MODULE 01 : Introduction aux systèmes d'exploitation

Le Système d'Exploitation (Operating System) est l'ensemble des logiciels effectuant la **gestion optimale des ressources** d'un système informatique. Installé (on dit encore "chargé") en **mémoire centrale** lors du démarrage, le Système d'exploitation prend en charge la gestion complexe des constituants d'un ordinateur (processeur, mémoire, périphériques, etc.) et en optimise l'utilisation.
Il permet tout simplement de rendre opérationnel un ordinateur.

Les fonctions principales sont:

- gestion des travaux (partage de l'UC),
- gestion de la mémoire centrale,
- gestion de la mémoire secondaire,
- gestion des utilisateurs (notion de sécurité).

Un système d'exploitation possède au moins un interpréteur de commande et offre un certain nombre de services: éditeur de texte, compilateur, etc.

1. Définitions.

Un processeur est l'ensemble des moyens matériels et logiciels permettant l'exécution des instructions.

Un processus est le déroulement dynamique d'un ensemble d'instructions exécutables sur le même processeur. Cette définition s'applique également aux **Threads.**

Un programme est un ensemble de processus, éventuellement réduit à un seul élément.

Une ressource est tout moyen logiciel ou matériel nécessaire au lancement d'un processus, autre qu'un processeur. Les ressources peuvent être une zone de mémoire centrale, de la mémoire secondaire ou un périphérique (dérouleur de bandes, imprimante, etc).

2. Processus, notion de multi-tâches.

Un processeur, plusieurs processus ➔ nécessité de multitâches

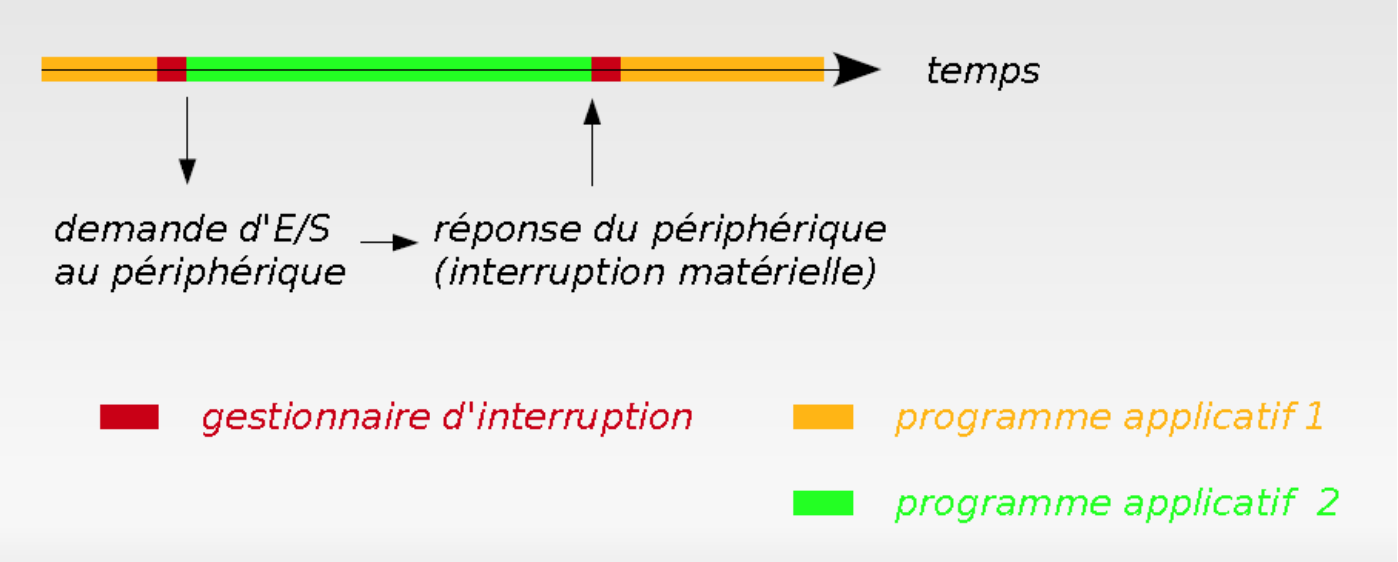
Multitâche = Gestion de l'attribution du processeur aux processus.

Un processus peut prendre plusieurs états :

- **Actif** : Il dispose de toutes les ressources nécessaires et d'un processeur adéquat
- **Activable** : Il lui manque un processeur.
- **Bloqué ou en attente de ressource** : Il lui manque une ressource.

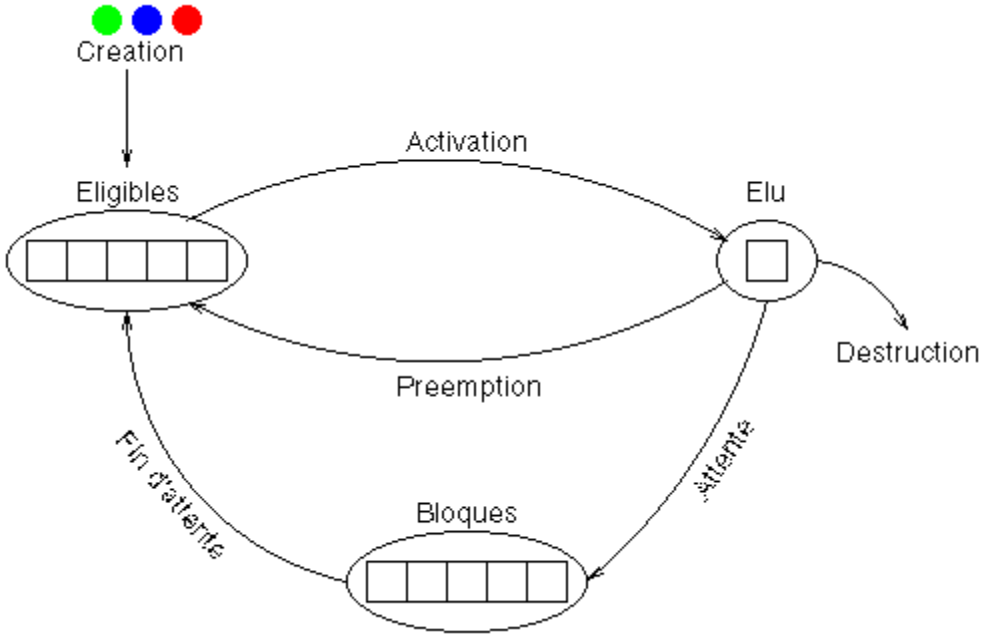
Méthodes d'allocation du processeur par les processus :

- **Préemptive** (***premptive scheduling***): C'est le système d'exploitation qui "commande" et donne ou retire le processeur aux processus (Système d'exploitations actuels)
- **Non préemptive** (***collaborating scheduling***) : Chaque processus rend le processeur "à sa guise". (Windows 3).



Méthode du tourniquet ou round robin ou balayage cyclique:

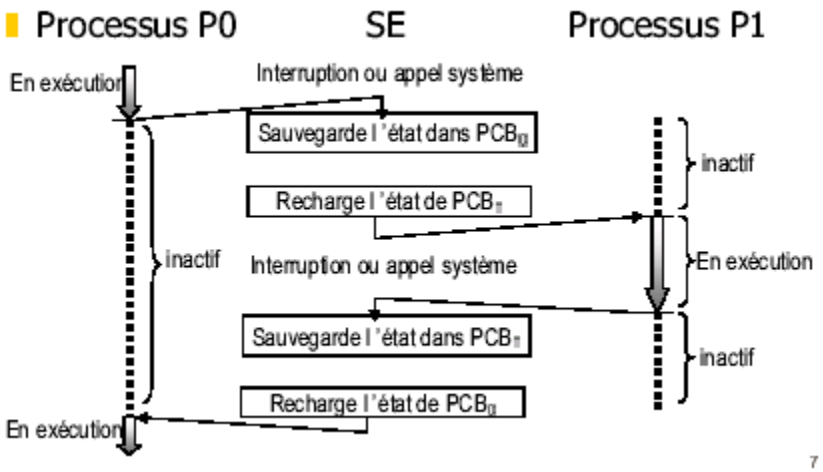
Chaque processus accède au processeur chacun à son tour, pour un temps fixé (le quantum).



Méthode du tourniquet multiniveaux

Avant d'accéder au processeur, les processus sont rangés dans les files correspondant à leur **niveau de priorité**. Un processus ne peut accéder au processeur que s'il n'existe plus de processus dans les files de plus haute priorité.

Commutation de l'UC entre processus :



PCB = Process Control Bloc : contient toutes les informations nécessaires à la gestion du processus.

3. Mémoire virtuelle.

Mémoire virtuelle = support de l'ensemble des informations potentiellement accessibles.

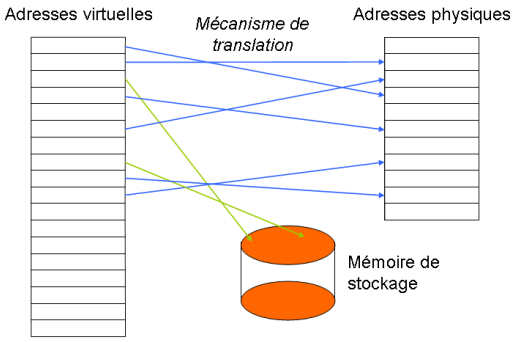
= Ensemble des emplacements dont l'adresse peut être engendrée par le processeur.

≠ Mémoire physique = Ensemble des emplacements RAM physiquement présents dans l'ordinateur.

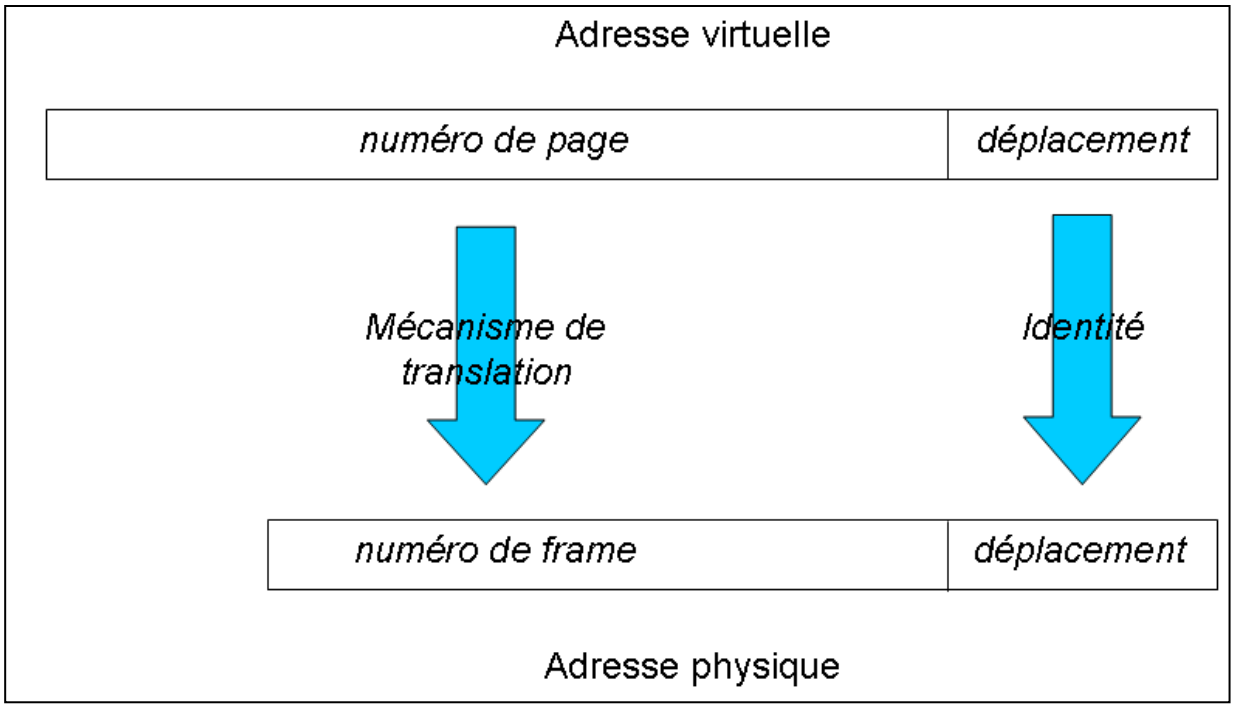
Objectif : **Utiliser la mémoire secondaire** (= disque) pour réduire la quantité de mémoire vive coûteuse.

Principe général

Il s'agit de conserver en mémoire **une "partie" des programmes et des données** en cours d'exécution.
Si un programme A veut s'exécuter alors qu'il n'y a plus de place en mémoire, un "bout" d'un autre programme est "viré" en mémoire secondaire et remplacé par un "bout" de A.
Donc, un programme est découpé en bouts que l'on nomme **pages**, de taille fixe. La mémoire physique est elle aussi découpée en pages, de même taille, ainsi que la mémoire secondaire.



Une fonction du processeur appelé **MMU (Memory Management Unit)** est capable de traduire une adresse virtuelle en adresse réelle à l'aide de la **table des pages mémoires**. Cette table contient également l'information de l'emplacement et de la validité de la page mémoire.



Le déplacement correspond à l'emplacement mémoire dans la page (une page contient plusieurs cellules mémoire).

Lorsqu'un programme demande une case mémoire, il connaît l'adresse de la mémoire virtuelle.
3 cas :

- **La case mémoire correspondante est présente** : l'adresse est convertie en adresse physique
- **La case mémoire correspondante n'est pas présente mais il reste de la mémoire physique libre :** Le processus va être mis en attente de ressource, le temps que le système charge la page voulue.
- **La case mémoire correspondante n'est pas présente et la mémoire physique est entièrement utilisée** (=**défaut de page**).**:**Il faut d'abord décharger une autre page vers la mémoire secondaire. la page sera choisie suivant un certain nombre de critères comme l'ordre d'arrivée, la page la moins utilisée, une page non modifiée (inutile de la réécrire sur le disque),…

Un système à mémoire virtuelle nécessite donc à la fois un processeur prévu pour et un système d'exploitation capable de gérer cette possibilité.

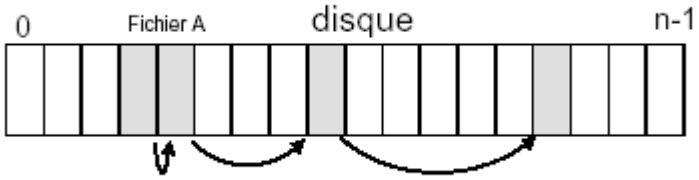
4. Gestion des fichiers.

Un **fichier** est une unité de stockage logique de l'information.
Le **disque** est l'unité physique, qui comporte un certain nombre de blocs d'informations.
Il est nécessaire d'avoir une méthode d'allocation des blocs aux fichiers :

- **Allocation contiguë** : Chaque fichier occupe un nombre de blocs contigus sur le disque

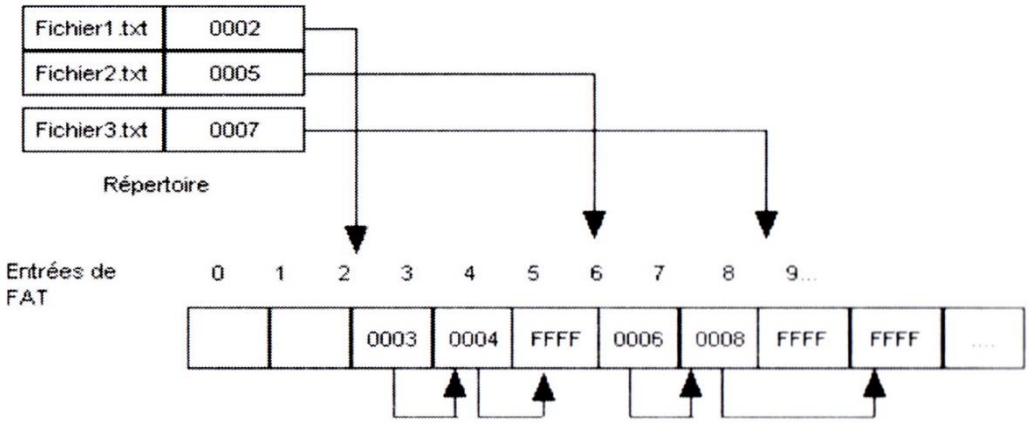
Avantages : Simple à implémenter, accès direct possible, Adapté aux supports "Write once" (Principe du format ISO9660 des CDROM).
Inconvénients : Impossible d'augmenter la taille d'un fichier; perte de place sur les supports à écriture multiple (création de "trous" inutilisables).

- **Allocation chaînée** : Chaque fichier occupe une liste chaînée de blocs sur le disque.



Chaque bloc contient des données + un pointeur vers le bloc suivant
Avantages : Possibilité d'étendre un fichier, tout bloc peut être alloué
Inconvénients : Accès direct impossible, il faut suivre la chaîne

- **Allocation chaînée et indéxée** : On sépare les données des pointeurs ➔ on créée une table d'allocation (FAT).



Avantages : Extension des fichiers, accès direct possible, les blocs ne contiennent pas de pointeurs
Inconvénients : La FAT doit résider en mémoire centrale ce qui pose problème pour des disques de grande capacité. (1Go en bloc de 1Ko occupe 4Mo).

Ce mode est principalement utilisé pour les supports amovibles. On distingue :

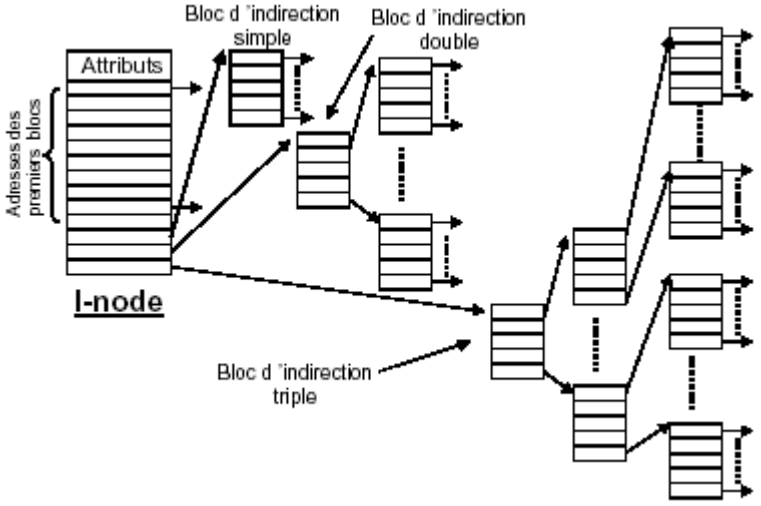
- **FAT16** : Adresses sur 16 bits soit 65536 blocs possibles donc au maximum (bloc de 64ko), volume de 4Go.
- **FAT32 :** Adresses sur 28 bits –4 bits non utilisés– soit 268 425 456 blocs possibles donc au maximum (bloc de 32ko) en théorie volume de 8To. Ce maximum est limité à 2To.De plus, la taille d'un fichier est limitée à 4Go.
- **ExFAT (Extended Fat) :** Introduit en 2007 par Microsoft pour résoudre les problèmes de fichiers et volumes larges dans des environnements sans NTFS.

Adresses sur 64 bits, soit en théorie 64 Zo (1021), supports des fichiers jusqu'à 16Eo.

- **Allocation par nœud d'information inode** :

  Les **inodes** (contraction de « index » et « node »; en français, nœud d'index) sont des structures de données qui contiennent des informations à propos des fichiers stockés dans les systèmes de fichiers de type Linux/Unix. À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé.

  Chaque inode contient entre autres un certain nombre d'adresses de blocs de données (en général 10) permettant d'accéder directement aux données du fichier. Si le fichier est plus important, 3 autres champs renvoient sur des blocs d'adresses (**bloc d'indirection**) qui eux mêmes renvoient soit sur des blocs de données (**indirection simple**) soit vers des blocs d'adresses (**indirection double ou triple**).



  Avantages : Economie de mémoire. Seuls les inodes des fichiers ouverts sont montés en mémoire.
  Les blocs d'indirections sont montés en mémoire à la demande.
  Par contre cela oblige à gérer en parallèle la liste des blocs libres puisqu'on ne dispose plus d'une carte complète du disque.

  Les systèmes de fichier utilisés par Linux sont basés sur ce principe :
  - **Ext 2** : ou Ext2fs (en anglais, *second extended file system*) est le système de fichiers historique de GNU/Linux.
  - **Ext 3** : Même système avec l'utilisation d'un fichier journal qui permet de conserver une trace des dernières modifications pour une reprise sur incident plus rapide.

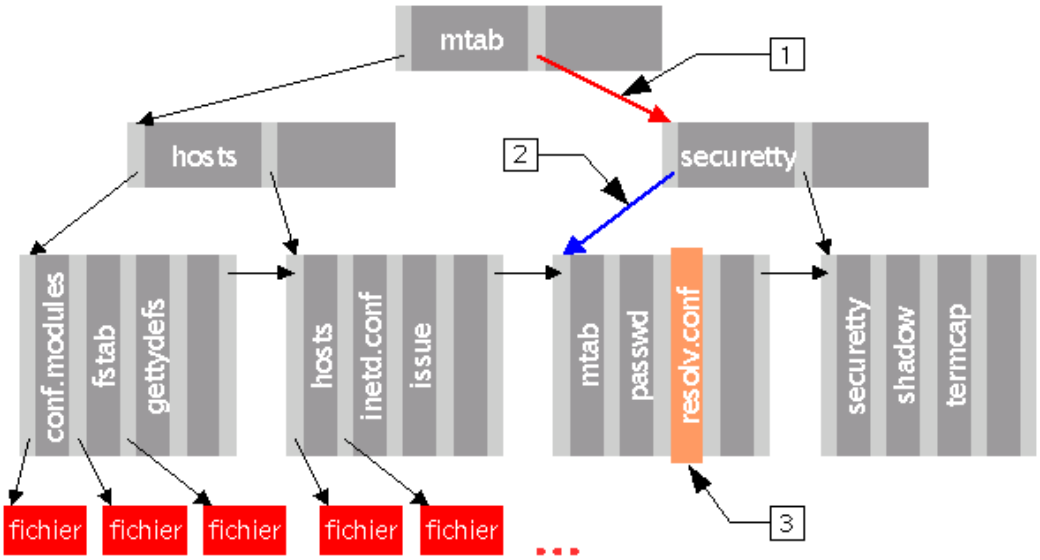
- **Le système NTFS** : Les systèmes récents de Microsoft (Windows NT) utilisent le système de fichiers NTFS qui présente les caractéristiques suivantes :

  Les informations sont regroupées dans une base de données dont la table principale s'appelle **MFT (Master file table).**
  Cette table contient des **FRS (File record segment),** enregistrements de 1 à 4 Ko qui regroupent tous les attributs nécessaires à la gestion de chaque fichier (nom, informations de sécurités) mais également des données (attribut résident).
  Un petit fichier peut donc être entièrement contenu dans la MFT ce qui accélère énormément son accès

Pour les gros fichiers, on retrouve dans le FRS des informations de pointeurs permettant de localiser les données sur le disque (attribut non résident). Pour les gros fichiers, il peut y avoir plusieurs entrées dans la MFT pour contenir toutes ces infos.

Un autre caractéristique de NTFS est la capacité de créer **des "versions" multiples** des données dans un fichier (appelées flux ou lots) permettant d'avoir des données nommées différemment au sein d'un même fichier. On distingue alors les **flux anonymes** des **flux nommés**.
Cette possibilité est peu utilisée par les applications à ce jour.

La localisation des fichiers dans la MFT utilise la technique de l'arbre binaire (B+) permettant un accès très performant. Le principe de fonctionnement est illustré ci-dessous :



1. Pour localiser le fichier resolv.conf, nous commençons à la racine de l'arbre, lisons séquentiellement, et trouvons qu'il n'y a pas de clé supérieure à celle de resolv.conf, donc nous suivons le dernier pointeur (en rouge).
2. Nous sommes redirigés sur un autre noeud de l'arbre. On fait de même qu'avant, on lit les clés jusqu'à "securetty", qui est supérieure à "resolv.conf". On suit donc le lien associé (en bleu).
3. Nous arrivons ainsi à une feuille de l'arbre. Il est temps de lire séquentiellement par ordre ascendant les clés présentes. Nous trouvons enfin la clé désirée, nous permettant d'utiliser le pointeur associé vers les données du fichier "resolv.conf"

D'autres systèmes de fichiers récents pour Unix/Linux utilisent ce principe (**ReiserFS, XFS**)

COURS ARCHITECTURE LOGICIELLE DES SYSTEMES D'INFORMATION

MODULE 02 : LINUX – Installation et Notions de base (distribution DEBIAN)

5. Introduction.

Une "distribution" GNU/Linux comporte en général les éléments suivants :

- ⇨ Un système basé sur le **noyau Linux**,
- ⇨ un **programme d'installation** convivial,
- ⇨ un **système de "packages"** (=logiciel) permettant de gérer facilement les applications installées.

Debian utilise un système de gestion des versions très rigoureux qui fait cohabiter systématiquement 3 niveaux de version :

- ⇨ **Une version "Stable"**, actuellement la version 5.0 (baptisée lenny). Cette version n'évolue quasiment pas (sauf mises à jour de sécurité) mais est garantie très très stable!
- ⇨ **Une version "Testing"**, actuellement la version "squeeze ". Cette version bénéficie des versions plus récentes des différents logiciels mais n'est pas entièrement validée. Elle est malgré tout suffisamment stable pour être installée sans soucis.
- ⇨ **Une version "Unstable"**, toujours appelée version "sid". Cette version bénéficie des dernières versions de chacun des packages mais n'est pas testée. Elle est réservée aux "maniaques des dernières versions" près à passer du temps pour stabiliser un système…

Le site officiel de Debian offre les **CDs d'installation en téléchargement**. Plusieurs solutions :

- ⇨ Téléchargement de la version complète: plusieurs DVD qui contiennent tous les packages…
- ⇨ Téléchargement de la version minimum (netinst): 1 CD de 160Mo qui permet de réaliser l'installation du système et ensuite de télécharger les packages nécessaires.

Le seconde solution, plus intéressante, nécessite d'avoir une connexion Internet opérationnelle lors de l'installation donc un mode de connexion supporté en standard par le noyau (du moins si l'on veut installer une version complète immédiatement. Le CD Netinst est suffisant pour installer un système opérationnel mais rustique !)

6. Installation du système.

Il suffit de booter sur le CD netinst pour démarrer l'installation.

Sur le premier écran, il est possible de choisir le mode d'installation (voir l'aide <F1>) :

- ⇨ <Ent> uniquement ➔ Installation classique
- ⇨ installgui <Ent> ➔ Installation en mode graphique

Après quelques écrans de configuration basiques, on arrive à la **configuration des partitions** :

Un système Linux fonctionne en principe avec un **minimum de 3 partitions**:

- ⇨ La partition système (formatée en par défaut en Ext3),

- La partition utilisateurs (formatée par défaut en Ext3),
- La partition du Swap ou fichier d'échange de la mémoire virtuelle (formatée en swap),

mais cela n'est pas une obligation.
Pour réaliser cela, on peut utiliser l'assistant qui nous proposera 3 modèles mais utilisera obligatoirement tout l'espace restant sur le disque sélectionné :

- **Tout dans une seule partition** = 2 partitions (données + Swap),
- **Mode Ordinateur de bureau** = 3 partitions comme décrites ci dessus,
- **Mode Station de travail multi utilisateur** = 6 partitions.

Il est également possible de créer les partitions à la main pour contrôler leur taille. Il est conseillé de créer au moins 2 partitions (une données, une swap).

Après la copie des fichiers nécessaires, il est question du **gestionnaire de boot**. Il est possible de choisir **LILO** (le plus classique) ou **GRUB** (le défaut de Debian).

Pour utiliser LILO, il faut choisir "Revenir en arrière", pour arriver sur la liste des étapes d'installations et choisir l'installation de Lilo.

Le système redémarre et le reste de la configuration de base est alors demandée :

- Le fuseau horaire
- le **mot de passe de root** (=administrateur)
- Un **compte utilisateur de base**
- La source par défaut des packages
- Eventuellement l'installation de premiers packages standards
- Eventuellement la configuration du serveur de messagerie (utilisé en interne par Linux)

Mis à part le mot de passe root, indispensable à la connexion, tous ces paramètres sont modifiables par la suite.

C'est terminé… Debian est installé et prêt à être utilisé en mode texte!

7. Utilisation d'un système Linux.

3.1. Où trouver de l'aide?

La commande man permet d'afficher les pages du manuel d'utilisation pour une commande donnée.
Syntaxe : `man [options] [section] commande`

section = section du manuel
Il est possible de faire une recherche lors de la lecture du man en utilisant `/{chaine à rechercher}`, puis `n` pour le suivant et `N` pour le précedent.
`Q` ou **q** permet de sortir.

La commande whatis permet d'afficher une courte description d'une commande donnée.

Syntaxe : `whatis [options] commande`

La commande apropos permet d'afficher la liste des commandes concernées par un mot clé.

Syntaxe : `apropos [options] mot_cle`

La commande info lance un environnement complet d'aide destiné à remplacer l'historique man.

Syntaxe : `info [options] [mot_cle]`

3.2. Les commandes Linux.

La syntaxe générale des commandes est toujours la suivante :

`commande [options] [paramêtres]`
Les options sont précédées d'un **–** et précisent (ou modifient) l'action de la commande**.**

Les redirections
Toute commande utilise les 3 flux de données suivants :

- L'entrée standard – stdin (par défaut le clavier)
- La sortie standard – stdout (par défaut l'écran)
- La sortie d'erreur – stderr (par défaut l'écran)

Il est possible de rediriger ces flux standards vers des fichiers :

- `<` redirige l'entrée standard vers le fichier spécifié ➔ on n'utilise plus le clavier mais les données du fichier pour les entrées de la commande.
- `>` redirige la sortie standard vers le fichier spécifié ➔ tous les affichages seront enregistrés dans le fichier. `>>` permet de rediriger sans écraser le fichier (ajout à la fin).
- `2>` redirige la sortie erreur vers le fichier spécifié ➔ tous les messages d'erreur seront enregistrés dans le fichier. `2>>` permet de rediriger sans écraser le fichier (ajout à la fin).

par exemple `man ls > manuel.txt` enregistre dans le fichier *manuel.txt* le résultat de la commande **man ls** (affichage du manuel pour la commande ls).

Il est également possible de rediriger la sortie d'une commande vers l'entrée d'une autre commande = **enchaîner les commandes** à l'aide du caractère **|**.

Par exemple `ps –ef | grep user` enchaîne la commande `ps` (affichage des processus actifs) et la commande `grep` (affichage des lignes selon un critère). Le résultat est donc la liste des processus contenant le mot user.

Le mode de lancement.
Par défaut une commande est lancée en mode "Intéractif" c'est à dire qu'elle est exécutée dans la console où elle a été lancée et le symbole d'invite sera à nouveau affiché lorsque la commande sera finie.
& placé après la commande permet de lancer cette commande en arrière plan. L'invite réapparaît immédiatement.

3.3. Connexion, Déconnexion, arrêt du système.

L'authentification est une opération obligatoire. Elle permet de lancer un interpréteur de commande (shell) et de positionner un répertoire par défaut.

`whoami` permet de connaître l'utilisateur connecté.

La déconnexion est réalisée par une des commandes suivantes :

- ⇨ `logout`, `exit`, ou `<crtl>D`

L'arrêt du système est provoqué par une des commandes suivantes :

- ⇨ `halt` : Arrêt immédiat de la machine
- ⇨ `shutdown :` arrêt de la machine avec des options supplémentaires
- ⇨ `reboot :` Redémarrage

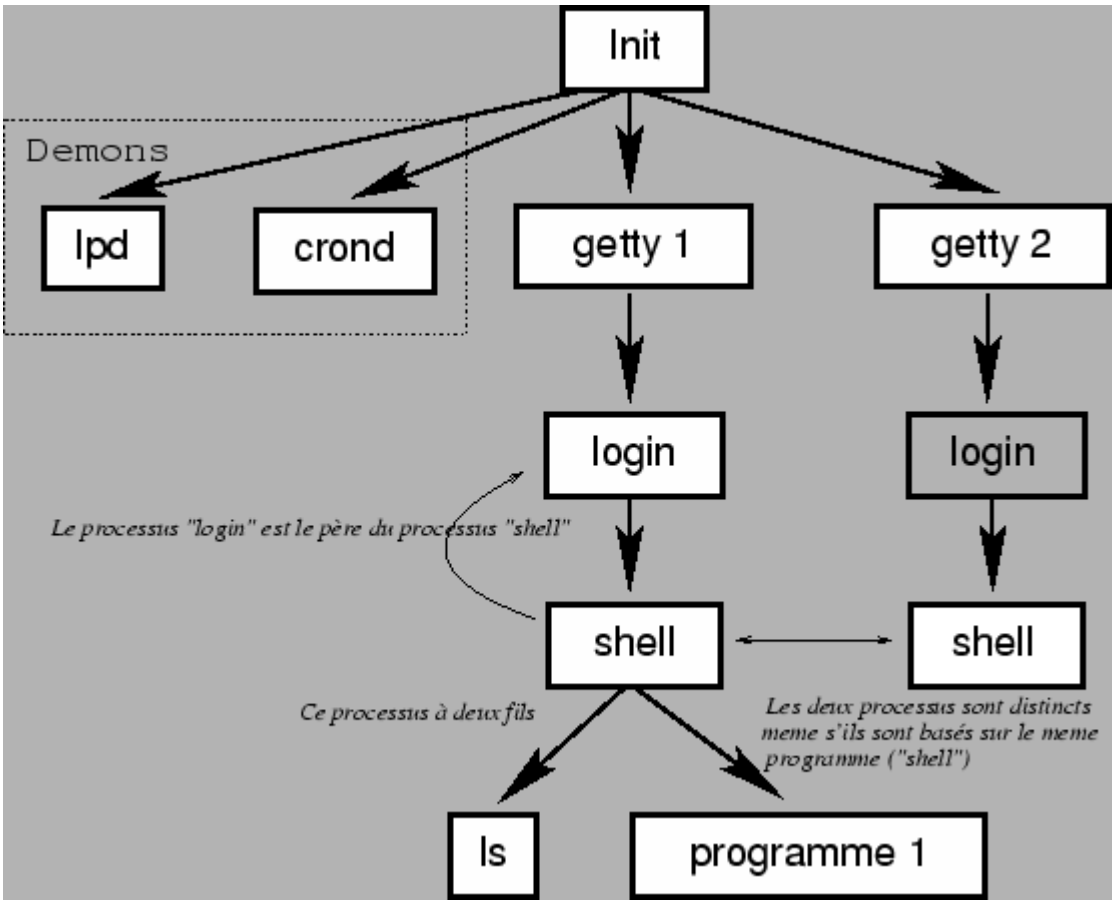
Un système Linux stable n'a pas besoin d'être redémarré et il n'est pas rare de voir des serveurs fonctionner de nombreux mois sans arrêt.

3.4. Les processus.

Chaque processus est une **entité indépendante** disposant de son propre espace mémoire.

Tout travail réalisé par Linux est pris en charge par un processus. Le principal rôle du noyau consiste à gérer les processus pour leur fournir les ressources nécessaires à leur déroulement et leur permettre d'interagir avec le monde extérieur.

Dès son lancement, le noyau lance **un premier processus (init)** qui est l'ancêtre de tous les processus lancés sur le système par la suite. **Chaque processus peut lancer d'autres processus**, il est alors le père de ces derniers (et ces processus sont ses fils).



Chaque processus est identifié par un nombre (le **PID**).

Commandes principales de manipulation des processus

Commande	Syntaxe	Description
ps	ps [options]	Permet de consulter la liste des processus en cours. Par défaut, seuls les processus lancés depuis l'interpréteur de commande courant sont listés. L'option '**-edf**' permet de retourner la liste exhaustive des processus actifs dans le système.
kill	kill -signal pid	Permet de demander l'arrêt de l'exécution d'un processus. L'option '**-9**' est plus violente et force l'arrêt immédiat du processus, sans passer par la procédure de sortie éventuellement prévue dans le programme. Cette option est à utiliser lorsque le processus ne répond pas à la première "sommation".
jobs	Jobs	Permet de consulter la liste des travaux lancés en arrière plan depuis l'interpréteur de commande courant. Rappel : un processus est lancé en arrière plan en ajoutant un "**&**" derrière la commande.
top	top [options]	Application permettant de surveiller en temps réel l'activité du système (processus actifs, utilisation mémoire et processeur...).

MODULE 03 : LINUX – Utilisation du système de fichiers (distribution DEBIAN)

1. Les fichiers Unix

Linux reprend le principe général des systèmes Unix suivant lequel **"tout est fichier"** (ou presque).

Les différents types de fichiers sous Linux sont :

- ⇨ **les fichiers normaux** : ce sont des collections d'octets. Il n'existe pas de différence entre les fichiers texte et les fichiers binaires.
- ⇨ **les répertoires** : ce sont des fichiers contenant les noms des fichiers et leurs numéros d'inode. Un répertoire contient toujours deux sous répertoires spéciaux '.' et '..' qui correspondent respectivement au répertoire courant et au répertoire père.
- ⇨ **les liens symboliques** : permettent de présenter une image d'un fichier sous un autre nom ou à un autre endroit sans dupliquer les données. Un lien symbolique ``pointe'' vers un autre fichier, appelé fichier cible. La notion de lien symbolique peut être comparée à celle de raccourcis sous Windows.
- ⇨ **les fichiers spéciaux en mode bloc** : sont les portes sur les périphériques fonctionnant par blocs de données (ex : disques). L'accès à un périphérique en mode bloc peut se faire aléatoirement (accès à n'importe quel emplacement du fichier).
- ⇨ **les fichiers spéciaux en mode caractère** : sont les portes vers les périphériques fournissant ou consommant les données octet par octet. L'accès à un périphérique en mode caractère ne peut se faire que séquentiellement (accès successif à chaque emplacement du fichier), si bien qu'il est impossible d'accéder au bloc n avant de parcourir le bloc n-1. Ce type d'accès séquentiel est assez bien illustré par le fonctionnement d'un lecteur de bande.
- ⇨ **les tubes nommés "FIFO"** : permettent à deux processus sans relation de parenté de s'échanger des données comme par un tube. L'écriture et la lecture dans un tube sont nécessairement séquentielles. L'écriture remplit le tube, la lecture le vide. On dit que la lecture des données d'un tube est destructive

Les systèmes de fichiers Unix sont structurés hiérarchiquement, et regroupent les fichiers dans des répertoires. Il existe **un répertoire racine** (/), d'où débutent tous les chemins possibles dans le système de fichiers.

La qualification complète d'un fichier se fait en précisant le nom du répertoire à chaque niveau et en séparant par des slashes chacun de ces noms. Cette qualification porte le nom de ``chemin'' d'accès ("path" en anglais). Par exemple :**/home/remi/CNAM/linux/linux.tex**

Microsoft a préféré la barre oblique inverse (nommée "backslash" en anglais).

Les chemins d'accès Unix ne comportent **pas de spécification de lecteurs**. Les systèmes de fichiers Unix sont dits mono-tête, ce qui signifie qu'ils n'ont qu'un seul point de départ : le répertoire racine.

La correspondance entre l'arborescence du système de fichiers et l'emplacement physique des données sur le ou les disques se réalise par une opération "de montage" qui consiste à indiquer **le point de montage** de chaque volume logique. (cf montage des systèmes de fichiers plus loin dans ce document).

Il est par exemple courant de monter le lecteur de CD dans le répertoire **/mnt/cdrom**; Pour accèder à un fichier appelé "mesdonnees" sur le CDROM, on utilisera donc le chemin **/mnt/cdrom/mesdonnees**.

Remarque : Ce mode de fonctionnement est partiellement possible avec Microsoft Windows depuis Windows 2000 en utilisant NTFS.

On peut noter 2 autres différences avec les systèmes Microsoft :

- ⇨ Les fichiers **ne sont pas identifiés par leur extension**. Le point ne signifie rien de particulier.
- ⇨ Les noms de fichier sont **SENSIBLES A LA CASSE** (distinction minuscules, majuscules).

2. Commandes de parcours du système de fichiers.

Linux étant un système "tout fichier", il est primordial de pouvoir naviguer dans le système de fichiers :

Commande	Syntaxe	Description
`cd`	`cd repertoire_dest.`	Change de répertoire courant. Retourne au répertoire personnel si on ne précise pas de répertoire de destination. L'option '-' permet de retourner au répertoire précédent.
`pwd`	`pwd`	Retourne le nom du répertoire courant.
`ls` `dir`	`ls [-l] chemin` `dir [-l] chemin`	liste le contenu du répertoire. L'option –l permet d'afficher les informations sur chaque répertoire / fichier listé. Les "jockers" `(*, ?)` sont possibles.
`mkdir`	`mkdir nom_repertoire`	Crée un nouveau répertoire.
`rmdir`	`rmdir repertoire`	Supprime un répertoire vide. On peut également utiliser la commande `rm -R`.

Pour naviguer dans les répertoires, on dispose des éléments suivants:

Symbole	Signification
`..`	Répertoire précédant
`.`	Répertoire courant
`/`	Racine du système de fichier
`~`	Répertoire de connexion de l'utilisateur courant
`~util`	Répertoire de connexion de l'utilisateur "util"

Par exemple : `cd ../../usr/opt/exple` provoque la remontée de 2 niveaux de l'arborescence puis l'accès au répertoire usr/opt/exple.

`ls /root` provoque le listage du répertoire /root (chemin absolu).

`ls ./user` provoque le listage du sous répertoire user (chemin relatif).

3. Organisation type du système de fichiers

L'organisation présentée correspond aux "cas les plus fréquemment rencontrés". Elle peut néanmoins varier en fonction de la distribution.

⇨ `/`

Répertoire racine.
Point de départ de toute la hiérarchie du système de fichiers.
Le système de fichiers contenant **ce répertoire est monté automatiquement** par le noyau pendant l'amorçage du système.
Ce système de fichiers est appelé système de fichiers racine (``root'' en anglais).

⇨ `/bin`

Répertoire contenant les **commandes générales** nécessaires à l'utilisation courante du système. Tous les utilisateurs peuvent utiliser les commandes de ce répertoire.

⇨ `/sbin`

Répertoire contenant les **commandes nécessaires à l'administration du système**.
Seuls les administrateurs ont accès à ces programmes, les autres utilisateurs n'y ont pas accès.

Exemple de commandes de /sbin/ :

- `reboot` : permet de redémarrer le système,
- `halt` : permet d'arrêter le système,
- `fdisk` : permet de modifier le partitionnement des disques,
- `fsck` : permet de vérifier l'état des systèmes de fichiers,

⇨ `/home`

Répertoire contenant **les répertoires des utilisateurs** disposant d'un compte sur la machine.
Chaque utilisateur possède généralement son propre répertoire.
Le répertoire personnel de chaque utilisateur contient ses fichiers personnels et les fichiers de configurations qui lui sont propres.

Exemples de fichiers de configuration présents dans le répertoire personnel d'un utilisateur :

- `.bashrc` : script exécuté à chaque connexion de l'utilisateur. Il permet de définir des alias, d'exécuter des commandes ou de positionner des variables d'environnement, par exemple.
- `.bash_logout` : idem pour la déconnexion.
- `.bash_history` : historique des commandes utilisées.

Remarque : Un fichier qui commence par un point est "caché" aux utilisateurs.

⇨ `/root`

Répertoire contenant **le répertoire personnel de l'administrateur** (root).

Même s'il serait tout à fait envisageable de placer ce répertoire sous /home/, il est recommandé de le placer au plus près de la racine pour éviter qu'un problème sur le système de fichiers des utilisateurs (utilisation de NFS et lien réseau cassé, par exemple) ne l'empêche de travailler.

Ce répertoire est généralement uniquement accessible à l'administrateur système.

⇨ /etc

Ce répertoire contient **les fichiers de configuration du système**.

Exemples de fichiers situés dans /etc/ :

- /etc/passwd contient la liste des utilisateurs du système,
- /etc/group contient la liste des groupes du système,
- /etc/fstab contient la définition des montages de systèmes de fichiers,
- /etc/lilo.conf contient la définition des paramètres d'amorçage (utilisé pour configuré un double boot, par exemple),
- /etc/exports description des répertoires exportés par NFS,
- /etc/hosts traduction d'adresses IP en noms d'hôte,
- /etc/resolv.conf règles de résolution des noms de domaine...

On trouve les sous-répertoires suivants :

- /etc/X11/ contient les fichiers de configuration de XWindow,
- /etc/rc.d/ contient les scripts de démarrage du système,
- /etc/init.d/ contient les scripts de démarrage des différents services (deamons) du systèmes (apache, MySql...),
- /etc/cron/ contient les tâches à effectuer à la périodicité donnée (daily, hourly, monthly, weekly),
- /etc/skel/ contient les fichiers à recopier dans le répertoire d'un nouvel utilisateur,
- /etc/sysconfig/ contient les fichiers de configuration des périphériques.

Le répertoire /etc/opt/ contient les fichiers de configuration des applications.

⇨ /var

Répertoire contenant toutes **les données variables du système**, c'est à dire les données pouvant être modifiées lors de l'utilisation courante du système ou de ses applications.

Par exemple :

- /var/log/ contient les fichiers de trace de fonctionnement du système.

⇨ /usr

Répertoire contenant **les fichiers du système partageables** en réseau et en lecture seule.

Le répertoire /usr contient de nombreux sous-répertoires. On retrouve presque **la même organisation que sous la racine**, mais le contenu est destiné aux utilisateurs plus qu'au système lui-même.

La structure de /usr est la suivante :

- /usr/X11R6/ contient la hiérarchie des fichiers Xwindow. On retrouve, dans ce répertoire, une hiérarchie de fichiers ressemblant à celle de la racine, mais dédiée à l'environnement XWindow.
- /usr/bin/ contient les commandes utilisateurs supplémentaires du système,
- /usr/doc/ contient les documentations en ligne,
- /usr/etc/ contient la configuration des commandes utilisateurs,
- /usr/include/ contient les fichiers d'en-têtes du système pour le compilateur C/C++.
- /usr/lib/ contient les bibliothèques partagées de tous les programmes de /usr/bin/ et /usr/sbin/ et les bibliothèques statiques pour la création de programmes.
- /usr/local/ contient les outils installés en dehors du contexte de la distribution. Ce répertoire contient les sous-répertoires bin, lib, include et src, qui ont la même signification que les répertoires du même nom de /usr/.
- /usr/man/ contient les fichiers des pages du manuel en ligne,
- /usr/sbin/ contient les commandes d'administration supplémentaires. Ces commandes ne sont normalement utilisées que par l'administrateur système.
- /usr/src/ contient les fichiers sources du noyau et des applications de la distribution. Normalement, ce répertoire ne doit contenir que le code source des applications dépendantes de la distribution que vous utilisez. Il est vivement recommandé de conserver les sources du noyau de Linux sur son disque, afin de pouvoir changer la configuration du système à tout moment.

⇨ **/opt**

Répertoire contenant **les applications complémentaires** n'appartenant pas à la distribution installée. Souvent vide car faisant double emploi avec /usr/local/.

⇨ **/dev**

Répertoire contenant tous **les fichiers spéciaux permettant d'accéder aux périphériques**.
Il existe un tel fichier pour chaque périphérique.

Ce répertoire peut être géré de deux façons :

- *dans un système de fichier classique*. Dans ce cas, il est nécessaire de créer un fichier spécial pour chaque périphérique installé. Cette solution présente l'avantage de permettre un contrôle précis des droits d'accès aux périphériques.
- *dans un système de fichiers virtuel de type devfs*, géré directement par le noyau. Le répertoire /dev/ ne contient dans ce cas que les fichiers spéciaux des périphériques pour lesquels le noyau dispose d'un gestionnaire intégré ou chargé dynamiquement. Cette dernière solution présente l'avantage de gérer correctement les périphériques branché à chaud sur le système (sur le bus USB, par exemple).

Exemples de fichiers du répertoire /dev/ :

- /dev/hda désigne le premier disque dur IDE du système,
- /dev/hda1, la première partition de ce disque.
- /dev/sda1, la première partition du premier disque SCSI ou de la clef USB.
- /dev/cdrom désigne le lecteur de CD-ROM.
- /dev/usb/scanner0 désigne le premier scanner USB.
- /dev/console désigne la console du système.

⇨ **/mnt**

Répertoire réservé au **montage des systèmes de fichiers non-permanents** (CD-ROM, disquettes, etc.).
Ce répertoire peut contenir plusieurs sous-répertoires pour chaque périphérique amovible, afin de permettre d'en monter plusieurs simultanément.

⇨ `/boot`

Ce répertoire contient **le noyau de Linux** et ses informations de symboles.
Ce répertoire est souvent le point de montage d'un système de fichiers de très petite taille, dédié au noyau. Dans ce cas, il est recommandé de monter ce dernier en lecture seule.

⇨ `/tmp`

Ce répertoire permet de **stocker des données temporaires**.
Il dispose généralement de droits d'accès particuliers permettant à chaque utilisateur d'y créer des fichiers sans pouvoir consulter les fichiers des autres utilisateurs.

⇨ `/lib`

Répertoire contenant **les bibliothèques partagées (``DLL'' en anglais**, pour ``Dynamic Link Library'') utilisées par les programmes du système.

`/lib/modules` contient les modules additionnels du noyau. Ces modules sont des composants logiciels du noyau, mais ne sont pas chargés immédiatement pendant l'amorçage. Ils peuvent en revanche être chargés et déchargés dynamiquement, lorsque le système est en fonctionnement.

⇨ `/proc`

Ce répertoire représente le point de montage du **pseudo système de fichiers du noyau**. Ce dernier contient des fichiers permettant d'accéder aux **informations sur le matériel, la configuration du noyau et sur les processus** en cours d'exécution.

On retrouvera, par exemple, un répertoire par processus actif. Chacun porte le numéro du processus décrit et contient les fichiers suivants :

- `cmdline` contient la ligne de commande qui a créé le processus,
- `status` contient des informations sur l'état du processus (en attente, en exécution, propriétaire...),
- `exe` est un lien vers le fichier exécutable utilisé par le processus...

Dans le répertoire `/proc/`, on retrouve des fichiers contenants des informations générales sur le système. Par exemple :

- `uptime` contient le temps de fonctionnement du système,
- `stat` contient diverses statistiques sur l'utilisation des ressources du système (CPU, mémoire...),
- `meminfo` contient un récapitulatif de l'utilisation de la mémoire.
- `cpuinfo` contient une description des CPU du système...

4. Montage des systèmes de fichier

Le montage des systèmes de fichier correspond à la mise en relation entre une branche de l'arborescence et une ressource physique (partition de disque dur, répertoire réseau, lecteur de disquette…)

Cette opération est réalisée à l'aide de la commande **mount** :

```
mount -t <type de système de fichier> <nom fichier> <emplacement>
```

- **Type de système de fichier** décrit le format utilisé par le système de fichier à monter.

Type	Signification
ext2	Système de fichier standard de Linux
ext3	Système de fichier journalisé de Linux
fat	Système de fichier FAT (DOS)
vfat	Système de fichier FAT32 (depuis Windows 95r2)
iso9660	Système de fichier des CD-ROM
ntfs	Système de fichier de Windows NT (lecture seule)
nfs	Système de fichier réseau (Unix)
smb	Système de fichier réseau (Windows)

- **Nom fichier** est le nom du fichier contenant le système de fichier à monter. Il s'agit généralement de fichiers du répertoire `/dev/`

- **Emplacement** est le répertoire de la hiérarchie des fichiers où sera monté le système de fichiers. Le contenu précédent du répertoire (s'il n'était pas vide), ainsi que son propriétaire et ses modes d'accès initiaux deviennent invisibles tant que le nouveau système de fichiers reste monté. Le chemin d'accès du répertoire représente alors la racine du système de fichiers se trouvant sur le périphérique.

Par exemple :

```
mount -t ext2 /dev/hda2 /home
```

La commande inverse est **umount**

Le fichier **/etc/fstab**, contient des lignes décrivant les systèmes de fichiers habituellement montés, leurs répertoires, et leurs options. Ce fichier est utilisé dans trois buts :

- **Le montage automatique des systèmes de fichiers** couramment utilisés. La commande `mount -a [-t type]` (généralement exécutée dans un script de démarrage) monte tous les systèmes de fichiers indiqués dans fstab (ou uniquement ceux du type indiqué), à l'exception de ceux dont la ligne contient le mot-clé ``noauto''.

- **La simplification du montage des systèmes de fichiers**. Lorsque l'on monte un système de fichiers mentionné dans la fstab, il suffit d'indiquer le point de montage, ou le périphérique (exemple : mount /mnt/cdrom pour monter le CD-ROM).

- **La "démocratisation du droit de montage des systèmes de fichiers"**. Normalement, seul le Super-Utilisateur peut monter des systèmes de fichiers. Néanmoins, si la ligne de la fstab contient l'option user, n'importe quel utilisateur peut monter le système de fichiers correspondant.

5. Commandes de manipulation des fichiers.

Commande	Syntaxe	Description
`cat`	`cat chemin/fichier`	Liste le contenu du fichier. Si le contenu dépasse l'écran, seule la fin est visible.
`more`	`more chemin/fichier`	Liste le contenu du fichier page par page. Lors du listage utilise les mêmes commandes que le man. Le more peut aussi s'utiliser en enchaînement de commandes (ex : `ls | more`)
`grep`	`grep critère [fichier]`	Recherche un critère dans des fichiers ou dans l'entrée standard. Le more peut aussi s'utiliser en enchaînement de commandes
`cp`	`cp sources destination`	Copie les fichiers source vers la destination. Il peut y avoir plusieurs sources, mais il n'y a toujours qu'une seule destination. S'il y a plusieurs sources, la destination est nécessairement un répertoire.
`mv`	`mv sources destination`	Déplace les fichiers source vers la destination.
`rm`	`rm fichier(s)`	Supprime un ou plusieurs fichier(s). L'option '-R' permet de parcourir le contenu des éventuels répertoires.
`ln`	`ln -s fichier nom_lien`	Crée un lien vers un fichier existant = raccourcis. L'option '-s', qui permet de créer un lien symbolique au lieu d'un lien physique, est a privilégier car la gestion des liens physiques peut très vite devenir complexe.
`find`	`find repertoire [options]`	Permet de rechercher, suivant de nombreux critères, des fichiers à partir d'un répertoire de base. L'option -name permet, par exemple, de filtrer la recherche sur le nom des fichiers.
`locate`	`locate critère`	Permet de rechercher un fichier en se basant sur une base de donnée et non sur un parcours de l'ensemble du système de fichiers (beaucoup plus rapide). La base de donnée doit être tenue à jour à l'aide de la commande `updatedb`.
`nano`	`nano fichier`	Editeur de texte mode caractère

Gestion des utilisateurs.

Au niveau du noyau, **chaque utilisateur est identifié** de manière unique par un numéro dans le système : son **UID (User IDentifier)**. Ce numéro est utilisé pour vérifier les droits de l'utilisateur qui comprennent la possibilité de lire ou écrire un fichier, d'accéder ou non à une ressource ou d'exécuter un programme.

Il est possible de définir plusieurs utilisateurs ayant le même UID. Ceux-ci auront les mêmes droits d'accès, mais pourront bénéficier de mots de passe, de répertoires de départ ou d'interpréteur de commandes différents.

Il est également possible de créer un ou plusieurs **groupes d'utilisateurs**, et de leur donner des droits particuliers. Chaque groupe est aussi identifié par un numéro : **GID (Group IDentifier)**. Tous les utilisateurs qui font partie de ce groupe recevront les droits du groupe.

Il existe un utilisateur spécial, qui a tous les droits : l'administrateur du système : **root**. Aucune restriction ne s'applique à cet utilisateur

Il existe plusieurs façons de gérer les utilisateurs et les groupes :
- ⇨ **Les fichiers locaux**
  situés dans le système de fichier et utilisés par les outil de connexion et de vérification de droits d'accès.
- ⇨ **Un serveur d'authentification (NIS ou LDAP)**
  à qui le système s'adresse pour identifier de façon centralisée les utilisateurs, groupes et leurs droits.

L'identification par fichier locaux se base principalement sur deux fichiers :

- ⇨ Le fichier `/etc/passwd` qui contient la liste des utilisateurs du système,
- ⇨ Le fichier `/etc/group` qui contient la liste des groupes d'utilisateur du système.

1.1. Le fichier `/etc/passwd`

Par exemple, un utilisateur est représenté par une ligne du type :

```
remi:x:500:500:Rémi LEBLOND:/home/remi:/bin/bash
```

Cette ligne comporte les champs suivants séparés par le caractère "**:**" :

- ⇨ **nom de l'utilisateur** pour l'identification sur le système (login),
- ⇨ **mot de passe** crypté. Le "x" indique que le système "shadow" est activé c'est à dire que les mots de passes sont stockés sur un fichier séparé appelé **shadow** afin d'améliorer la sécurité.
- ⇨ **numéro d'utilisateur** (uid),
- ⇨ **numéro de groupe d'utilisateur par défaut** (gid). Tout utilisateur est affecté à un et un seul groupe de base. Il peut, par ailleurs, faire partie d'un grand nombre d'autres groupes.
- ⇨ **nom complet** ; ce champ peut contenir de nombreuses informations, il correspond à un champ de remarque,
- ⇨ **répertoire de base** de l'utilisateur,
- ⇨ **programme à lancer au démarrage** (programme de base en environnement caractère), généralement un interpréteur de commande (shell). La durée de vie de ce processus correspond

à celle de la session utilisateur, c'est à dire que la session de l'utilisateur se terminera avec le processus. Il est possible de préciser ici tout type de programme, ce qui permet de limiter le champ d'action d'un utilisateur (en le connectant directement au programme qu'il doit utiliser, par exemple).

1.2. Le fichier /etc/group

Par exemple :

```
actrices:x:400:sandra,meg,michelle
```

Une ligne de ce fichier comporte les champs suivants, séparés par des caractères `:' :

- **Le nom du groupe**,
- **Le mot de passe** encrypté du groupe. En général, inutilisé (Les mots de passes sont stockés dans le fichier **gshadow**.
- **numéro du groupe** (gid),
- **liste des utilisateurs** appartenant au groupe séparés par des virgules.

Lors de la connexion, un utilisateur est associé à tous les groupes dans lesquels il est inscrit. Il peut le vérifier à l'aide de la commande **id**, ou **groups**.

1.3. Utilitaires de gestion des utilisateurs

Commande	Syntaxe	Description
adduser	adduser	Ajoute un utilisateur en utilisant un dialogue. Les options de fonctionnement sont réglées par le fichier /etc/adduser.conf
addgroup	addgroup	Ajoute un groupe
gpasswd	gpasswd [options] group	gestion du fichier group.
users	users	Utilisateurs actuellement connectés
groups	groups utilisateur	Affiche les groupes auquel appartient un utilisateur
chage	chage [options] utilis.	Gestion de la péremption des mots de passe (nécessite l'utilisation du fichier shadow).
deluser	deluser	Supprime un utilisateur (utilise deluser.conf)
delgroup	delgroup	Supprime un groupe

Le fichier **login.defs** définit les paramètres par défaut des utilisateurs.

2. Gestion des droits d'accès aux fichiers.

2.1. Concepts

Pour chaque fichier, il est défini 3 droits :
- ⇨ Le droit du **propriétaire du fichier (u)**
- ⇨ Le droit du **groupe du fichier (g)**
- ⇨ Le droit pour **tous les autres utilisateurs (o)**.

Par défaut, le propriétaire d'un fichier est l'utilisateur qui l'a créé. De même, le groupe du fichier est le groupe par défaut de cet utilisateur.

On peut décomposer chaque droit en 3 droits élémentaires :

pour les fichiers :

Abrév.	Nom	Signification
r	Droit de lecture (``r'' pour Read)	Droit de consulter le contenu d'un fichier
w	Droit d'écriture (``w'' pour Write)	Droit de modifier le contenu d'un fichier
x	Droit d'exécution (``x'' pour eXecutable)	Droit d'exécuter un fichier contenant un programme

pour les répertoires :

Abrév.	Nom	Signification
r	Droit de listage	Droit de consulter la liste des fichiers d'un répertoire
w	Droit de modification	Droit de modifier la liste des fichiers d'un répertoire. Cela signifie donc ajouter de nouveaux fichiers ou en supprimer d'anciens.
x	Droit d'accès	Droit d'accéder au contenu du répertoire.

Chacun de ces droits élémentaires sont en fait représentés par des bits et ont ainsi une correspondance numérique de 0 à 7(appelée **octale**):
- ⇨ **r** correspond à **100** soit **4**
- ⇨ **w** correspond à **010** soit **2**
- ⇨ **x** correspond à **001** soit **1**

Donc, par exemple la combinaison des droits r et w ➔ 4 + 2 = 6

La commande **ls –l** permet de visualiser les droits des fichiers :

```
drwxr-xr-x  2 root root 4096 2005-05-09 18:51 td1
-rwxrwx---  1 root root   28 2005-05-16 15:33 test
```

Dans cet exemple, les droits du répertoire `td1` (lettre d au début de la ligne) sont :
- ⇨ rwx pour le propriétaire root (soit tous les droits)
- ⇨ rx pour le groupe root

⇨ rx pour tous les autres utilisateurs,
et ceux du fichier test sont :
⇨ rwx pour le propriétaire root (soit tous les droits)
⇨ rwx pour le groupe root (soit tous les droits)
⇨ aucun droit pour tous les autres utilisateurs,

Le droit qui s'applique est en priorité le droit utilisateur, puis le droit groupe si l'utilisateur ne correspond pas et enfin le droit autres si le groupe ne correspond pas.

Lors de la création d'un fichier ou d'un répertoire, le droit par défaut est :
⇨ **666** soit `rw_rw_rw_` pour un fichier créé par une commande courante (par programme, tout est possible!).
⇨ **777** soit `rwxrwxrwx` pour un répertoire.
Il est possible de modifier ce comportement en utilisant un masque défini par la commande **umask**. Cette commande fait en général partie du script de démarrage de chaque utilisateur. Le masque définit en fait **les droits que l'on veut enlever** lors de la création du fichier.

Ainsi, par exemple la création d'un fichier quelconque donne les droits suivants : **rw_rw_rw_** soit en notation octale **666**.
Si la valeur du masque = 022 (soit _____w__w_), cela signifie que l'on veut ôter le droit d'écriture pour groupe et autres.
Le droit résultant sera **rw_r__r__** soit en notation octale 644. Le masque "se soustrait" donc numériquement.

Il n'existe pas de notion d'héritage dans Linux! (sauf attribut SETGID voir plus loin)

2.2. Utilitaires de gestion des droits

Commande	Syntaxe	Description
chown	chown user[:group] fichier	Modifie le propriétaire et éventuellement le groupe du ou des fichiers concernés. Pour un répertoire, l'option -R permet d'appliquer la modification à toute l'arborescence.
chgrp	chgrp group fichier	permet de modifier le groupe du ou des fichiers.
chmod	chmod mode fichier	modifie les droits du ou des fichiers. Le mode peut être exprimé en numérique (par ex 770) ou en symbolique ➔ voir le man.

2.3. Autres attributs du système de fichiers

3 autres attributs peuvent modifier les règles de sécurité d'un fichier :

⇨ L'attribut **SETUID** indique que, lors de l'exécution du programme, c'est l'identifiant du propriétaire du fichier exécutable qui est utilisé plutôt que celui de l'utilisateur l'ayant lancé. Cela signifie que tous les fichiers ouverts par ce programme le seront au nom du propriétaire de l'exécutable.
Un exemple classique pour illustrer cela est le programme passwd qui permet de changer le mot de passe d'un utilisateur. Tout le monde peut l'utiliser pour changer son mot de passe. Mais ce dernier est stocké dans un fichier accessible seulement au root en écriture (/etc/passwd ou /etc/shadow). Pour que passwd puisse le modifier, ce programme a l'attribut SETUID lui conférant les droits du root pendant son exécution.
Cet attribut est représenté par la lettre "**s**" à la place du premier "**x**".

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

⇨ L'attribut "**STICKY**". Cet attribut remplace l'attribut exécutable pour les autres utilisateurs. Contrairement aux bits setuid et setgid, il est représenté par la lettre "**t**" (pour ``sTickky''). Pour les fichiers exécutables, sa signification est désuète : elle permettait de faire en sorte que les programmes restent chargés en mémoire après leur terminaison, ce qui permettait de les relancer plus rapidement. Linux ignore cette fonctionnalité.
Pour les répertoires, sa signification est totalement différente : elle permet de restreindre les droits des utilisateurs sur les répertoires ayant ce bit positionné. Ce bit fait en sorte que même si un utilisateur dispose des droits d'écriture sur le répertoire, **il ne peut pas supprimer** tous les fichiers de ce répertoire. Les seuls fichiers qu'il est autorisé à supprimer sont ses propres fichiers. Bien entendu, il est toujours possible d'ajouter des fichiers dans le répertoire en question.

En résumé :

"**s**" : dans le cas d'un fichier exécutable, celui-ci sera exécuté avec les droits du propriétaire ou du groupe en fonction de la place du symbole. Dans le cas d'un répertoire, tous les fichiers créés dans ce répertoire appartiendront au même groupe que celui du répertoire.
"**t**" : pour les fichiers exécutables, demande de garder le code en mémoire après l'exécution. Pour les répertoires, permet de limiter la destruction des fichiers au propriétaire du répertoire, du fichier ou au super utilisateur .

Modification des attributs spéciaux

La déclaration des attribus spéciaux se fait avec la commande `chmod`

- Soit en mode octal en ajoutant un premier chiffre (4=SetUid, 2=SetGid,1=Sticky) par exemple `4770`.
- Soit en mode symbolique (plus pratique à mon avis) : `u+s` = SetUid, `g+s` = SetGid, `o+t` = Sticky.

MODULE 05 : LINUX – Logiciels complémentaires (distribution DEBIAN)

Notion de package

Au début de Linux, pour installer une option sur un système, il était nécessaire de récupérer les sources sous forme de fichier d'archive tar (fichier tar.gz), de les décompresser et de les compiler.

Debian a introduit (le premier !) la notion de package (dpkg) permettant une gestion simple et fiable de l'installation des différents logiciels serveurs ou clients.

Pour gérer correctement les packages et notamment leurs dépendances, l'outil apt (advance packaging tool) a été créé.

8. Configuration de l'outil apt

Le plus important est de définir la ou les sources de récupération des packages.

Cela est décrit dans le fichier **/etc/apt/sources.list**

Les lignes de ce fichier sont dans ce format :

```
deb http://host/debian distribution section1 section2 section3
deb-src http://host/debian distribution section1 section2 section3
```

deb signifie package debian binaire, deb-src, package source.

En général, le fichier standard après l'installation ressemble a l'exemple ci-dessous :

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-
free
deb http://security.debian.org stable/updates main contrib non-free
# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib
non-free
```

Il est nécessaire de le modifier pour l'adapter à la version utilisée et optimiser les téléchargements:

On peut trouver la liste des miroirs sur **http://www.debian.org/mirror/mirrors_full.**

Pour la France, un fichier correct permettant de gérer la version stable (version 4 en 2007) par internet pourrait être :

```
deb ftp://ftp.fr.debian.org/debian/ stable main contrib.
deb ftp://ftp.fr.debian.org/debian-non-US/ stable/non-US main contrib
```

Dans le cas où l'on veut gérer des packages par CD-ROM, il faut ajouter des lignes dans le fichier sources.list.

L'outil **apt-cdrom add** permet de le faire automatiquement. Il faut répéter l'opération pour chaque CD.
Il est également possible de gérer sa propre bibliothèque de package et/ou créer ses propres cd-roms contenant des fichiers .deb.
La procédure complète se décompose comme suit :

- Copie des fichiers .deb dans un répertoire, par exemple /root/debs
- Positionné sur /root, utilisation de la commande
  `dpkg-scanpackages debs /dev/null | gzip > debs/Packages.gz`
- Pour utiliser les packages copiés dans le répertoire, il suffit d'ajouter la ligne suivante dans sources.list :

`deb file:/root debs/`

- Pour utiliser ces fichiers une fois copiés dans un CD (ou une image ISO), il faut utiliser la commande `apt-cdrom add` qui demande d'insérer le CDROM et l'identifie. Cela ajoute une ligne dans sources.list

9. Utilisation de l'outil apt

3.1 Préparation

Le système de packages possède sa propre base de données qu'il est nécessaire de synchroniser régulièrement avec les bases de données de debian (dans le cas d'une utilisation internet).

On utilise pour cela la commande `apt-get update`
Cette commande doit être obligatoirement utilisée une première fois après la mise à jour du fichier sources.list.
Il est recommandé de la lancer avant toute nouvelle installation ou même automatiquement à l'aide d'un scheduler comme cron.

3.2 Installation de nouveaux package

La commande `apt-get install` permet d'installer de nouveaux packages :

`apt-get install package options`

Il faut donc connaitre le nom du package.

L'option `-s` permet d'effectuer une simulation. Il est conseillé de l'utiliser notamment pour contrôler les dépendances et la taille nécessaire à l'installation.
L'option `-d` permet d'effectuer le téléchargement sans effectuer l'installation. Celle ci pourra être faite plus tard en utilisant les option `--no-download` et `-m` (par exemple pour un PTI !).

Le plus difficile est d'avoir le nom exact du package.

L'outil `apt-cache` permet de rechercher des packages en fonction d'un mot clef :

`apt-cache search motcle`

Options intéressantes : `-i` : ne donne que les dépendances importantes et `-n` ne recherche que sur les noms.

Apt-cache sert également à en savoir plus sur les packages :

```
Apt-cache show package
Apt-cache depends package
```

3.3 Mise à jour des logiciels

La commande `apt-get upgrade` lance la mise à niveau de tous les packages installés.

La version comparée dépend des sources indiquées dans le fichier sources.list. Il est important de lancer un apt-get update au préalable.
Par cette procédure, il est possible de changer de version en modifiant avant le fichier sources.list.

La commande `apt-get dist-upgrade` est équivalente. Elle offre en plus une gestion intelligente de l'ordre de mise à jour des dépendances.

3.4 Suppression des packages

Les logiciels inutiles peuvent être supprimés par la commande :

```
Apt-get remove package
```

Pour supprimer les fichiers de configuration correspondants, on peut utiliser l'option `--purge`

3.5 Nettoyage du cache des packages

Les package sont d'abord copiés sur le système dans le répertoire `/var/cache/apt/archive` puis installés. les fichiers `.deb` restent ensuite dans ce répertoire et peuvent utiliser beaucoup d'espace disque.
La commande `apt-get clean` permet de nettoyer correctement ce cache.

RECAPITULATIF DES COMMANDES et autre commandes utiles

Commande	Syntaxe	Description
`apt-cdrom`	`apt-cdrom`	Renseigne le fichier sources.list en fonction du CDROM Debian fourni
`apt-get update`	`apt-get update`	Met à jour la base de données locale de packages
`apt-get install`	`apt-get install package`	Charge et installe le package nommé
`apt-get upgrade`	`apt-get upgrade`	Met à jour tous les packages
`apt-get remove`	`apt-get remove package`	Supprime le package nommé
`apt-cache`	`apt-cache search motcle` `apt-cache show package`	Recherche de package Affichage des infos sur les packages
`dpkg-reconfigure`	`dpkg-reconfigure package`	Relance le script de configuration du paquet (écrase les éventuelles modifs manuelles des fichiers de config correspondants)
`dpkg -l`	`dpkg -l`	Liste les packages installés

10. Autres méthodes d'installation

4.1 Outil make

Packages nécessaires : **make, gcc**.

L'outil **make** permet de simplifier l'installation de programmes en général **à partir des sources**. Il peut également être utilisé pour simplifier le déploiement des binaires

En effet, un exécutable est en général le regroupement de plusieurs fichiers sources et fonctionne avec des composants extérieurs qui doivent être compilés et placés au bon endroit.

Pour éviter de connaître tout cela (et éviter au programmeur de faire 50 fois la même chose!), l'outil make permet de gérer la compilation. Il utilise pour cela un fichier texte d'un format bien particulier appelé `makefile` (ou Makefile) qui contient toutes les instructions nécessaires.

En bref, les différentes étapes pour installer un logiciel de cette manière sont :

- Récupération de l'archive contenant les sources (en général un fichier au format tar compressé). la logique de l'arborescence veut que l'on place ce fichier dans le répertoire `/usr/src` s'il s'agit un logiciel de la distribution et `/usr/local/src` s'il s'agit d'un logiciel hors distrib.
- Extraction des différents fichiers par la commande tar
  Si fichier **.tar.gz** : `cd /usr/local/src` puis `tar xvfz nom_de_larchive`
  Si fichier **.tar.gz2** : `cd /usr/local/src` puis `tar xvfj nom_de_larchive`
- Vérification de la présence d'un fichier `makefile` et/ou d'un fichier `README`. Lire impérativement le fichier readme.
- Il existe parfois un script configure qui permet de créer le fichier makefile en tenant compte du contexte de la machine. dans ce cas, lancer ce script (`./configure`)
- S'il s'agit d'une installation à partir des sources, lancer une première fois l'utilitaire make sans option (`make`) pour compiler les programmes.
- lancer `make install` pour réaliser les tâches prévues pour l'installation (déplacement des fichiers dans les bons répertoires).

4.2 Scripts d'installations

Certains logiciels sont fournis sous la forme d'un fichier tar avec un script d'installation qui se charge de toutes les opérations nécessaires.

Pour connaître la procédure, il faut lire (entièrement) le fichier readme ou install!

4.3 Système de paquets RPM

Packages nécessaires : **rpm**,**alien**

RPM est le système de packages utilisé notamment par les distributions red-hat, fedora et Mandriva. Il n'est pas directement utilisable avec Debian mais les packages rpm sont quand même exploitables à l'aide du système alien (voir man alien).

RPM est moins puissant que le système apt dans la mesure où il ne gère pas de base de données des packages disponibles mais uniquement une liste des packages installés. Des systèmes

équivalents à APT basés sur RPM existent néanmoins pour les distributions "RPM" (par exemple yum sur Red-Hat).

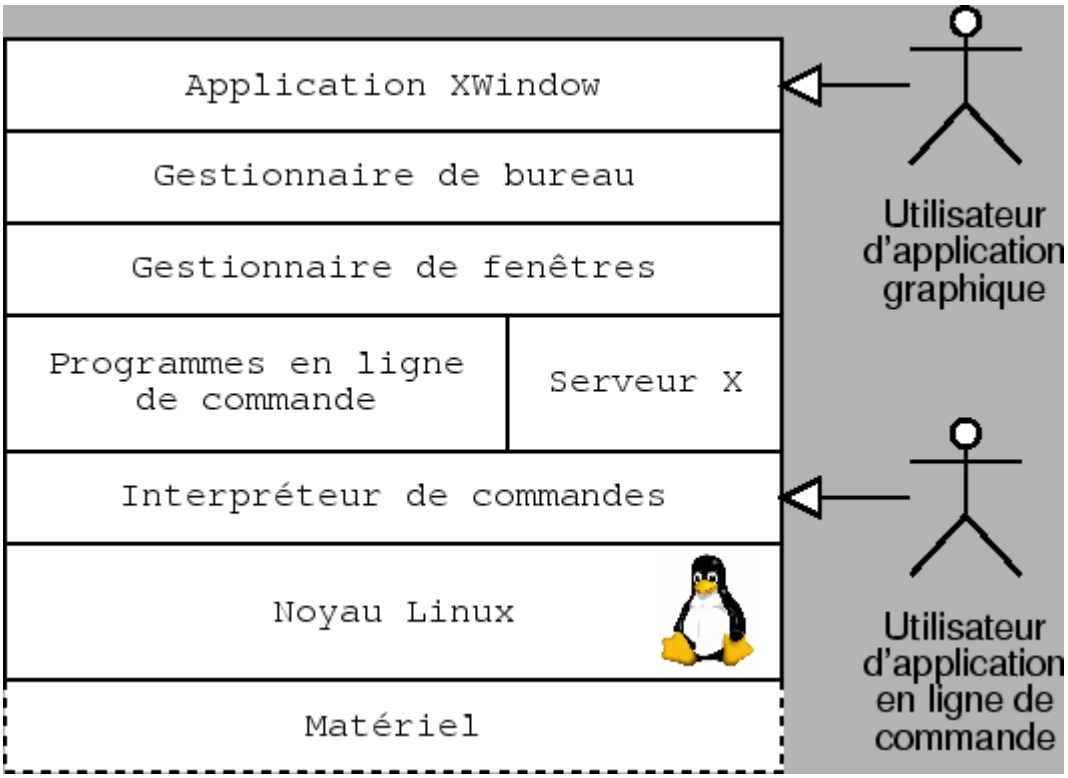
Ici, il est nécessaire de récupérer le fichier archive (.rpm) pour pouvoir l'installer et si une gestion des dépendance existe, elle ne sert qu'à générer une erreur si un paquet dépendant n'existe pas.

Sinon, le principe reste le même, seules les commandes sont différentes
(*NON UTILISABLES AVEC DEBIAN*):

Commande	Description
`rpm -ivh pakage.rpm`	Installation d'un package (i=install, v=verbose, h=barre de progression)
`rpm -Uvh package.rpm`	Mise à jour d'un package (fontionne également pour l'installation)
`rpm -qilp package.rpm`	Liste infos du package (q=query, i=infos, l=liste des fichiers, p=fichier package)
`rpm -e package.rpm`	Suppression d'un package

11. Exemple d'installation : Mise en place d'un environnement graphique

RAPPEL :



L'environnement graphique a donc besoin de 3 composants pour fonctionner :

- Le serveur X Windows
- Le gestionnaire de fenêtres
- Le gestionnaire de bureau

Il est donc nécessaire d'installer au moins 2 packages :

Le serveurs X :
`x-window-system-core` : Le minimum pour bénéficier du graphisme (x.org).
`x-window-system` : Le même avec quelques outils supplémentaires et notamment xdm permettant une ouverture de session graphique.

La configuration du serveur X se trouve dans le fichier /etc/X11/XF86Config-4. Normalement correctement rempli à l'installation du package, il peut être nécessaire de retoucher la profondeur des couleurs en cas de problème au démarrage ou de déclarer une carte vidéo qui n'aurait pas été reconnue…

Le démarrage du serveur X s'effectue de 2 manières:

- La commande `startx` (si aucun gestionnaire de démarrage n'est installé).
- Par un gestionnaire de démarrage graphique comme `xdm`

Le gestionnaire de fenêtres/sessions :

Les plus populaires mais les plus lourds sont **GNOME** et **KDE**.
D'autres sont plus légers mais moins complets :

Blackbox /fluxbox
Xfce
IceWM
Enlightenment

Pour les installer, il suffit de lancer un apt-get install. Je recommande IceWM, simple mais suffisant pour faire des tests.

MODULE 06 : LINUX –Mise en réseau (distribution DEBIAN)

1.1 Configuration du réseau

Il existe 2 méthodes pour configurer le réseau :

- A l'aide de commandes. La configuration sera active jusqu'à la fermeture du système et devra être relancée à chaque démarrage
- A l'aide de fichiers de configuration.

1.2 Configuration réseau par commandes

3 commandes et 1 fichier permettent de configurer la ou les interfaces réseau :

Commande	Syntaxe	Description
ifconfig	ifconfig [nom_ interface] [adresse IP] options up|down	Assigne une adresse IP à une interface et permet de configurer les différentes options.
route	route (voir man)	Gestion de la table de routage
ip	ip link commande ip addr commande ip route commande	Nouvelle version des outils de configuration réseau. Plus complète mais également plus complexe.
/etc/resolv.conf		Fichier contenant la liste des serveurs DNS. Peut être mis à jour par DHCP.

Exemples :

`ifconfig eth0 192.168.1.128 netmask 255.255.255.0` : attribution d'une adresse.
`route default gw 192.168.1.1` : Déclaration d'une route par défaut.

1.3 Configuration réseau par fichier

Il est plus efficace de conserver la configuration du réseau dans un fichier!

Debian fournit un environnement permettant de configurer le réseau à l'aide de 2 commandes et d'un fichier. Il est déconseillé d'utiliser à la fois cette méthode et les commandes précédentes.

Le fichier /etc/network/interfaces contient la configuration. Exemples :

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1
        dns-nameservers 195.238.2.21 195.238.2.22
```

Signifie : Configurer l'interface eth0 avec l'adresse 192.168.0.123/24, la passerelle 192.168.0.1 et 2 serveurs DNS

```
iface eth0 inet dhcp
```

Signifie : Interface eth0 configurée en dhcp (➔ client DHCP).

la ligne auto eth0 signifie de démarrer l'interface au démarrage.

Beaucoup de configurations sont possibles avec le fichier interfaces ➔ man interfaces

2 commandes permettent de gérer l'état de l'interface et de prendre en compte un éventuel changement : `ifdown` et `ifup`

2.1 Partages de données sur le réseau

2 principales méthodes sont disponibles :

- **NFS** (Network file system) réservé aux systèmes Linux / Unix
- **SAMBA** : Netbios pour Linux. Permet de dialoguer avec des systèmes microsoft.

2.2 Configuration de NFS

NFS necessite le package `nfs-kernel-server`.

Il faut ensuite configurer le serveur :

- Le fichier `exports` doit contenir la liste des répertoires "exportables" sur le réseau détaillés par machine (ou groupe de machines). selon ce format :

```
/home          192.168.1.10(rw)
/usr       debian*(ro no_root_squash)
```

signifie : Partage du repertoire /home pour la machine 192.168.1.10 en lecture/écriture
partage du répertoire /usr pour toutes le machines dont le nom commence par debian en lecture seule, et conservation des droits roots).

- Les fichiers `hosts-allow` et `hosts-deny` permettent de gérer les autorisations des machines (par nom de machine ou adresse IP). Il suffit de donner la liste des machines. S'ils sont absents, aucune vérification n'est effectuée.

- Pour prendre en compte toute modification du fichier exports, il est nécessaire de lancer la commande `exportfs -a`.
- Pour contrôler les exports en cours : `exportfs`
- Le programme `portmap` doit également être lancé (à vérifier dans rc2.d).
- Lancer le service par `/etc/init.d/nfs-kernel-server start`.

Puis, au niveau du client :

- Le programme `portmap` doit être lancé (donc package **portmap** installé).

- Il faut faire un mount du partage selon la syntaxe suivante :

```
mount machine:répertoire_exporté point_de_montage
```

- Si l'on veut automatiser le mount ou le rendre dispo aux utilisateurs, il faut alors ajouter une ligne dans la table fstab selon le format suivant :

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/sda1       /               ext3    defaults,errors=remount-ro 0       1
/dev/sda5       none            swap    sw              0       0
/dev/hdc        /media/cdrom0   iso9660 ro,user,noauto  0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
192.168.1.11:/home      /remote/home    nfs
```

La dernière ligne signfie : Montage du répertoire /home de 192.168.1.11 dans le répertoire /remote/home.

NFS et sécurité :

Pour fonctionner correctement, NFS nécessite des UID et GID identiques sur toutes les machines.

Sans options supplémentaire, chaque utilisateur conserve son UID lors de la connexion et le droit correspondant lui est appliqué. Seul root voit ses droits modifiés sur les machines distantes où il est vu comme "nobody".

Les options de "squashing" permettent de modifier cela (root_squash, all_squash, …) et de faire passer tous les utilisateurs distants comme "nobody".

2.3 Configuration de SAMBA

Le package **samba** permet d'installer simplement la partie serveur. Il suffit d'indiquer dans le dialogue de l'installation le nom du Workgroup (ou domaine) et le type d'accès (/ partage soit style W9x ou / utilisateur soit style Windows NT).
Avec des clients W NT (2000, XP), il est vivement conseillé d'utiliser le mode user.

Si l'on choisit le mode user, il faut indiquer à Samba les utilisateurs concernés en utilisant la commande smbpasswd. Il sera demandé un mot de passe qui peut être différent du mot de passe Linux.

Le reste de la configuration se situe dans le fichier /etc/samba/smb.conf.

Par défaut, le serveur Samba est opérationnel avec un seul partage par utilisateur : son répertoire par défaut (en général /home/utilisateur).

On retrouve cette configuration dans la rubrique [homes].

Pour créer un partage, il faut ajouter une rubrique comme par exemple :

```
[commun]
path = /home/commun
writable = yes
public = yes
```

Les principaux paramètres au niveau partage sont :

paramètre	valeur par défaut	description
`path =`		chemin du rep à partager
`comment =`		texte visible dans le voisinage réseau client
`guest ok = yes|no (ancien nom : public)`	no	partage en accès libre sans authentification
`invalid users =`		liste des users non autorisés à se connecter à la ressource
`printable = true|false`	false	partage d'un service d'impression et non de rép.
`writeable = yes|no`	no	permet ou non l'écriture sur le rép., contraire de read only
`write list =`	tous les utilisateurs	liste des users autorisés à écrire
`browseable =`	yes	visibilité du partage par tous, *même les users non autorisés*
`create mode | mask =`	0744	droits maxi accordés à un fichier créé dans la ressource ces droits seront en intersection (and) avec les droits Linux (umask)
`directory mode | mask =`	0755	droits maxi accordés à un répertoire créé dans la ressource ces droits seront en intersection (and) avec les droits Linux (umask)
`hide dot files =`	yes	cache les fichiers cachés au sens Linux, commençant par un point
`hosts allow` `hosts deny =`	toutes les stations aucune	ressource réservée|interdite à la liste des stations (adresses IP)
`max connections =`	0	nb de connexions à la ressource illimité, sinon maxi

Les services (daemons) correspondants sont :

- `nmbd` : Résolution des noms netbios
- `smbd` : Samba

Ne pas oublier de relancer Samba après chaque modification! (`smbd restart`).

Samba comporte de très nombreuses options (gestion des imprimantes, serveur Wins, Contrôleur de domaine): http://www.samba.org

Il est possible de configurer Samba en mode graphique en installant un outil spécifique comme **swat** ou **Webmin**

Si l'on veut accèder à un partage Windows ou Samba à partir une machine Linux, il faut installer un logiciel client :

- **smbclient** : Client Smb en mode ligne de commande (comparable à un ftp ligne de commande)
  Une fois installé, la commande `smbclient -L nom_machine` permet de connaître les partages (identique à net view en environnement Microsoft)
  `smbclient \\\\machine\\partage` permet de travailler sur le partage concerné.

- **smbfs** : Permet de monter un partage netbios comme n'importe quel volume Linux. Permet donc de fonctionner comme avec nfs
  Une fois installé, il suffit d'utiliser la commande smbmount avec les paramètres suivants :

  ```
  smbmount //machine/partage /point_montage -o username=user,password=password
  ```

- **Clients graphiques** : Il existe de nombreux clients graphiques intégrés ou non dans des navigateurs. (smb4k, xsmbrowser…)

MODULE 07 : LINUX – Administration centralisée (distribution DEBIAN)

L'objectif de ce module est de lister les solutions linux qui permettent d'administrer un parc de machines sous Windows ou sous linux de manière centralisée soit principalement la gestion des utilisateurs, des répertoires réseau et des droits.

12. SAMBA

Le logiciel SAMBA (déjà vu précédemment) permet de transformer une machine Linux en PDC (Primary Domain Controller) Microsoft soit en un contrôleur de domaine « compatible Windows NT4 ».
Il n'existe pas à ce jour de version compatible Active Directory…

1.4 Installation du serveur Samba

Pour cela, il faut configurer Samba spécialement :

Dans le fichier `smb.conf` :

Ajouter :
`domain logons = yes` (indique précisément que c'est un contrôleur)
`os level = 34` (version à indiquer aux clients)
`domain master = yes` (indique contrôleur primaire)
`logon path = \\%N\profils\%U` (indique l'emplacement des profils. Ce répertoire doit exister et être accessible en écriture par les utilisateurs donc partagé par Samba…).
Vérifier et/ou modifier :
`encrypt password = true`
`writable = yes` sur partage **home** (pour permettre la création du profil itinérant).
Supprimer ou commenter `invalid users = root`

Samba n'est pas capable de créer les comptes machine, il faut donc le faire à sa place :

- ⇨ Créer un compte utilisateur Unix (sans possibilité de connexion ➔ pg initial = /bin/false) ayant comme nom **nom_machine$.**
- ⇨ Ajouter ce compte dans Samba avec la commande `smbpasswd –a –m nom_machine$` (le –m indique précisement qu'il s'agit d'un compte machine).

Ajouter ensuite les comptes utilisateurs devant fonctionner dans le domaine :
`smbpasswd –a nom_utilisateur` sans oublier root.

1.5 Installation des clients Windows

Les clients Windows utilisent la procédure habituelle pour s'inscrire dans le domaine (Windows 2000 et au-delà).
Comme il s'agit de controleur NT4, il n'y a pas de problématique DNS, tout se passe en Netbios. Il suffit donc de donner le nom du domaine dans les propriétés du poste de travail et s'authentifier en root pour voir apparaitre le message « Bienvenu dans le domaine … ».

Le profil est automatiquement itinérant (stocké dans le répertoire donné par « logon path ») et un lecteur Z : est disponible pointant sur la home directory

1.6 Gestion des scripts de démarrage

Samba permet également de centraliser la gestion des scripts de démarrage des sessions Windows (notamment pour créer les unités réseau).

Pour cela, il faut ajouter dans smb.conf :
`logon script = %U.bat` (permet de faire un script par utilisateur -%U=nom utilisateur- Il est également possible d'affecter tout le monde au même script en donn
ant un nom simple)
`logon home = \\%L\%U` (Indique le « home directory » de l'utilisateur – facultatif).

Il faut également activer le partage netlogon :

```
[netlogon]
path = /home/samba/netlogon
guest ok = no
writable = no
share modes = no
```

Il faut ensuite écrire un script « .bat » qui peut être par exemple :
`net use h : /home` (affecte le « home directory » au lecteur h :).

1.7 Administration du serveur par interface Web

Samba permet donc de gérer à bon compte un réseau de machines Windows.
Néanmoins, l'administration reste complexe (ajout des utilisateurs Unix en mode texte, puis validation dans Samba ; ajout manuel des comptes machines …).

Pour améliorer cela, il existe un outil d'administration graphique et en mode Web : **Webmin**.

Pour l'utiliser, il faut ajouter les packages `webmin, webmin-core et webmin-samba`.

Il est préférable d'avoir un navigateur (comme firefox) sur le serveur :
Lancer le navigateur et appeler la page https://localhost:10000.
Il est ensuite possible de configurer les machines qui peuvent utiliser Webmin pour l'utiliser à distance.
Si le serveur n'a pas d'interface graphique, il faut autoriser manuellement les autres machines à utiliser Webmin pour s'y connecter directement d'une autre machine :
Modifier le fichier `/etc/webmin/miniserv.conf` et enlever la ligne `allow=127.0.0.1`.

Une machine du réseau pourra donc accéder à l'administration en ssl (https://) sur le port 10000 (avec le mot de passe root) ; Il est même possible d'automatiser la synchronisation entre utilisateurs Unix et samba.

13. NIS (Network Information System)

Ce logiciel est adapté pour centraliser l'administration de machines linux dans un environnement relativement simple (quelques dizaines de machines maximum).

Il s'agit d'une simple mise en commun des fichiers de configuration comme /etc/passwd ou /etc/hosts en utilisant les RPC (remote procedure call) au même titre que NFS.

A l'origine appelé "Yellow page" et développé par SUN (d'où les commandes yp*).

Cela nécessite la mise en place d'un serveur NIS, puis la configuration des clients :

2.1 Installation du serveur NIS

- Ajuster au besoin le hostname (fichier **/etc/hostname**). Par exemple: serv-nis.
- Mettre à jour le fichier **/etc/hosts** pour préciser le nom du serveur (selon son adresse IP et non 127.0.0.1) et retirer le nom (hostname) de la ligne 127.0.0.1. Ajouter par exemple :
  ```
  192.168.1.12 serv-nis.nisdomain serv-nis
  ```
- Installer les packages **netbase, portmap, nis**. Lors de l'installation, il est demandé un nom de domaine nis. Le plus simple est de remettre le même nom de domaine que dans le fichier hosts soit dans l'exemple **nisdomain.** Cette information est copiée dans le fichier **/etc/defaultdomain**.
- Mettre à jour le fichier **/etc/default/nis** pour déclarer la machine comme serveur
  ```
  NISSERVER = master
  NISCLIENT = false
  ```
- Arrêter et redémarrer le service (**/etc/init.d/nis stop**, **/etc/init.d/nis start**)
- Enfin initialiser le service par **/usr/lib/yp/ypinit –m**

Cette configuration de base exporte les users (**passwd, shadow**) et les groups (**group, gshadow**) dont l'identifiant est >= à 1000. Toute modification à ces fichiers (directement ou au travers de commandes) nécessitera la mise à jour des bases NIS en lancant la commande **make** dans le répertoire **/var/yp**.

D'autres possibilités sont offertes en modifiant le fichier **/var/yp/Makefile** et **/etc/default/nis** et en consultant **/usr/share/doc/nis/nis.debian.howto**

2.2 Installation du client NIS

- Ajuster au besoin le hostname (fichier **/etc/hostname**). Par exemple: clt-nis. Reporter cette modif dans le ficier hosts ligne 127.0.0.1.
- Installer les packages **netbase, portmap, nis**. Lors de l'installation, il est demandé un nom de domaine nis. Mettre le même nom que pour le serveur (dans l'exemple : **nisdomain).** Cette information est copiée dans le fichier **/etc/defaultdomain**.
- Si le serveur est hors du réseau local, il est nécessaire de configurer le fichier /etc/yp.conf et d'indiquer manuellement le nom ou l'adresse des serveurs NIS.
- Arrêter et redémarrer le service (**/etc/init.d/nis stop**, **/etc/init.d/nis start**).

- Vérifier le contenu du fichier `/etc/nsswitch.conf` pour définir le comportement de chaque fichier:

```
passwd:   compat
group:    compat
shadow:   compat
…
netgroup: nis
```

Le mode compat signifie que l'on utilise à la fois les fichiers locaux et les fichiers du serveurs selon les instructions fournies dans les fichiers locaux (lignes + / -).

- Ajouter au minimum :

la ligne `+::::::` dans le fichier `/etc/passwd`
la ligne `+:::` dans le fichier `/etc/group`
la ligne `+:::::::::` dans le fichier `/etc/shadow`
la ligne `+:::` dans le fichier `/etc/gshadow`

Cela signifie "*Importer toutes les lignes mises à disposition par NIS*". Il est possible de réaliser des filtrages selon n'importe quel critère.

- Le client NIS est prêt!

2.3 Utilisation du service "Automount"

Automount/autofs permet (entre autres) de déporter le home directory des utilisateurs sur un serveur. Couplé à NIS, cela reproduit les "profils intinérants" de Microsoft. Ce service est basé sur NFS

Installation coté serveur :

- Installer et configurer NFS afin de rendre disponible le répertoire global des "home directories" (par exemple `/home/nis` qui devra être créé sur le serveur). Ne pas oublier de lancer le service NFS.
- Créer les fichiers `/etc/auto.master` et `/etc/auto.home` qui seront utilisés par les clients pour le service autofs :

auto.master doit contenir :

```
/home/nis       /etc/auto.home
```

auto.home doit contenir :

```
* -fstype=nfs,rw,soft,intr serv-nis:/home/nis/&
```

Cela signifie : auto.master prend en charge les montages en /home/nis et renvoie à /etc/auto.home. Ce dernier indique que pour tout utilisateur il faut monter un répertoire /home/nis/nomuser.

Dans l'exemple, le nom serv-nis devra être résolu au niveau du client. Si c'est impossible, il faut utiliser l'adresse IP.

- Retourner dans le Makefile (/var/yp) de NIS pour indiquer la prise en charge de ces 2 nouveaux fichiers :

```
# If you don't want some of these maps built, feel free to comment
# them out from this list.

ALL =   passwd group hosts rpc services netid protocols netgrp
ALL +=  auto.master auto.home
```

- Relancer `/usr/lib/yp/ypinit -m`

Installation coté client :

- Installer le package `autofs.`
- Modifier le fichier `/etc/auto.master` qui doit contenir :

```
+auto.master
```

  Ce qui signifie que ce fichier est pris en charge par NIS; c'est donc le contenu de celui du serveur qui sera utilisé. :

- Relancer le service autofs

```
/etc/init.d/autofs restart
```

- Il reste à faire un test de connexion: Le répertoire courant de l'utilisateur doit être situé sur le serveur.

14. LDAP (Lightweight Directory Access Protocol)

Une base LDAP est une base de données où les informations sont enregistrées de manière hiérarchique sous forme d'arbre et non sous forme tabulaire.

```
dc:          com
               |
dc:         genfic          (Organisation)
           /       \
ou:    people    servers     (Unités organisationelles)
       /    \      ..
cn: ..    john             (Données)
```

Une base LDAP est optimisée pour la lecture d'un nombre important de petits enregistrements et convient donc parfaitement pour stocker des annuaires ou des profils utilisateurs.

Chaque objet contient plusieurs **attributs** (obligatoire ou facultatifs) = informations. Les objets et les attributs sont normalisés dans des **schémas** pour assurer les échanges entre les logiciels.

Chaque donnée enregistrée dans la base est identifiée par son **DN (Distinguished Name)**. Ce DN est comparable au chemin complet d'un fichier. Exemple ci dessus:

cn=john,ou=people,dc=genfic,dc=com

2.1 Installation et configuration du serveur LDAP (Open-Ldap).

- Installer **slapd ldap-utils** : Il faut donner un nom de domaine de type DNS (domain.com), un nom à l'organisation et un mot de passe pour l'administrateur de cette base. Ces informations sont enregistrées dans `/etc/ldap/slapd.conf.`
- La gestion en ligne de commande d'une base LDAP étant limitée et complexe, il est vivement conseillé d'installer un module de gestion en mode graphique comme **php-ldapadmin** (➔ php, apache) ou **gq**. Debian propose un package "**gq**" directement utilisable.
- Une fois installé, utiliser la commande `gq` pour lancer l'utilitaire (interface graphique obligatoire!).
- Dans **file/préferences/servers**, choisir **localhost**, puis **Edit**. Dans l'onglet **detail**, ajouter à la ligne **bind DN** le DN de l'administrateur (**cn=admin,dc=domain,dc=com**) puis le mot de passe (ou cocher "ask password on first connect"). Cela permet de se connecter à la base ldap en temps qu'administrateur et ainsi de pouvoir la modifier.
- Dans l'onglet "**browse**" de la fenêtre principale, il doit être possible de parcourir la base ldap qui ne doit contenir au départ que l'administrateur de la base.
- Il est également possible de vérifier le fonctionnement de la base avec la commande :

```
ldapsearch -x -b "dc=domain,dc=com" "objectclass=*"
```

2.2 Configuration d'un poste client pour accepter un authentification à partir de la base ldap

- Installer **libpam-ldap et libnss-ldap** : Il faut donner l'adresse IP du serveur LDAP (`host 192.168.1.2`), le DN de la base (`base dc=domain,dc=com`), la version ldap utilisée (`ldap_version 3`), et le nom de l'administrateur (`root_bin_dn cn=admin,dc=domain,dc=com`). Ces informations sont enregistrées dans `/etc/pam_ldap.conf`.
- Modifier le fichier `/etc/pam.d/common-auth` et ajouter la ligne

`auth sufficient pam_ldap.so`

Pour indiquer que pam doit utiliser l'authentification ldap.

- Modifier le fichier `/etc/nsswitch.conf` et ajouter le mot `ldap` à `passwd, group, shadow` pour indiquer au système d'utiliser la base ldap pour l'authentification.
- Pour tester, il est nécessaire de créer un utilisateur dans la base ldap. Utiliser pour cela l'outil gc et créer un nouvel enregistrement avec au moins les caractéristiques suivantes :
  - **DN : cn=user,dc=domain,dc=com**
  - **Object classes : top, account, posixAccount, shadowAccount**
  - **userid : user**
  - **cn : user**
  - **uidnumber : 1010 (par exemple)**
  - **gidnumber : 1010 (par exemple)**
  - **homedirectory : /home/user (par exemple)**
  - **user password**
  - **login shell : /bin/bash**
- Cet utilisateur doit être capable de se connecter sur le client (avec probablement un message d'erreur concernant son répertoire de base!).

Beaucoup d'autres possibilités existent avec une base ldap et les utilitaires clients pam et nss mais leur mise en œuvre nécessite de connaître les "objects classes" à configurer dans la base ldap et les fichiers de configuration à personnaliser au niveau du client (principalement dans le répertoire /etc/pam.d).

MODULE 08 : LINUX –Compléments (distribution DEBIAN)

15. Démarrage d'un système Linux.

Lors de l'installation, nous avons choisi un utilitaire de boot (en général GRUB ou LILO). Ces utilitaires sont configurables par les fichier **/boot/grub/menu.lst** ou /etc/lilo.conf. Cela permet de gérer des options de lancement du noyau.

Le gestionnaire de boot lance donc le noyau qui à son tour va lancer le premier processus appelé init. Ce programme est recherché aux emplacements suivants : **/sbin/init, /etc/init, /bin/init, /bin/sh**.
Le fonctionnement du processus init est configuré par le fichier **/etc/inittab**. Ce fichier contient notamment le programme à lancer selon le **niveau d'exécution**.

Le niveau d'exécution ou **mode d'exécution** permet de gérer des modes de démarrages différents sur un système GNU/Linux.
Bien que ces modes soient entièrement paramétrables, il est usuel de trouver les cas suivants :

Numéro	Désignation	Description
0	**Arrêt**	Passer dans ce niveau provoque un arrêt de la machine.
1	**Maintenance**	On a directement accès à un shell, mais quasiment aucun service n'est lancé. Utile pour le dépannage en cas de problème important.
2	**Multi-utilisateurs simple**	Plusieurs utilisateurs peuvent se connecter en mode texte. Mais les services sont limités (souvent pas de réseau par exemple).
3	**Multi-utilisateurs complet**	Tous les services nécessaires sont démarrés et plusieurs utilisateurs peuvent se connecter en mode texte.
4	**Mode utilisateur**	Généralement non utilisé, il peut être librement utilisé.
5	**Graphique**	Identique au mode 3, mais les utilisateurs peuvent se connecter en mode graphique et disposer d'un gestionnaire de fenêtre.
6	**Redémarrage**	Passer dans ce niveau redémarre la machine.

DEBIAN n'utilise pas par défaut les niveaux 3, 4 et 5 c'est à dire que même si l'on met en place l'environnement graphique le mode actif reste le 2.

La commande `telinit` permet de modifier le mode d'exécution. la commande `runlevel` permet de connaître le mode actuel (et le précédent).

Dans le fichier inittab de Debian, on trouve les lignes suivantes :

```
id:2:initdefault:
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

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

Pour chacune de ces lignes, le premier champ représente un identifiant, le second le mode d'éxécution, le troisième l'action et le quatrième le programme à lancer.

On voit donc que c'est toujours le programme **/etc/init.d/rc** qui est utilisé avec un paramètre représentant le mode d'exécution. La première ligne (initdefault) indique le mode utilisé par défaut au lancement de la machine.

Le programme **rc** fonctionne de la manière suivante :

- ⇨ Il existe un répertoire par mode d'exécution dans /etc avec pour nom rcX.d (où X est le numéro de mode d'exécution).
- ⇨ Chacun de ces répertoires contient des fichiers commençant par "K" ou "S" (K signifie Kill, S signifie Start). Il s'agit en fait de liens vers les scripts de démarrage (et d'arrêt) de chaque service (en général dans le répertoire /etc/init.d).
- ⇨ Le programme rc lance chaque fichier "K" pour arrêter le service correspondant et chaque fichier "S" pour démarrer le service correspondant.

Par exemple, le contenu du répertoire rc2.d de Debian de base (caractère) est le suivant :

```
.S10sysklogd -> ../init.d/sysklogd          ➔ Logs systèmes
 S11klogd -> ../init.d/klogd                ➔ Logs kernel (noyau)
 S14ppp -> ../init.d/ppp                    ➔ Connexion ppp (inutile!)
 S18portmap -> ../init.d/portmap            ➔ Mappage ports pour RPC
 S20exim4 -> ../init.d/exim4                ➔ Serveur mail
 S20inetd -> ../init.d/inetd                ➔ Services IP
 S20lpd -> ../init.d/lpd                    ➔ Spooler d'impression
 S20makedev -> ../init.d/makedev            ➔ Prise en charge cartes dvb
 S20ssh -> ../init.d/ssh                    ➔ Serveur ssh
 S21nfs-common -> ../init.d/nfs-common      ➔ Services pour NFS
 S89atd -> ../init.d/atd                    ➔ File d'attente de travaux
 S89cron -> ../init.d/cron                  ➔ Planificateur de taches
 S99rmnologin -> ../init.d/rmnologin        ➔ marque la fin du boot
 S99stop-bootlogd -> ../init.d/stop-bootlogd ➔ marque la fin du log de boot.
```

Pour contrôler les services lancés au démarrage, il "suffit" donc de gérer les liens symboliques présents dans les répertoires rcX.d. Par exemple, si on ne désire pas activer le serveur ssh au démarrage, il suffit de supprimer le fichier S20ssh dans le répertoire /etc/rc2.d. Comme il s'agit d'un lien, il n'y a pas grand risque, il sera toujours possible de le recréer. La commande `update-rc.d` permet de simplifier la création de ces liens.

16. Environnement des utilisateurs.

Lorsque l'on utilise le shell bash, ce qui correspond à la plupart des cas, l'environnement utilisateur est défini dans les fichiers suivants :

/etc/profile et **/etc/bash.bashrc** : Scripts de démarrage généraux

~/.bash_profile et **~/.bashrc** : Scripts de démarrage pour chaque utilisateur.

Ces scripts contiennent entre autres :

- La définition des **variables d'environnement** :
  Une variable d'environnement est une variable commune à un shell. Cela permet de stocker des informations et de les utiliser ensuite. Par exemple la variable PATH contient le liste des répertoires où sont cherchées les commandes passées. Cette variable est utilisée par l'interpréteur de commandes.

  Pour définir une variable d'environnement : `VAR="Bonjour"` définit la variable VAR et lui affecte la valeur "Bonjour".
  Pour utiliser une variable d'environnement : `$VAR` fait référence à la variable VAR
  Pour afficher le contenu : `echo $VAR`
  Une variable d'environnement est donc valide pour un shell donné. Pour étendre sa validité aux shells "fils" (lancés à partir du premier shell), il faut utiliser la commande `export`

- La définition des **alias** :
  Un alias permet de créer des raccourcis de commande en fixant les options courantes. par exemple, il est très courant de créer l'alias `ll` qui correspond à la commande `ls -l`.