriétaire, groupe…)

Ps : affichage des processus

Syntaxe : ps options

Pour plus d'infos : man ps

Mv : déplacement ou renommage de fichiers

Syntaxe :mv <source> <destination>

Si <destination> est un répertoire existant, <source> est déplacé dans <destination>, sinon <source> est renommé en <destination>

Pour plus d'infos : man mv

Chown : changement de propriétaire d'un fichier

Syntaxe : chown user[:group] fichier

Option –R : changement récursif de propriétaire

Chmod : changement des droits d'un fichier

Syntaxe : chmod mode fichier

Mode peut être en notation octale ou symbolique

Pour plus d'infos : man chmod

Chgrp : changement de groupe d'un fichier

Syntaxe :chgrp groupe fichier

Mount : montage d'un système de fichiers dans l'arborescence unix

Syntaxe : mount –t type périphérique répertoire

Où type est le type du système de fichiers, périphérique le nom du périphérique, répertoire le répertoire de montage.

Exemple : monter le système de fichiers windows se trouvant sur la première partition du disque maitre sur la nappe ide 1 au point de montage /mnt/windows :

mount –t vfat /dev/hda1 /mnt/windows

Pour plus d'infos : man mount

Umount : démontage d'un système de fichiers

Syntaxe : umount <périphérique> ou umount <point de montage>

On ne peut pas démonter un périphérique s'il est occupé (pas exemple si on utilise des fichiers sur ce périphérique)

Ln : création d'un lien physique ou symbolique

Syntaxe : ln –option <source> <destination>

Pour créer un lien symbolique, utiliser l'option -s

Tar : archivage – désarchivage

Syntaxe : voir man tar

Rpm : installation de paquetage logiciels au format RedHat

Syntaxe : rpm –type options paquetage

Type peut être : installation : i – mise à jour : U – désinstallation : e – interrogation : q

Pour plus d'infos : man rpm

Grep : recherche d'expression régulières

Syntaxe : grep –options <motif> [<fichier>]

Cherche le motif <motif> dans le fichier <fichier> (<motif> étant une expression régulière)

Grep est souvent utilisé à la suite d'un tube (voir redirections et tubes)

Find : recherche de fichier

Syntaxe extrêmement simplifiée : find <chemin> -name <motif>

Il y a beaucoup d'autres options. Voir man find

Exemple : trouver dans le répertoire courant les fichiers dont le nom commence par c :

find . –name 'c*'

Nice : définition de la priorité d'exécution d'un processus

Syntaxe :nice –n <valeur> <commande>

Lance la commande <commande> avec la priorité <valeur>

La priorité va de –20 (le plus prioritaire) à 19 (le moins prioritaire). Les priorités négatives (c'est à dire plus prioritaires) ne peuvent être définies que par le superutilisateur (root)

More : affichage page par page

Syntaxe :more <fichier>

Souvent utilisé à la suite d'un tube, auquel cas <fichier> n'est pas indiqué.

Less : affichage par page évolué

Syntaxe : less <fichier>

Souvent utilisé à la suite d'un tube, auquel cas <fichier> n'est pas indiqué.

Beaucoup d'option : voir man less

Man : pages de manuels

Syntaxe : man commande

Ifconfig : configuration IP

Permet de configurer les interfaces réseau. Pour la configuration : voir man ifconfig

Permet d'afficher la configuration des interfaces réseau :

Syntaxe :ifconfig [<interface>]

Si <interface> n'est pas fourni, affiche toutes les interfaces configurées.

Cat : affiche sur la sortie standard le contenu d'un ou plusieurs fichiers.

Syntaxe : cat [fichier] [fichier…]

Tableau de récapitulation

commande	Fonction	Équivalent DOS	Équivalent graphique
cd	Change de répertoire	Cd	Cliquer sur le répertoire dans un gestionnaire de fichiers
echo	Affiche à l'écran	Echo	-
exit	Sort du shell courant	Exit	-
kill	Tue un processus	-	Xkill
ulimit	Définit les ressources système	-	-
export	Exporte une variable	Set	-
pwd	Affiche le répertoire de travail	-	-
test	Teste une expression	-	-
cp	Copie de fichiers	Copy	Faire glisser dans un gestionnaire de fichiers
mkdir	Création de répertoire	Md	Commande dans le menu
rmdir	Suppression de répertoire	Rd	Commande dans le menu, touche suppr
rm	Efface un fichier	del	Commande dans le menu, touche suppr
ls	Listage du contenu du disque	Dir	Contenu affiché dans un un gestionnaire de fichiers
ps	Affiche les processus	-	Kpm – gtop…
mv	Déplace ou renomme un fichier	Ren	Faire glisser dans un gestionnaire de fichier, propriétés du fichier…
chown	Change le propriétaire d'un fichier	-	Propriétés du fichier
chmod	Change les droits d'un fichier	-	Propriétés du fichier
chgrp	Change le groupe d'un fichier	-	Propriétés du fichier
mount	Monte un système de fichiers	-	Certains gestionnaires de fichiers, certains utilitaires de configuration
umount	Démonte un système de fichiers	-	idem
ln	Crée un lien physique ou symbolique	-	Gestionnaires de fichiers
tar	Archive des fichiers	-	Ark, gzip
rpm	Installe des logiciels	-	Kpackage, rpmdrake…
grep	Recherche d'expressions régulières	-	Fonction incluse dans certains utilitaires de recherche de fichiers
find	Cherche un fichier	Find	Fonction incluse dans certains utilitaires de recherche de fichiers
nice	Définit la priorité d'exécution d'un processus	-	-
more	Affichage page par page	More	-
less	Affichage page par page évolué	-	-
man	Pages de manuel	-	Fris en charge par certains

			afficheurs d'aide
ifconfig	Configure les interfaces réseau	Ipconfig	Utilitaires de configuration
cat	Concatène et affiche des fichiers	?	-

Les variables

La notion de variable sous linux est la même que sous msdos. Les variables sont propres à un environnement. Si vous ouvrez deux consoles différentes, chacune bénéficie de son environnement propre. Les commandes lancées à partir d'un environnement héritent de cet environnement.

Souvent, les noms de variables sont en majuscules, mais ce n'est pas une obligation. Le contenu de la variable est référencé en faisant précéder le nom de la variable du signe $.

Exemple : je crée la variable TRUC et je lui donne la valeur « texte de truc » :

TRUC=''texte de truc''

J'affiche le contenu de la variable TRUC :

echo $TRUC (affiche : texte de truc)

par contre, echo TRUC affiche TRUC

Pour concaténer le contenu de variables ou de variables et de texte, il suffit de les juxtaposer.

Exemple : je crée la variable MACHIN et je lui donne la valeur « muche »

MACHIN=muche

J'affiche le contenu de la variable machin :

Echo $MACHIN (affiche : muche)

Je veux concaténer le contenu des variables : je définis une nouvelle variable MACHINTRUC et je lui affecte le contenu des deux variables TRUC et MACHIN :

MACHINTRUC=$TRUC$MACHIN

MACHINTRUC contient maintenant : texte de trucmuche

On le vérifie en faisant : echo $MACHINTRUC (affiche : texte de trucmuche)

On peut aussi faire : echo $TRUC$MACHIN (affiche : texte de trucmuche)

Pour afficher l'environnement courant et ses variables : printenv

Les variables définies lors d'une session ne sont pas conservées. Pour que les variables restent définies, il faut les redéfinir à chaque fois. C'est le rôle des fichiers de configuration.

Le path

Le path n'est qu'une variable comme les autres, qui est utilisée par le système pour trouver les fichiers exécutables. Elle se définit comme les autres variables. Par défaut sous linux (pour des raisons de sécurité), le répertoire courant n'est pas dans le path. Les commandes sont uniquement recherchées dans le path et pas ailleurs. Pour lancer une commande située dans un répertoire qui n'est pas dans le path, il faut donner le chemin d'accès complet à cette commande. Une commande située dans le répertoire courant de l'utilisateur ne s'exécutera pas si ce répertoire n'est pas dans le path. Il faudra donner le chemin d'accès (soit absolu, soit relatif). Par exemple, j'ai créé une commande dans mon répertoire. Pour la lancer, je dois taper : ./nom_de_commande (chemin relatif) ou /home/chartier/nom_de_commande (chemin absolu).

On peut utiliser aussi la variable prédéfinie $HOME (qui désigne le répertoire de travail de l'utilisateur – voir commande pwd) ou le tilde ~ pour référencer le chemin. Par exemple, si la commande est située dans mon répertoire utilisateur : ~/nom_de_commande ou $HOME/nom_de_commande.

Il est possible de mettre le répertoire courant dans le path : il suffit d'y rajouter un point (.) qui désigne, comme sous msdos, le répertoire courant (les deux points .. désignant le répertoire de niveau supérieur, comme sous msdos également). C'est toutefois déconseillé de procéder ainsi.

Comme pour les autres variables, le path n'est pas conservé. Il faut donc le définir dans un fichier de configuration qui sera lu à chaque démarrage de la machine ou à chaque login utilisateur.

Redirections et tubes

Ces notions existaient dans msdos, rien de nouveau donc.

Redirection en sortie : caractère >

Redirection en entrée : caractère <

Ajout : >>

Tube : |

Quelques exemples :

Écrire le contenu du répertoire courant dans le fichier « listrep » :

ls > listrep

y rajouter le contenu du répertoire /root :

ls /root >>listrep

rechercher dans la liste des fichiers du répertoire courant le motif « test »

ls | grep test

recherche si un fichier <fichier> contient le mot « test »

cat <fichier> |grep test

trie le contenu d'un fichier « fichier » en l'écrit dans « fichier_trié »

cat fichier | sort > fichier_trié

crée un fichier contenant une phrase :

echo « ceci est une phrase » > fichier

Complétion

Le shell bash permet la complétion automatique des noms de fichiers ou de commandes. Il est possible d'utiliser d'autres complètements (nom d'utilisateurs…).

Pour cela, c'est simple : tapez le début d'une commande, puis la touche tabulation : l'ensemble des commandes commençant par ces lettres est affiché.

Pour les chemins, c'est pareil : taper le début du chemin, puis la touche tab : il vous est proposé les chemins correspondants. Le choix peut être affiné en rajoutant des caractères. S'il n'y a qu'un choix possible, la ligne de commande est automatiquement complétée.

A l'usage, c'est extrêmement rapide. Ca permet aussi de retrouver des commandes dont on ne connaît que le début.

Les scripts, structures conditionnelles, boucles

Le shell n'aurait qu'un intérêt moindre s'il n'offrait que la possibilité de taper des commandes dans une console. Mais il est possible de regrouper des commandes dans un fichier appelé script, qui est la même chose que les fichiers batch (.bat ou .cmd) sous dos/windows.

Le shell constitue en fait un langage de programmation permettant des comparaisons, tests, structures conditionnelles et boucles.

Les tests sont nombreux. 24 tests concernent les fichiers, 6 concernent les chaines de caractères, 6 concernent les opérateurs arithmétiques.

Autres opérateurs : négation, et, ou

Structures conditionnelles :

Structure si alors sinon (if else elif fi)

Structure de cas (case esac)

Structures de boucles :

Structure pour (for do done)

Structure tant que (while do done)

Structure répète (until do done)

Les variables peuvent être utilisées.

Les paramètres de lancement du script peuvent être récupérés dans le script : ils vont de $0 à $9. $0 désigne le nom de la commande, $1 le premier paramètre, etc. la variable $# désigne le nombre de paramètres sur la ligne de commande, $@ l'ensemble des paramètres $1 à $n.

Le script doit être exécutable : lui donner l'attribut d'exécution (chmod +x script).

Pour que le script puisse être interprété directement par le shell, mettre en première ligne la commande de lancement du shell comme ceci :

!/bin/bash

Enchaînement des commandes inconditionnelles et conditionnelles

Les commandes peuvent être enchainées.

Enchaînement inconditionnel :

commande*1* ; commande*2* ; …. ; commande*n*

ou dans un script

commande*1*

commande*2*

…

commande*n*

Enchaînement conditionnel :

Commande1 && commande2 : commande2 n'est exécutée que si commande1 a réussi (code de retour égal à 0)

Commande1 || commande2 : commande 2 n'est exécutée que si commande1 a échoué.

Exemple : n'effacer un fichier que s'il existe

[-e fichier] && rm fichier

On aurait pu écrire : test –e fichier && rm fichier

Les fichiers de configuration

Contrairement à windows depuis W95, la configuration d'un système linux repose quasi totalement sur des fichiers texte (comme les fichiers .ini dans windows 3.1) et non sur une base de registre.

La modification de la configuration peut donc se faire avec un simple éditeur de texte. Mais elle peut se faire également par l'application elle-même (qui va écrire ses fichiers de configuration) ou par des utilitaires graphiques ou non de configuration.

Configuration système

La configuration globale du système se situe dans le répertoire /etc

Ce répertoire contient en général d'autres répertoires pour chaque application.

Configuration utilisateur

La configuration utilisateur est placée dans le répertoire utilisateur (correspond à la branche HKCU du registre). Ces fichiers sont en général cachés (leur nom commence par un point). On peut également trouver une structure arborescente.

Fin de la deuxième partie

Ce document est un survol du sujet et ne peut remplacer l'apprentissage des commandes et du système.

Support formation de base à linux

Troisème partie : environnement graphique, services, accès distant

l'environnement graphique, le serveur X, les windows managers, les services, l'accès distant, l'administration distante, l'installation des programmes

L'environnement graphique

L'environnement graphique sous linux lui offre une interface similaire à Windows : utilisation de la souris, des boutons, menus déroulants…

Cet environnement n'est pas indispensable pour l'utilisation d'une machine linux en tant que serveur. Il n'est pas même souhaitable d'utiliser un environnement graphique dans ce cas.

En effet, les fonctionnalités d'une interface graphique requièrent des ressources système importantes qui sont tout à fait inutiles pour une machine dont l'écran n'est même pas allumé en usage normal.

D'autre part, les tâches à effectuer sur un serveur sont généralement des tâches d'administration système (nécessitant donc les droits correspondants) pour lesquelles l'interface graphique, si elle est lancée avec les droits d'administrateur (root), provoque inutilement un risque potentiel de sécurité, en raison de la complexité de l'interface graphique et des droits élevés d'accès au matériel qu'elle comporte. Il est donc plus sûr d'utiliser une interface en mode texte, ou en mode web.

Il reste cependant possible avec moins de risques d'utiliser une session graphique en tant qu'utilisateur normal et d'ouvrir au sein de cette session une console en tant qu'administrateur, permettant éventuellement de lancer des programmes graphiques d'administration système. Ceci pour les administrateurs ayant pris la mauvaise habitude des interfaces graphiques…

Par contre, pour l'usage en tant que poste de travail (bureautique, graphisme…), l'interface graphique est indispensable.

Le mode graphique est constitué par une couche supplémentaire ajoutée au système d'exploitation, totalement indépendante de celui-ci. Cela signifie qu'il n'est pas nécessaire d'installer le mode graphique pour le fonctionnement d'une machine linux (contrairement à windows depuis W95 ou d'autres systèmes tels QNX où l'interface graphique (photon) fait partie du système).

Le mode graphique est le plus souvent implémenté à l'aide d'un serveur graphique (serveur X) qui offre une API de bas niveau pour les fonctions graphiques. Le serveur X est en fait une interface entre les fonctions de l'API et le matériel.

Une couche supplémentaire est ajoutée au serveur X pour offrir des fonctions de plus haut niveau : c'est le Window Manager (KDE, Gnome, IceWm…). C'est cette couche supplémentaire qui offre l'aspect de l'interface graphique utilisateur. La multiplicité des window manager fait qu'un système linux peut avoir des aspects très différents. Chaque WM s'appuie sur une bibliothèque graphique (Qt pour KDE, GTK pour Gnome). Ces bibliothèques sont souvent installées de manière à pourvoir utiliser les applications qui y font appel. Il y en a d'autres (fox par exemple, qui offre un aspect visuel très proche de windows). L'aspect des applications graphiques dépend pour une bonne part des bibliothèques utilisées et n'est donc pas uniforme, contrairement à windows où la plupart des applications utilise l'API graphique standard de Windows (GDI). On trouve cependant également sous windows des API graphiques différentes (GTK et Qt existent également sous Windows).

Les services

Un service est un programme tournant en tâche de fond (il n'est pas relié à une interface tournant en avant plan et ne reçoit pas d'entrée du clavier et n'affiche rien à l'écran).

Sous linux/unix, les services sont le plus souvent appelés démons (de l'anglais daemon, **d**isk **a**nd **e**xecution **mon**itor).

Un service, comme son nom l'indique, offre un service à l'utilisateur ou au système. On y accède généralement par une socket (association d'une adresse IP et d'un numéro de port).

Les services sont à l'écoute de leur port pour pouvoir répondre aux requêtes qui leur sont adressées (état LISTEN). Les associations entre numéro de port et nom du service figurent dans le fichier /etc/services.

Par exemple, le serveur web Apache (httpd) écoute le port 80 en attente de requête http.

Souvent, le nom des exécutables des services se termine par la lettre d (comme démon).

Les services étant des programmes chargés en mémoire, ils consomment des ressosurces système.

Pour éviter une charge trop importante du système pour des services dont l'usage n'est pas permanent, il existe un « super-serveur » nommé inetd ou xinetd pour les distributions récentes, dont le but est d'écouter les ports des services mentionnés dans son fichier de configuration, et de lancer effectivement le service quand une requête lui est adressée (chargement dynamique des services). L'avantage est la moindre charge du système, l'inconvénient est le temps de chargement du service. A réserver donc pour les services non permanents ou pour ceux dont le temps de chargement est faible.

Un avantage important du super-serveur xinetd est sa faculté de filtrage d'accès.

Accès distant

Un bon système serveur est un système capable de tourner 24/24 dans son coin sans nécessiter d'intervention. Mais même le meilleur des systèmes nécessite cependant d'être administré, et de pouvoir l'être à distance, l'administrateur n'étant pas nécessairement sur le lieu géographique d'installation du serveur.

Linux possède en standard un certain nombre de services d'accès distant.

Il y a bien sûr telnet, mais sa sécurité est très faible et il n'est pas concevable de l'utiliser dans un environnement peu sûr. On lui préfèrera dans tous les cas ssh, qui permet le login sécurisé sur une machine distante. On peut alors travailler comme si on était sur la machine locale, mais en mode texte. Cela suppose la connaissance des commandes unix.

L'avantage est la rapidité (sur une liaison lente) et l'universalité.

Mais les administrateurs habitués à une interface graphique trouveront peut-être difficile de se retrouver devant un écran noir, comme au bon vieux temps !

D'autres solutions existent donc pour avoir un accès graphique à un serveur :

- Login graphique distant (protocole XDMCP qui utilise udp et ne peut donc pas utiliser un canal ssh)
- Export de l'affichage (qui peut utiliser un canal ssh)
- Interface web (en http ou https, préférable pour des raisons de sécurité)

Les deux premiers cas consistent à déporter sur une autre machine l'affichage et la saisie. Les données d'affichage étant assez lourdes, ces solutions sont inadaptées aux liaisons lentes. Le client doit de plus être sous linux ou avoir un client X pour windows.

Le troisième cas (accès Web) est le plus universel (il suffit d'un navigateur), mais il requiert un serveur des ressources désirées sur le serveur.

Pour l'administration, on pourra utiliser webmin (https), qui dispose de modules pour beaucoup de tâches de configuration, swat (http), pour l'administration du serveur samba, ou cups (http), pour l'administration du serveur d'impression.

Ces outils consistent en des serveurs web spécifiques sur la machine, lancés en tant que services.

Il est certain que l'utilisation d'une interface graphique est beaucoup plus facile et intuitive qu'une interface texte. Mais celle-ci n'est pas limitée aux seules fonctions implémentées, puisqu'elle offre un accès complet au système, elle fonctionne correctement même sur des liaisons à faible débit, et elle permet l'utilisation de ce qui fait la puissance d'un système unix, les scripts.

Malgré son côté austère, on privilégiera l'interface texte autant que possible, pour sa puissance et sa sécurité.

Installation de programmes

S'il y a bien un domaine qui donne lieu à discussions, notamment aux habitués du cliquodrome à la InstallShield, c'est l'installation des programmes !

Traditionnellement, sous linux, les programmes sont installés à partir des sources, par compilation. Cela permet :

- Une relative sécurité due au fait qu'il est possible de consulter le code source et donc de vérifier l'absence de fonction cachée, la disponibilité du code source étant en elle-même garante d'une certaine sécurité (si nous on n'y comprend rien, d'autres comprennent le C dans le texte !)
- Une adaptation fine du binaire à la plate-forme d'exécution
- La possibilité, par les options de compilation, d'adapter le produit à l'usage requis

Les inconvénients étant : la nécessité d'avoir sur la machine un compilateur ainsi que les librairies de développement, le temps de compilation, la difficulté de désinstallation.

Certains éditeurs ou développeurs ont donc conçu des programmes permettant l'installation de binaires. Ces binaires ainsi que les fichiers nécessaires sont groupés dans un fichier compressé nommé package, qui contient également les informations sur les dépendances (programmes ou bibliothèques nécessaires au programme).

Il y a actuellement deux formats principaux de package : le format RedHat (rpm), le format Debian (deb).

Chaque format a ses utilitaires dédiés. L'éditeur MandrakeSoft a développé un outil spécifique basé sur les packages RedHat pour faciliter les opérations d'installation, notamment la gestion automatique des dépendances (programme urpmi). Néammoins, chacun s'accorde à reconnaître que le format Debian est plus performant. Mais il n'est utilisable que sur la distribution Debian, le format RedHat étant utilisable sur la majorité des distributions (RedHat, Mandrake, Suse, Conectiva…)

Il existe aussi comme sous windows des programmes d'installation totalement automatiques, comme sous windows. Par exemple les programmes d'installation de Acrobat Reader, Mozilla, Netscape, OpenOffice. Il suffit de cliquer !

Fin de la troisième partie

Ce document est un survol du sujet et ne peut remplacer l'apprentissage des commandes et du système.

Prochaine partie : authentification, samba (implémentation linux du protocole smb)

Support formation de base à linux

Quatrième partie : authentification, serveur samba

L'authentification sous linux, PAM, le serveur Samba, autres systèmes de fichiers exportés

Authentification

Ce paragraphe présente de manière extrêmement sommaire l'authentification des utilisateurs sur un système linux.

L'authentification se fait traditionnellement par le biais d'un fichier contenant les utilisateurs et leur mot de passe, ainsi que les groupes auxquels ils appartiennent. Par la suite, les mots de passe ont été déplacés dans un autre fichier, pour des raisons de sécurité.

L'authentification peut se faire également sur une base NIS (yellows pages), mais aussi par d'autres moyens comme LDAP ou même un DC NT.

Pour permettre l'authentification par n'importe quel moyen, une nouvelle infrastructure de gestion des utilisateurs/mots de passe a été définie. Elle permet l'abstraction du système d'authentification pour les applications. Une requête d'authentification sera effectuée par l'application au système, qui va se charger, suivant la configuration de l'authentification, de retourner le résultat de la requête. Le système qui permet cette abstraction est PAM (pluggable authentification modules).

Le serveur Samba

Samba est l'implémentation unix/linux du protocole windows smb/cifs (server message block / common internet file system, également appelé LanManager ou Netbios protocol). Ce protocole (netbios over IP) utilise les ports 137 (name service), 138 (datagram service) et 139 (session service).

Cette implémentation ne fonctionne que si tcp/ip est installé (le protocole implémenté est netbios over tcp/ip - nbt). Il ne fonctionne pas avec d'autres protocoles de la couche réseau comme netbeui (netbios over llc) ou netware (netbios over ipx)

Le serveur samba se compose en fait de deux serveurs distincts : le serveur de nom netbios nmbd et le serveur de ressources smbd.

Samba offre les mêmes fonctionnalités que le serveur lancé par windows quand on choisit le partage de fichiers et d'imprimantes sous windows 9x. Il offre également d'autres possibilités plus proches de windows NT : il peut être contrôleur principal de domaine (mais pas encore contrôleur secondaire), maître explorateur du réseau…

Les différents tests effectués ont montré qu'un serveur samba était plus performant qu'un serveur windows.

Il est donc tout à fait envisageable s'utiliser une machine linux avec le serveur samba pour offrir à un réseau windows des ressources sur un serveur, avec l'avantage de la stabilité de linux et son faible coût.

Ce qui fonctionne

Logon dans le domaine pour les clients NT4, et pour 2000 et XP en mode de compatibilité

Logon des clients 9x

Profils

Ressources fichiers et imprimantes

Serveur wins

Ce qui ne fonctionne pas

Relation d'approbation entre domaines (domain trusts)

Réplication de la base SAM avec un DC windows

Ajout d'utilisateurs via user manager for domains

Contrôleur windows 2000 (kerberos et active directory)

Configuration

L'ensemble de la configuration de samba tient dans un seul fichier texte nommé smb.conf.

Ce fichier a une structure du même type que les fichiers .ini de windows 3.1.

Il est composé de plusieurs sections :

Section [global] : options générales de configuration

Section [homes] : permet la configuration automatique de répertoire utilisateur lors du logon

Section [printers] : imprimantes partagées

Toutes les autres sections sont des noms de partage. Par exemple : [public], [netlogon], [print$]…

Tous les termes du fichier de configuration sont en anglais. Si on le comprend, c'est beaucoup plus facile…

Section [global]

Workgroup : définit le nom de workgroup ou de domaine

Server string : description du serveur

Netbios name : nom netbios du serveur

Security : niveau de sécurité (share, user, domain, server)

Encrypt passwords : active ou non le cryptage des mots de passe

Character set : jeu de caractères à utiliser

Section [homes]

Comment : description du partage

Read only : partage en lecture seule ou non

Browseable : ce partage apparaît ou non dans le voisinage réseau

Section [printers]

Comment : description du partage

Path : chemin d'accès au répertoire de spool

Guest ok : un invité (client non identifié) peut imprimer

Printable : si on veut pouvoir imprimer…

Print command : commande d'impression à utiliser par le serveur

Lpq command : commande d'affichage de l'état des imprimantes

Lprm command : commande de suppression d'un travail d'impression

Browseable : cette imprimante apparaît ou non dans le voisinage réseau

Section [nom_de_partage]

Comment : description du partage

Path : chemin d'accès au répertoire correspondant à ce partage

Guest account : définit le compte utilisateur (avec les droits correspondants) pour l'accès en mode invité

Guest ok : autorise l'accès en mode invité

Read only : accès en lecture ou écriture

Hosts allow : liste des hôtes autorisés à accéder à ce partage

Hosts deny : liste des hôtes non autorisés à accéder à ce partage

Browseable : ce partage apparaît ou non dans le voisinage réseau

Available : ce partage est-il accessible ? (permet de désactiver un partage sans le supprimer)

Il y a encore une quantité d'autres options de configuration permettant une configuration fine du serveur.

Notamment, si le serveur samba est contrôleur de domaine, les paramètres suivants seront utilisés (section [globals]) :

Adduser script : script permettant l'ajout d'un utilisateur

Delete user script : script permettant la suppression d'un utilisateur

Logon script : script de logon

Logon path , logon drive, logon home : chemins d'accès aux ressources du logon

Domain logon : oui ou non

Dans tous ces paramètres, certaines variables prédéfinies peuvent être utilisées (nom du serveur, nom de l'utilisateur…), ce qui permet de personnaliser chaque ressource pour chaque utilisateur (scripts de démarrage par exemple).

On peut également paramétrer le comportement du serveur en ce qui concerne le parcours du réseau (maitre explorateur du domaine, maitre local…)

L'utilisation de samba en tant que contrôleur de domaine nécessite la création d'un partage nommé netlogon, non public, caché et en lecture seule (comme sous windows).

Tous ces paramètres peuvent être modifiés à l'aide d'interfaces graphiques d'administration.

Administration de Samba : swat, webmin

On peut administrer son serveur samba localement ou à distance avec swat ou webmin, qui sont des interfaces web d'administration. Swat est l'interface « native » fournie avec samba, webmin contient un module samba.

Ce qu'il faut savoir

Portée des paramètres

Un certain nombre de paramètres sont définis dans la section [global]. Si ce paramètre n'est pas surchargé dans une autre section, ce sont les paramètres globaux qui sont appliqués. Si les paramètres globaux ne sont pas définis, c'est la valeur par défaut qui est utilisée.

Valeur par défaut → paramètres de la section [global] → paramètres de la section [section]

Il faut donc veiller à bien définir les paramètres globaux pour éviter d'avoir à les redéfinir pour chaque partage. Les paramètres des partages ne devraient contenir que les paramètres propres au partage et différents de la valeur globale.

Utilisateurs

Les utilisateurs doivent exister sur le serveur linux ou pour le moins doivent pouvoir être trouvés par lui.

Ils doivent également figurer dans la base des utilisateurs de samba.

Il y a (dans la version actuelle) nécessité d'avoir deux bases d'utilisateurs : celle du système et celle de samba. Les mots de passe peuvent (et devraient) être différents. Il est toutefois possible, lorsque l'authentification a été déléguée à un serveur NT, de créer à la volée les compte unix lors du premier login. Une fois l'authentification faite sur le contrôleur NT, samba peut créer le compte unix local correspondant. Idem pour la suppression.

Pour les clients NT et assimilés pour lesquels existe un compte machine, il est également nécessaire de créer un compte pour cette machine (dont le nom se termine par $) au niveau du système linux et de samba.

Par sécurité, les utilisateurs de samba se verront attribuer un shell nul pour qu'ils ne puissent pas se loguer directement sur le serveur linux.

Les commandes à connaître :

Ajouter un utilisateur (au système) : adduser <utilisateur>

Par défaut, le compte unix correspondant est désactivé.

On emploiera certaines options dans le cas de la création d'un utilisateur pour samba :

adduser -g <nom de groupe> -c <commentaire> -s /bin/false –n <nom utilisateur>

ajouter un utilisateur à samba :

smbpasswd –a <nom utilisateur>

supprimer un utilisateur du système :

userdel <nom utilisateur> (utiliser l'option –r pour supprimer le répertoire et les fichiers de l'utilisateur)

supprimer un utilisateur de samba :

smbpasswd –x <nom utilisateur>

Il est possible d'écrire un petit script réalisant ces deux opérations en même temps.

Ces commandes peuvent être lancées à distance via ssh.

Droits sur les fichiers

Les fichiers étant physiquement situés sur un système de fichiers unix, les droits définis au niveau du système existent et ne sont pas outrepassés par samba. En d'autres termes, si un fichier est en lecture seule pour l'utilisateur, il le sera aussi par un partage samba.

Samba permet de définir des masques de droits lors de la création d'un fichier via samba :

Create mask : les droits n'étant pas gérés sur des systèmes de fichiers fat/fat32, la création d'un fichier à partir d'un client windows va conduire à attribuer des droits unix à ce fichier. Par défaut, le masque 0744 est appliqué, ce qui signifie que le propriétaire a tous les droits sur son fichier (chiffre 7), mais que le groupe et les autres n'ont que le droit de lecture (chiffre 4).

Directory mask : idem mais pour les répertoires. Par défaut, le masque est de 0755 : le groupe et les autres peuvent lister le répertoire et y entrer (chiffre 5 : 4+1).

Il est possible également de définir les droits par héritage : c'est le paramètre inherit permissions. Les répertoires héritent des droits du répertoire de niveau supérieur, les fichiers des attributs lecture et écriture. Le bit sgid est aussi hérité, mais jamais le suid. Par défaut, cette fonctionnalité est désactivée.

Les attributs fat/fat32 système, caché et archive peuvent être « mappés » sur l'attribut exécution unix. C'est le rôle des paramètres map archive, map system, map hidden. Ces mappages ne sont actifs que si les bits exécution (valeur impaire de l'attribut correspondant en octal) sont respectivement positionnés pour l'utilisateur, le groupe ou les autres.

En lecture, les droits unix sont mappés ainsi : un fichier unix caché aura l'attribut caché (si le paramètre hide dot file est mis à yes), un fichier en lecture seule sera en lecture seule, un fichier exécutable aura l'attribut archive.

Il est possible de forcer l'utilisateur ou le groupe lors de la création de fichiers/répertoires. Cela peut être utile pour des partages publics où l'on souhaiterait que les fichiers et répertoires créés n'appartiennent pas à celui qui les a créé, mais a un utilisateur unique représentant l'ensemble des utilisateurs (même dans un partage public, un fichier créé par un utilisateur lui appartient ! Il a par défaut les droits définis par le create mask, ce qui veut dire que le fichier n'est pas accessible aux autres personnes en écriture !). Les paramètres force group et force user permettent de forcer le groupe et le propriétaire des fichiers créés.

Cette méthode est préférable à l'emploi d'un compte invité, qui autoriserait n'importe qui à créer (ou supprimer) des fichiers (l'accès invité est à éviter, surtout en écriture). Il serait aussi possible de positionner le create mask à 0764, autorisant ainsi le groupe à accéder en écriture, mais il faut alors que tous les utilisateurs appartiennent à un même groupe.

Ceci dit, ce n'est pas forcément une bonne idée de permettre l'écriture (et donc la suppression) de fichiers par d'autres personnes que leur propriétaire !

Les paramètres remplaçables (macros) dans smb.conf

Un certain nombre de macros (paramètres remplaçables) sont utilisables dans le fichier de configuration, permettant ainsi des actions personnalisées. Les principales sont les suivantes :

%u : nom d'utilisateur du service

%U : nom de l'utilisateur de la session

%m : nom netbios de la machine cliente

%I : adresse IP de la machine cliente

%g et %G : nom de groupe principal de %u et %U respectivement

%a : architecture du client (valeur peu sûre, un client W98 renvoie win95 !)

%H : répertoire home de %u

%T : date et heure du serveur

%L : nom netbios du serveur

%$(variable) : valeur de la variable d'environnement « variable »

%v : version de samba

Les paramètres %m et %u sont souvent utilisés pour les scripts de login personnalisés en fonction de la machine ou de l'utilisateur. Ces paramètres sont utilisables dans les chemins.

Fin de la quatrième partie

Ce document est un survol du sujet et ne peut remplacer l'apprentissage des commandes et du système.

Prochaine partie : à définir selon les besoins