Administration Linux

Gestion des Processus

Qu’est ce qu’un processus

- Un processus est une instance d'un programme en cours d'exécution sur un ordinateur.
- Chaque processus a un numéro d'identification unique (PID), une quantité de mémoire allouée et des informations sur l'état de l'exécution
- Un processus peut lancer un autre processus : dans ce cas on par lera de processus "père" et de processus "fils".
- Plusieurs processus peuvent s’exécuter en même temps

Qu’est ce qu’un processus

- Dans Linux, les processus sont gérés par le noyau, qui est le cœur du système d'exploitation.
- Lorsqu'un utilisateur lance un programme, le noyau crée un processus pour exécuter ce programme.
- Le noyau attribue ensuite un certain nombre de ressources, telles que de la mémoire et des threads d'exécution, au processus afin qu'il puisse fonctionner correctement.

Qu’est ce qu’un processus

- Le noyau gère également les processus en exécution en leur attribuant des temps d'exécution sur le processeur,
- Il les met en attente lorsque des ressources sont manquantes
- Il les termine lorsqu'ils sont terminés ou lorsqu'ils sont arrêtés par l'utilisateur.

systemd et processus init

- Le programme init, est père de tous les autres processus
- Ce premier processus a donc un PID à 1.
- Dans de nombreuses distributions init a été remplacé par le programme de gestion systemd,

Processus: Commandes de base

- Il existe de nombreuses commandes et outils en ligne de commande pour afficher et gérer les processus dans Linux, comme: ps, top, kill, nice , etc
- **ps** : qui affiche la liste de processus, par défaut elle affiche les processus lancés par l'utilisateur actuel , mais elle on peut utiliser des options pour afficher tous les processus. Par exemple la commande ps -aux affiche tous les processus du système avec des informations détaillées
- **top** : Affiche les processus en cours d'exécution sur le système, classés par utilisation des ressources, top actualise constamment l'affichage pour montrer les changements en temps réel.

Processus: Commandes de base

- **Kill** : Envoie un signal à un processus pour le tuer. On peut utiliser la commande **kill** en spécifiant le numéro d'identification du processus (PID) qu'on veut tuer. Par exemple pour la commande **kill 1234** tue le processus ayant pour identifiant 1234.

on peut utiliser la commande avec d'autres options pour envoyer d'autres types de signaux aux processus comme **SIGSTOP** pour mettre un processus en attente ou **SIGCONT** pour le reprendre

Processus: Commandes de base

- **nice :** : Modifie la priorité d'un processus en cours d'exécution. Les processus avec une priorité plus élevée obtiendront plus de temps d'exécution sur le processeur par rapport aux processus avec une priorité plus faible. On peut utiliser nice en spécifiant le nom du programme et un nombre de priorité de -20 à 19. Par exemple la commande **nice –n 10 myprogram** exécute le programme myprogram avec la priorité 10.

Les services

- Un service est un programme qui s'exécute en arrière-plan sur un ordinateur, généralement pour effectuer une tâche spécifique ou pour fournir une fonctionnalité à d'autres programmes.
- Les services sont souvent utilisés pour exécuter des fonctionnalités en arrière-plan, telles que la connexion à un réseau ou la surveillance d'un système
- Les services sont gérés par le système de gestion des services, qui est généralement le daemon **systemd.**
- **systemd** est responsable de lancer les services au démarrage du système et de les maintenir en cours d'exécution en arrière-plan.

Les services : Commande systemtctl

systemctl est un utilitaire qui permet de gérer systemd et toutes les opérations faites par **systemd.**

Avec **l**a commande **systemctl** on peut:

- Lancer un service:

  systemctl start nom_service

- Arrêter un service:

  systemctl stop nom_service

Les services : Commande systemtctl

- Redémarrer un service:

  systemctl restart nom_service

- Vérifier l'état d'un service

  systemctl status nom_service

- Activer le lancement d'un service au démarrage du système:

  systemctl enable nom_service

- Désactiver le lancement d'un service au démarrage du système

  systemctl enable nom_service

Les services: Commandes service

- **service :** Cette commande permet de gérer les services avec le système de gestion des services précédent. L'option **status** permet de vérifier l'état d'un service.

  service nom_service status

Les services: Commandes chkconfig

- **chkconfig**: Cette commande permet de configurer les services pour qu'ils démarrent au démarrage du système. On peut utiliser **chkconfig** pour activer ou désactiver les services pour différents niveaux de démarrage (les runlevels on les verra plus tard).

Les deamons

- Pour les systèmes Unix, Les deamons ce sont des programmes qui s'exécutent en arrière plan, ils ne sont pas associés aux terminaux de la machine
- Un deamon a pour rôle d'offrir un service à d'autres processus ou à des utilisateurs (qui peuvent être distants)
- Le fonctionnement d'un deamon est géré par un script de démarrage ou par systemd

Processus et services en résumé

- En résumé, les processus sont des programmes en cours d'exécution et les services sont des programmes qui s'exécutent en arrière-plan et qui fournissent des fonctionnalités spécifiques à d'autres programmes ou à l'utilisateur.

Gestion des paquets

Paquetage

- Un paquetage contient :
  - les fichiers (exécutables, librairies, fichiers de configuration, ressources, etc.) nécessaires pour exécuter un programme ;
  - un ensemble de scripts qui configurent le programme automatiquement après son installation ;
  - les informations sur le propriétaire et permissions d'accès de chaque fichier ;
  - des informations optionnelles sur les paquetages dépendants et les services fournis ;
  - une description du paquetage.
- L'ensemble des informations de tous les paquetages installés sont stockés dans une base de données gérée par l'utilitaire **rpm** (Redhat Package Management) pour les distributions basées sur RedHat et les outils **dpkg** et **apt** pour la distribution Debian.

rpm

Le nom d'un fichier de paquetage est constitué du nom du paquetage, du numéro de version du logiciel et du numéro de version du paquetage séparés par des caractères « - ».

Par exemple : openssl-0.9.7a-35.rpm

rmp – Exemple d’utilisation

Exemples d'utilisation de rpm :

- installation d'un paquetage : rpm -i fichier.rpm
- mise-à-jour d'un paquetage : rpm -U fichier.rpm
- suppression d'un paquetage : rpm -e paquetage
- vérification d'un paquetage : rpm -V paquetage

rmp – Exemple d’utilisation

- interrogation sur les paquetages installés :
  - liste des fichiers d'un paquetage : rpm -ql paquetage
  - paquetage contenant un fichier : rpm -qf /chemin/vers/fichier
  - informations sur un paquetage : rpm -qi paquetage
  - liste triée de tous les paquetage installés : rpm -qa |sort
- interrogation sur un fichier de paquetage :
  - liste des fichiers : rpm -qpl fichier.rpm
  - informations : rpm -qpi fichier.rpm

rpm – Gestion des signatures GPG

rpm supporte désormais la signature des paquetages. Cette technique permet de garantir qu'un paquet n'a pas subi de modification et que la personne qui l'a construit est bien celle qu'elle prétend être.

Il faut pour cela importer les clés publiques **GPG** des personnes ou organisations qui réalisent les paquetages devant être installés avec l'option **--import** de rpm.

L'option **--checksig** permet de vérifier la signature d'un paquetage avant de l'installer.

rpm – Gestion des signatures GPG

Par exemple :

rpm --checksig webmin-1.150-1.noarch.rpm

webmin-1.150-1.noarch.rpm: md5 (GPG) PAS OK (CLES MANQUANTES: GPG#11f63c51)

wget http://www.webmin.com/jcameron-key.asc

.../...

rpm --import jcameron-key.asc

rpm --checksig webmin-1.150-1.noarch.rpm

webmin-1.150-1.noarch.rpm: md5 gpg OK

dpkg – Gestion des paquetages Debian

Le nom d'un fichier de paquetage est constitué :

- du nom du paquetage suivi ;

- du numéro de version du logiciel et du numéro de version du paquetage ;

- de l'architecture cible ;

- de l'extension .deb.

Par exemple : **openssl_0.9.7e-3sarge1_i386.deb**

Exemples d'utilisation de **dpkg** :

dpkg – Gestion des paquetages Debian

installation d'un paquetage : dpkg -i fichier.deb

- suppression d'un paquetage : dpkg -P paquetage
- liste des paquetages installés : dpkg –l
- afficher des informations sur un paquetage : dpkg -I fichier.deb
- afficher la liste des fichiers installés par un paquetage : dpkg -L paquetage

apt – Dépôts de logiciels

- Un dépôt Debian est un ensemble organisé et indexé de paquetages Debian et que l'infrastructure **apt** (Advanced Packaging Tool) de gestion des paquetages Debian sait utiliser pour installer des applications.
- Un dépôt Debian peut être stocké sur différents supports :
  - CD-ROM ;
  - DVD-ROM ;
  - un répertoire sur un système de fichiers ;
  - un emplacement sur le réseau accessible en HTTP ou FTP ;
  - etc.

apt – Dépôts de logiciels

Il existe plusieurs types de dépôts Debian :

- les dépôts officiels de la distribution ;
- le dépôt officiel contenant les correctifs de sécurité ;
- le dépôt « volatile » ;
- des dépôts Debian non officiels.

Système de fichiers et partitions disque

Les systèmes des fichiers

Un système de fichiers est une collection organisées de répertoires et fichiers selon un format donné. Linux reconnaît en lecture et en écriture plusieurs dizaines de format de systèmes de fichiers.

Selon le type de système de fichiers, le support servant à la persistance des données peu prendre l'une des formes suivantes (les exemples donnés entre parenthèses ne sont pas

exhaustifs) :

- partition sur un périphérique géré par un contrôleur de disque local (disque, contrôleur RAID matériel) ;
- périphérique amovible (disquette, CD-ROM) ;
- ressource mise à disposition sur le réseau (tel que serveurs NFS ou SMB) ;
- mémoire vive (utilisée par le système de fichiers tmpfs) ;
- espace virtuel (ce qui est le cas de systèmes de fichiers tels que proc, sysfs, usbdevfs ou devpts) ;
- périphérique de type bloc émulé (par exemple fournis par ramdisk, loop, NBD, LVM, RAID logiciel).

Créer un système de fichiers

Le système de fichiers le plus utilisé est ext3. Pour créer un nouveau système de fichiers et l'attacher de manière permanente à l'arborescence, il est nécessaire de :

1) configurer un périphérique de type bloc ;

2) créer le système de fichiers avec la commande **mkfs** ;

3) choisir un point de montage dans l'arborescence ;

4) attacher le système de fichiers au point de montage avec la commande **mount** ;

5) ajouter une ligne au fichier **/etc/fstab** afin que le système de fichiers soit attaché au point de montage à chaque démarrage du système.

Créer un système de fichiers

Les actions à mener pour réaliser la première étape dépendent du type de support utilisé pour recevoir le système de fichiers.

La plupart du temps il s'agit d'une partition sur un disque local qui sera créée avec la commande **fdisk**.

Les partitions Linux ont un identifiant système 0x83. C'est l'identifiant qui est positionné par défaut lorsqu'une partition est créée avec la commande **fdisk** de Linux.

Créer un système de fichiers

Exemple de création d'une partition (hda9) et d'un système de fichiers :

fdisk /dev/hda

[Utiliser la commande "p" pour afficher la table des partitions, "n" pour créer une partition et "w" pour écrire les modifications sur le disque. Un redémarrage peut s'avérer nécessaire.]

mkfs -t ext3 /dev/hda9 .../...

mount -t ext3 /dev/hda9 /opt

echo /dev/hda9 /opt ext3 defaults 1 2 >> /etc/fstab

Créer un système de fichiers

Exemple de création d'un système de fichiers dans un « fichier image »:

dd if=/dev/zero of=/var/local/myfs bs=1024 count=102400 .../...

mkfs -t ext3 /var/local/myfs

[Une confirmation est demandée car "/var/local/myfs" n'est pas un fichier spécial de type bloc.] .../...

mkdir /mnt/tmp

mount -o loop -t ext3 /var/local/myfs /mnt/tmp

echo /dev/hda9 /opt ext3 loop 1 2 >> /etc/fstab

Réparer un système de fichiers

La commande **fsck** permet de vérifier l'intégrité d'un système de fichiers et, le cas échéant, d'effectuer une tentative de réparation. Le système de fichier ne doit pas être monter. Pour cette raison ces tests sont souvent effectués en runlevel 1 :

```
# init 1
# umount /usr
# fsck -t ext3 -f /dev/hda7
# mount /usr
# init 5
```

Réparer un système de fichiers

Un autre moyen consiste à forcer la vérification de l'intégrité de tous les systèmes de fichiers locaux au démarrage du système. Pour cela, il suffit de créer un fichier vide nommé forcefsck dans la racine de l'arborescence :

touch /forcefsck

init 6

Les blocs de données orphelins retrouvés lors de la réparation de systèmes de fichiers **ext2** ou **ext3** sont placés dans les répertoire **lost+found** présents dans la racine de chaque système de fichiers **ext2** ou **ext3**.

Disques et points de montage

- Les disques durs sont des fichiers situés dans /dev ;
- Pour avoir accès à leur contenu, il faut les monter, c’est-à-dire les accrocher à un répertoire appelé point de montage ;

Partitions

- En réalité, on ne monte pas directement un disque mais bien ses partitions ;
- Tout disque est partitionné : une partition est un morceau du disque sur lequel se trouve un système de fichiers ;
- Lorsqu'on veut accéder au contenu d'une partition, on doit monter cette partition — pas le disque ;
- Le fichier **/etc/fstab** contient les informations de montage sur les périphériques ;

Organisation d’un disque - Découpage logique de disque

Un disque peut être vu comme une longue bande d’espace de stockage découpée en cases pouvant contenir une quantité donnée d’informations. Le disque peut être utilisé tel quel comme espace de stockage, rien n’empêche de créer un système de fichiers sur un disque sans passer par l’étape de partitionnement.

Il est cependant important de donner une organisation logique à cet espace et aux systèmes de fichiers qu’il contiendra.

La partitionnement consiste en un découpage logique du disque.

Le disque physique, réel, est fractionné en plusieurs disques virtuels, logiques, les partitions.

Chaque partition est vue comme un disque indépendant et contient son propre système de fichiers.

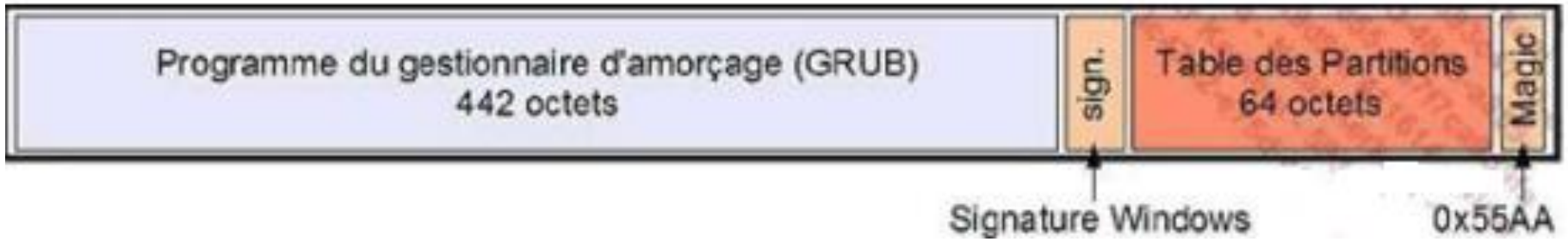
Organisation d’un disque - MBR

Le premier secteur est le MBR, Master Boot Record, ou zone d’amorce. D’une taille de 512 octets il contient:

- dans ses 444 premiers octets une routine (un programme) d’amorçage destiné soit à démarrer le système d’exploitation sur a partition active, soit un chargeur de démarrage (bootloader),
- puis 4 octets d’une signature optionnelle (Windows),
  2 octets nuls, et
- les 64 octets suivants contiennent la table des quatre partitions primaires.
- Le tout finit par une signature 0xAA55 sur deux octets

Organisation d’un disque - MBR

Format d’un MBR



Organisation d’un disque – Les partitions

Une partition est un découpage logique du disque. Il en existe trois sortes :

- Les partitions primaires, au nombre de quatre, sont celles décrites dans le MBR.
- Les partitions étendues (primaires étendues), une seule par disque (bien que théoriquement il soit possible de créer des partitions étendues au sein d’autres partitions étendues).
- Les partitions ou lecteurs logiques.

Organisation d’un disque – Les partitions

- Un disque peut contenir jusqu’à 63 partitions en IDE, 15 en SCSI (c’est une limite de l’implémentation officielle du SCSI).
- Il s’agit d’une limite par disque, et pas du nombre total de partitions gérées par le système.
- Les partitions sont numérotées de 1 à n (15 ou 63).
- Une partition d’une valeur supérieure ou égale à 5 indique qu’il s’agit forcément d’une partition logique.
- Comme il ne peut y avoir que quatre partitions primaires, la dernière (la 4) est souvent créée comme étendue

Organisation d’un disque – Les partitions

On a donc :

- Partitions 1 à 3 : primaires
- Partition 4 : étendue
- Partitions 5 à n : logiques

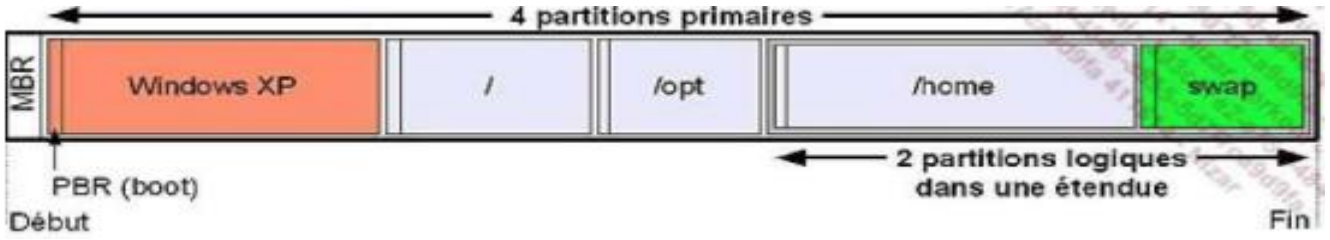
Organisation d’un disque – Les partitions

Le numéro de la partition apparaît à la suite du nom du fichier périphérique de disque :

- hda1 : première partition primaire du premier disque IDE
- hdb5 : cinquième partition, première partition logique du second disque IDE
- sda3 : troisième partition primaire du premier disque SCSI / libata
- sdc8 : huitième partition, soit quatrième partition logique du troisième disque SCSI/libata.

Organisation d’un disque – Les partitions

Description schématique d’un disque



Organisation d’un disque – EBR

Chaque partition étendue devant décrire les partitions logiques qu’elle contient, elle doit aussi disposer d’une table de partition.

L’EBR (Extended Boot Record) reprend la structure du MBR sauf qu’il n’y a que deux enregistrements possibles dans la table des partitions.

- Le premier enregistrement indique la position et la taille d’une partition logique,
- le second enregistrement est vide si c’est la seule partition logique, ou pointe sur un autre EBR.
- Il peut donc y avoir plusieurs EBR dans une partition étendue.
- Il n’y a qu’une seule partition logique décrite par EBR.

Organisation d’un disque – PBR

- Le PBR (Partition Boot Record), aussi appelé VBR (Volume Boot Record) ou Partition Boot Sector est le premier secteur de chaque partition primaire ou logique.
- Il peut contenir une routine de démarrage d’un système d’exploitation, un chargeur de démarrage, ou rien du tout si la partition n’a pas vocation à être bootée.
- Quand le MBR ne contient pas de routine, le bios tente de démarrer et d’exécuter la routine du PBR de la partition marquée active

Organisation d’un disque – Type de partitions

- Chaque partition dispose d’un type permettant de déterminer son contenu. C’est un identifiant numérique codé sur un octet généralement présenté en hexadécimal.
- Le type de partition refléte le système de fichiers qu’elle contient, une partition de type 0x0c devrait contenir un système de fichiers de type FAT32 LBA (gros disques). Une partition de type 0x83 contient un système de fichiers Linux.

Manipulation des partitions

Les outils fdisk, cfdisk, sfdisk ou encore parted permettent de manipuler les partitions.

- **fdisk** est le plus ancien et le plus utilisé des outils de partitionnement.
- **cfdisk** est un peu plus « visuel » et s'utilise avec les flèches directionnelles. Il permet les mêmes opérations que **fdisk** mais de manière plus conviviale.
- **sfdisk** fonctionne en interactif ou non, est assez compliqué mais plus précis.
- **parted** permet des opérations très avancées sur les partitions comme par exemple leur redimensionnement. Il est soit interactif (c'est un interpréteur de commandes) soit scriptable.

Manipulation des partitions

- **Lister**

  C’est l’outil fdisk, à la fois le plus ancien et le plus standard, qui est généralement utilisé par les administrateurs et les ingénieurs système. **fdisk** se lance en tant que root.

  fdisk [-l] [disque]

  Exemple: fdisk –l /dev/sda

- **Supprimer**

  Pour supprimer une partition, lancez **fdisk** avec le disque en argument, utilisez la touche d (delete) puis, si plusieurs partitions sont présentes, le numéro de partition (sdbX, X étant le numéro). Si une seule partition est présente, elle est prise par défaut.

Manipulation des partitions

- **Créer**

  Pour créer une partition, lancez **fdisk,** utilisez la touche n (new). Vous devez ensuite choisir le type de partition : primaire ou étendu.

- **Sauver**

  Quitter fdisk en sauvant votre table des partitions avec la touche w (write). Fdisk écrit la nouvelle table des partitions dans le MBR et/ou les EBR.

Manipuler les systèmes de fichiers - Bloc

- Le bloc est l'unité de base, atomique, de stockage du système de fichiers. Un fichier occupe toujours un nombre entier de blocs.
- Si un fichier ne contient qu'un seul octet et qu'un bloc a une taille de 4096 octets, 4095 octets sont gâchés.
- C'est ainsi qu'il est possible de remplir un système de fichiers avec n fichiers de 1 octets, n représentant le nombre de blocs, alors que le volume total des données n'est que de n octets !
- Il est très important de faire attention à la taille de blocs surtout si les fichiers à stocker sont de petite taille.

Manipuler les systèmes de fichiers – Super Bloc

Chaque système de fichiers dispose d'au moins un « super bloc ». Un superbloc est une zone de méta-données qui contient plusieurs informations sur le système de fichiers :

- son type
- sa taille
- son état
- des informations (position) sur les autres zones de méta-données (autres superblocs, table d'inodes, etc.).

Manipuler les systèmes de fichiers – Table d'inode

Un inode est la contraction de index node, c'est-à-dire nœud d'index. C'est une structure de données contenant les informations décrivant et représentant un fichier. Ces informations sont appelées des attributs.

Chaque fichier dispose d'un numéro d'inode (i-number). Tous les inodes sont présents au sein d'une table d'inodes.

Cette table est généralement découpée en plusieurs morceaux répartis après chaque superbloc. Une table d'inode fait partie des méta-données.

Un fichier ne peut avoir qu'un seul inode.

Un inode est unique au sein d'un seul système de fichiers.

Chaque système de fichiers dispose d'une table d'inodes indépendante.

Manipuler les systèmes de fichiers – Table d'inode

Le contenu d'un inode varie d'un système de fichiers à un autre, chacun d'eux dispose au moins des attributs suivants pour chaque fichier :

- sa taille
- l'identifiant du périphérique le contenant
- son propriétaire
- son groupe
- son numéro d'inode
- son mode (ses droits) d'accès
- sa date de dernière modification d'inode (change time)
- sa date de dernière modification de contenu (modification time)
- la date de dernier accès (access time)
- un compteur de hard links (liens physiques ou durs)

Manipuler les systèmes de fichiers – Table d'inode

- Un inode ne contient pas le nom du fichier.
- on obtenir quelques informations sur un inode avec la commande **stat** :

  sudo stat MyFile.doc

- L'inode contient aussi des champs d'adresses généralement répartis en deux types :
  - des adresses pointant sur les premiers blocs de données du fichier,
  - des adresses pointant sur des blocs contenant d'autres champs d'adresses,

  et dans ce dernier cas de manière récursive (adresses d'adresses pointant elles-mêmes sur d'autres adresses)

Manipuler les systèmes de fichiers – Table catalogues

- Un inode ne contenant pas le nom du fichier, celui-ci est placé ailleurs, dans une table de catalogue.
- Cette table n’est rien d’autre qu’un répertoire. Un répertoire contenant un liste de fichier, et un fichier étant représenté par un inode, chaque nom de fichier est associé au sein du répertoire à son inode.

Manipuler les systèmes de fichiers – hard link

- Un hard link permet d'ajouter une référence sur un inode. Le hard link rajoute une association dans une table catalogue. Les droits du fichier ne sont pas modifiés.
- Un hard link ne permet pas d'affecter plusieurs noms à un même répertoire, et ne permet pas d'effectuer des liens depuis ou vers un autre système de fichiers.
- La commande ln permet de créer un hard link

  Exemple:

  ln fic1 fic2

La journalisation

- Les systèmes de fichiers actuels disposent souvent de mécanismes permettant de garantir au mieux l'intégrité des données.
- Le système le plus courant est la journalisation
- Le système de fichiers maintient à jour un journal, généralement d'une taille donnée et circulaire (les nouvelles informations finissent par écraser les anciennes) dans lequel il trace tous les changements intervenus avant de les effectuer réellement.
- En cas de coupure brutale, le système pointe les enregistrements du journal et vérifie si les opérations ont été effectuées, éventuellement il les rejoue.
- Le journal contient des opérations atomiques (n opérations indivisibles) la cohérence des données est assurée soit par complétion du journal quand c'est possible, soit par retour en arrière.
- La réparation est fiable et rapide.

Créer un système de fichiers

- La commande pour créer un système de fichiers est **mkfs**.
- **mkfs** appelle d'autres programmes en fonction du type de système de fichiers sélectionné.
- C'est **typefs** qui détermine le type de système de fichiers et donc le programme appelé. Il existe un programme par type de système de fichiers :
  - ext2 : mkfs.ext2
  - ext3 : mkfs.ext3
  - reiserfs : mkfs.reiserfs
  - vfat : mkfs.vfat (pour tous les formats FAT, mais mkfs.msdos existe)
  - ntfs : mkfs.ntfs

Accéder un système de fichiers

- La commande **mount** permet d'accéder aux périphériques de type blocs (les partitions) sur lesquels un système de fichiers existe.
- La commande **mount** attache le répertoire racine du système de fichiers à un répertoire pré-existant appelé point de montage (mountpoint).

Exemple

```
mount -t typefs -o options peripherique point_de_montage
```

Fichier /etc/fstab

- Le fichier /etc/fstab contient une configuration statique des différents montages des systèmes de fichiers.
- Il est appelé à chaque démarrage du système car c'est ici qu'on indique les périphériques et leurs points de montage.
- Il contient six champs

  peripherique point_de_montage typefs options dump fsck
- Les champs sont séparés par des espaces ou des tabulations.

Le swap

- Dans un environnement 32 bits un processus peut théoriquement accéder à 4 Go d'espace mémoire. Il dispose de 4Go de mémoire virtuelle, rien qu'à lui et à laquelle aucun autre processeur ne peut accéder.
- Dans la pratique il y a plusieurs freins à cette possibilité :
  - L'espace mémoire adressable d'un processus est partagé entre zone de code et zone de données dont la taille peut varier selon le noyau utilisé.
  - Les ordinateurs ne disposent pas tous de 4 Go de mémoire (bien qu'il soit courant de trouver des serveurs Linux disposant de 16, 32 ou même 64 Go de mémoire).
  - Tous les processus doivent se partager la mémoire de l'ordinateur

Le swap

- Le swap est donc un espace d’échange qui sert à étendre votre mémoire vive (RAM) dans un fichier ou une partition sur un disque dur.
- Dans les nouvelles distributions ubuntu un fichier swap, à la racine du système, remplace la traditionnelle partition swap

Les quotas disques

Les quotas permettent de poser des limites à l'utilisation de systèmes de fichiers. Ces limites sont de deux types :

- inodes : limite le nombre de fichiers.
- blocs : limite la taille disque

- Les quotas sont implémentés par système de fichiers individuel et pas pour l'ensemble des systèmes de fichiers.
- Chaque utilisateur peut être géré de manière totalement indépendante. Il en est de même pour les groupes.

Les quotas disques

On peut mettre en place deux limites dans le temps :

- Limite dure (**hard**) : quantité maximale d'inodes ou de blocs utilisés que l'utilisateur ou le groupe ne peuvent absolument pas dépasser. Dans ce cas, plus rien ne sera possible (création de fichier ou fichier dont la taille dépasse la limite).
- Limite douce (**soft**) : quantité maximale d'inodes ou de blocs utilisés que l'utilisateur ou le groupe peuvent temporairement dépasser. Dans ce cas, les créations et modifications seront possibles jusqu'à un certain point : limite dure et délai de grâce.
- Un délai de grâce est mis en place. Durant ce temps, l'utilisateur peut continuer à travailler sur le système de fichiers. Le délai dépassé, la limite douce devient la limite dure.

Les quotas disques

- Les quotas sont implémentés dans le noyau Linux et au sein des systèmes de fichiers. Pour les utiliser, les outils de quotas (packages quota) doivent être installés. Les manipulations suivantes sont effectuées sur un système de fichiers ext3.
- La commande **quotaon** démarre les quotas pour le système de fichiers indiqué
- La commande **quotaoff** stoppe les quotas
- La commande **edquota** permet d'éditer les quotas pour les utilisateurs ou les groupes.

Niveau d’exécution runlevel

runlevel

Ubuntu s’inspire des niveaux d’exécution (runlevel) de type System V Unix pour pouvoir configurer l’accès aux services.

Chaque niveau d’exécution se trouve dans un répertoire de type

/etc/rcN.d ou N identifie le numéro d’ordre.

runlevel

La distribution Ubuntu distingue les niveaux d’exécution suivants :

- 0 pour l’arrêt du système.
- 1 pour le mode single ou monoutilisateur.
- 2 à 5 pour un mode multiutilisateur.
- 6 pour le redémarrage du système.
  Par défaut, Ubuntu comporte un réglage sur le niveau 2.
- Le mode 1 se réserve plus particulièrement à la maintenance, comme le changement du mot de passe administrateur en cas d’oubli.

runlevel

- La commande telinit change le niveau d'exécution

  Exemple:

  telinit 3

- La commande **runlevel** montre le niveau d'exécution courant
- Le retour affiche 2 3, c'est àdire les niveaux précédents et actuels.
- Les scripts contenus dans les répertoires /etc/rcN.d sont pour la plupart des pointeurs sur des fichiers regroupés dans /etc/init.d.

Le planificateur de tache

Le planificateur de tache cron

- Le planificateur de tâche cron permet à chaque utilisateur de configurer l'exécution périodique de commandes via l'utilisation de la commande crontab.
- Pour planifier l'exécution périodique de tâches relatives à la gestion du système, il est préférable d'utiliser des répertoires suivants :

  /etc/cron.hourly pour une exécution toutes les heures ;

  /etc/cron.daily pour une exécution tous les jours ;

  /etc/cron.weekly pour une exécution hebdomadaire ;

  /etc/cron.monthly pour une exécution mensuelle ;

  et /etc/cron.d

Le planificateur de tache cron

- L'utilisation des quatres premiers répertoires est simple : il suffit d'y copier un fichier exécutable (ou de créer un lien) afin d'en activer l'exécution périodique.
- Le fichier /etc/crontab contrôle les jours et heures d'exécution des commandes présentes dans les répertoires /etc/cron.* (excepté /etc/cron.d).
- Les planifications créées par les utilisateurs sont stockées à raison d'un fichier pour chaque utilisateur dans le répertoire /var/spool/cron

Le planificateur de tache cron

- La commande: **ls –d cron.*** (sous /etc) affiche mes services systèmes qui sont programmés pas jour, par semaine, par mois et par année.
- La commande: **sudo crontab –l** affiche les jobs planifiés par le root
- La commande: **crontab –l** affiche les jobs planifiés par l'utilisateur
- La commande: **crontab –e** permet de créer un crontab

  Syntaxe du fichier crontab:

  Minute(0-59) heure(0-23) jourDumois(1-31) mois(1-12) jourDeSemaine(0-6)
- La commande **crontab –r** permet d'effacer le contab

Planifier des taches avec la commande at

- Il est possible de planifier des tâches grâce à la commande at, pour qu'elles s'exécutent ponctuellement et à des moments bien précis.
- La commande **at** sur fait appel au démon **atd** pour planifier les tâches.
- la commande **at** permet une planification unique d'une tâche, contrairement à **cron** qui permet de planifier des tâches récurrentes.
- La commande **atq** affiche la liste des taches (identifiées par des numéros)
- La commande **atrm numero_tache** suprime la tache dont le numéro est donné en argument.
- La commande at –V permet d'afficher la version de at et de vérifier si le paquet est installé ou non.