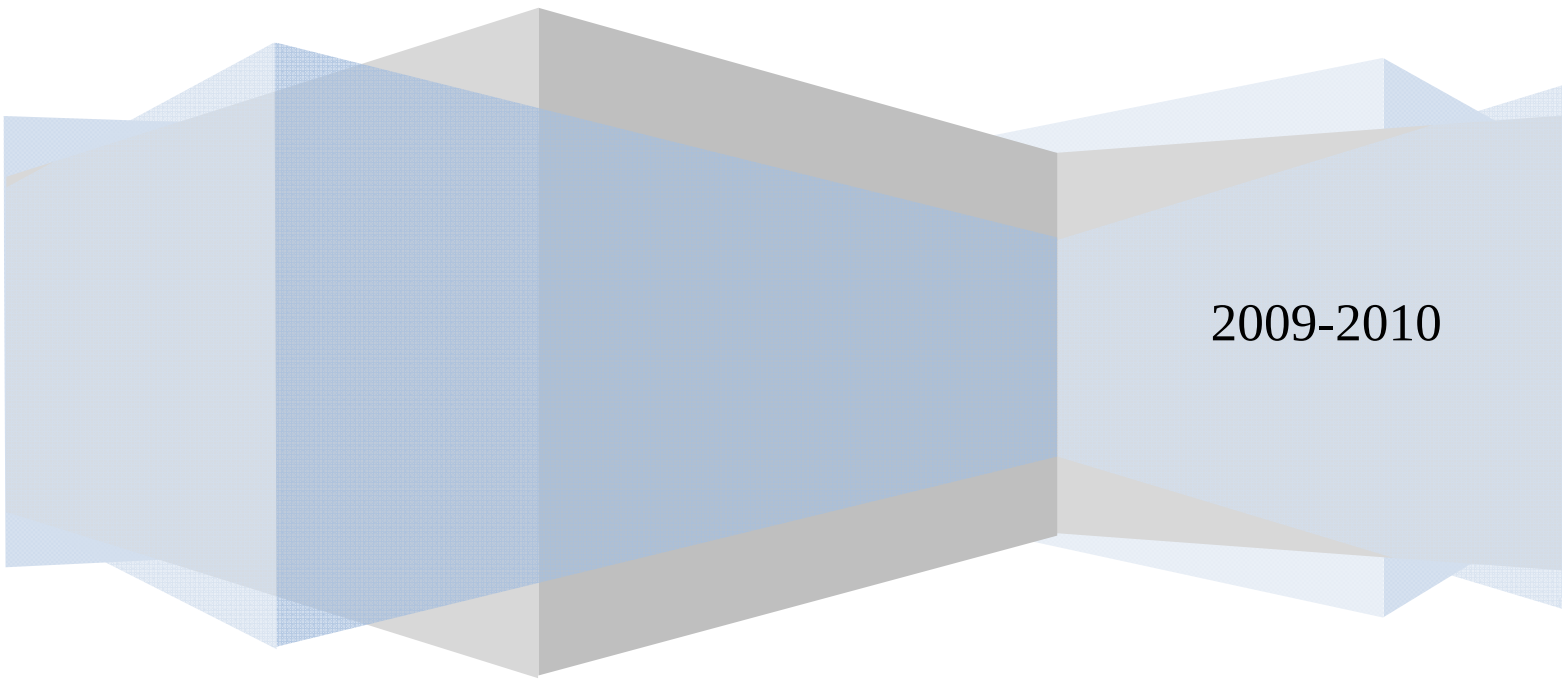
ISET Nabeul

Administration système Linux

Cours réalisé par :

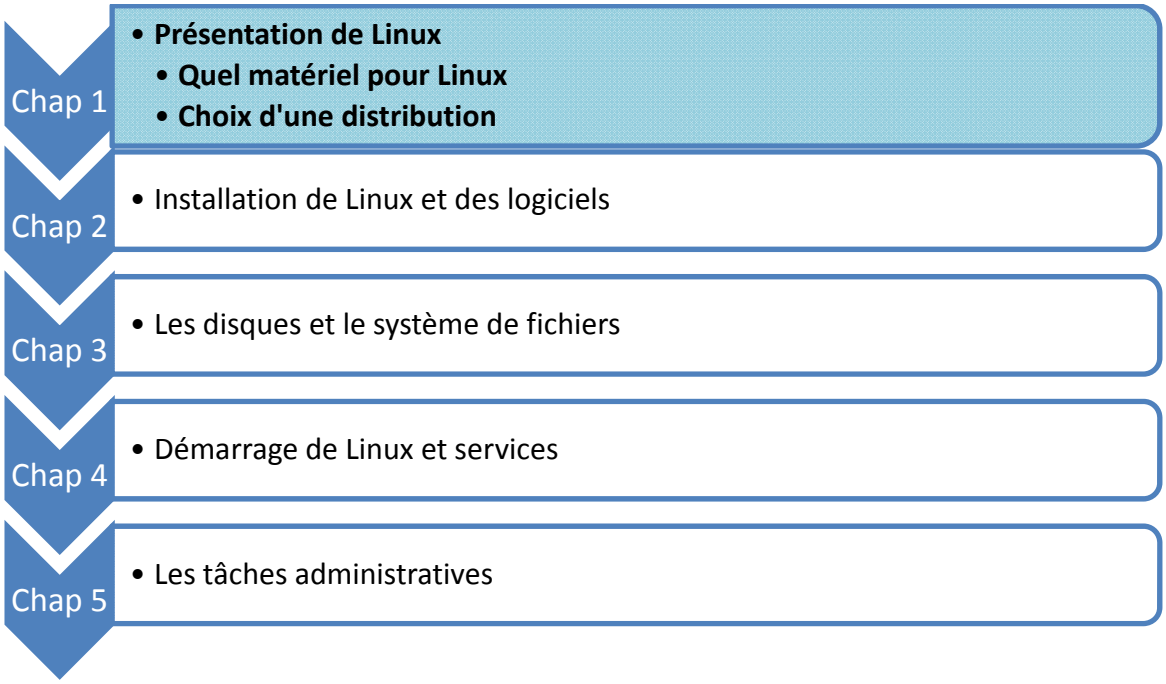
Elies Jebri

Technologue



2009-2010

Table des matières


Chap 1
• Présentation de Linux
• Quel matériel pour Linux
• Choix d'une distribution
Chap 2
• Installation de Linux et des logiciels
Chap 3
• Les disques et le système de fichiers
Chap 4
• Démarrage de Linux et services
Chap 5
• Les tâches administratives

Introduction

Installer Linux est très simple. Les tâches d'administration communes le deviennent aussi. La complexité du système est masquée par de nombreux outils, graphiques notamment, qui tendent à simplifier le travail des utilisateurs et des administrateurs. Cette simplicité apparente cache pourtant une réalité différente.

Chaque distribution est livrée avec une interface qui lui est propre. Les centres de contrôle de Redhat, Mandriva, openSUSE, Ubuntu, etc. sont tous différents. Il ne s'agit pas de se spécialiser dans l'une ou l'autre des interfaces. Toutes ces interfaces s'appuient sur les mêmes outils, ce sont des front-ends. Ils modifient les mêmes fichiers de configuration. Ces commandes et fichiers de configuration sont communs à l'ensemble des distributions. Plutôt que d'utiliser une interface qui risque d'être dépassée à la prochaine version, vous apprendrez directement à maîtriser les bases du système. Ainsi vous ne serez pas bloqué par votre dépendance à un outil spécifique.

Unix est avant tout un système d'exploitation qui est installé sur des serveurs, mais aussi grâce à Linux, à la maison. Tous les points essentiels sont abordés dans ce cours, de l'installation de base à l'administration avancée, tant système que réseau. Il n'est pas spécifique : son contenu est valable pour toutes les distributions.

Quel matériel pour Linux ?

1. L'architecture

Linux existe pour au moins trois architectures matérielles courantes :

- **x86** pour les ordinateurs dont les processeurs sont du type Intel (du 386 au Pentium 4) ou AMD (Athlon, Duron, Sempron) 32 bits. Cette version fonctionne aussi sur les machines à base de processeurs 64 bits.
- **x86_64** pour les ordinateurs dont les processeurs sont du type Intel (Pentium 4 à partir des séries 600, Xeon, Dual Core/Quad Core) ou AMD (Athlon 64, Sempron 64, Opteron) 64 bits. Cette version ne marche pas sur les processeurs 32 bits.
- **ppc** pour les ordinateurs dont les processeurs sont de type PowerPC c'est-à-dire les anciens ordinateurs de marque Apple. Cette version ne s'installera pas sur les dernières machines Apple basées sur un processeur de marque Intel.
- Certains pilotes matériels ou applications sont encore peu ou mal adaptés à la version 64 bits.

Configuration matérielle de base

Linux supporte théoriquement tous les types de processeurs depuis la version 386, et peut fonctionner avec seulement quelques Mo de mémoire. La distribution Polux Linux fonctionne sur un 386 avec 4 Mo de mémoire. La distribution Damn Small Linux fonctionne avec un 486, 16 Mo de mémoire et utilise 50 Mo d'espace disque. On trouve même des distributions sur une ou deux disquettes démarrant avec 2 Mo de mémoire.

Les pré requis suivants doivent être respectés :

- **Un processeur** (ou plus) de type Intel Pentium et supérieur ou un équivalent de marque AMD.
- **Au moins** 128 Mo de mémoire, mais 256 Mo ou plus apportent une réel confort d'utilisation. Pensez plutôt à disposer de 512 Mo voire 1 Go pour une utilisation optimale. Au prix de la mémoire ce n'est pas un luxe. Dans
- le cadre d'une installation minimale en mode texte, 64 Mo suffisent.
- 500 Mo d'espace disque pour une installation minimale (sans interface graphique et seulement les outils de base), mais 2,5 Go pour une installation standard, auquel il faut rajouter l'espace pour les données de l'utilisateur et la partition d'échange.
- Une carte graphique même ancienne compatible avec la norme Vesa, acceptant de préférence le 1024x768 en 65 356 couleurs pour l'environnement graphique, et sans aucune importance en mode texte.

Les performances globales restent acceptables sur un Pentium II 300 avec 256 Mo pour une utilisation bureautique ou Internet simples. Les performances s'écroulent lors du lancement simultané de plusieurs programmes. Sur un simple AMD Duron 800 avec 512 Mo, les performances sont excellentes pour la plupart des usages classiques.

Choisir une distribution

1. Debian

Le projet Debian a été fondé en 1993 par Ian Murdock à une époque où l'idée même de distribution Linux en était encore à ses balbutiements. Le nom Debian provient de Debra (la femme de Murdock) et Ian. Debian a longtemps été la seule distribution entièrement et uniquement composée de logiciels libres et Open Source ce qui lui vaut toujours le nom officiel de Debian GNU/Linux. Debian a aussi été supporté quelques temps officiellement par la FSF comme distribution Linux de référence. Les avantages de Debian sont nombreux :

- un nombre gigantesque de packages qui se chiffre en milliers,
- un logiciel d'installation appelé APT très pratique et performant,
- une distribution 100% open source,
- une stabilité à toute épreuve pour un environnement de production.

Ces avantages entraînent aussi quelques inconvénients :

- des packages souvent anciens,
- des mises à jour de la distribution irrégulières et trop espacées,
- des risques liés à la multiplication des paquets et des dépendances,
- une installation et une configuration compliquées.

Tous ces éléments font de Debian une distribution idéale pour les informaticiens, les ingénieurs et administrateurs système et réseau, les environnements de production en entreprise, les puristes du libre, les amateurs éclairés qui n'ont pas peur de mettre les mains dans le cambouis. Quant aux débutants ils passeront probablement leur chemin au début sauf à vouloir apprendre sur le tas.

2. Ubuntu

Le milliardaire sud-africain Mark Shuttleworth, principalement connu du monde entier pour avoir été l'un des premiers touristes de l'espace, mais aussi des informaticiens pour avoir fait fortune en revendant sa société Thawte spécialisée dans la sécurité à Verisign, est un vrai informaticien qui a contribué au projet Debian. Devant les quelques inconvénients de la distribution il crée la distribution Ubuntu Linux en 2005 avec un budget initial de 10 millions de dollars pour rémunérer les développeurs. Le mot Ubuntu est un mot du langage africain bantou signifiant « humanité aux autres » ou encore « je suis ce que je suis grâce à ce que nous sommes tous ». Cette définition reflète ce qu'est la distribution : un dérivé de Debian dont le but est de fournir des logiciels plus récents et très fortement axés sur la convivialité et l'ergonomie à l'aide du support du plus grand nombre :

- une distribution issue de Debian,
- une compatibilité avec les packages de Debian,
- un système d'installation très simple,
- une sortie tous les 6 à 8 mois,
- un environnement graphique agréable.

Cette distribution est idéale pour les étudiants, cependant la tentation est très forte de revenir au fonctionnement d'une distribution Debian, les deux étant compatibles.

3. Red Hat et Fedora

S'il y a bien une société commerciale dans le monde Linux qui a marqué et qui continue à marquer son époque, c'est bien la société Red Hat. Fondée en 1995 par Robert Young et Marc Ewing, elle édite la célèbre distribution éponyme dont la première version officielle date de 1994 (la société a été fondée après la sortie de la distribution). Le système de package RPM est apparu avec la version 2.0. Les distributions Red Hat ont très fortement marqué les esprits car elles sont restées la référence pendant presque dix ans. Chaque version était innovante tant dans l'intégration des logiciels que dans son installateur (appelé anaconda) et ses outils de configuration.

Cependant en 2003 la version 9.0 est la dernière destinée officiellement au grand public. Les versions suivantes ont été confiées au projet communautaire **Fedora** qui continue tous les six mois à sortir une nouvelle version. Red Hat se concentre maintenant sur le monde de l'entreprise avec des distributions commerciales appelées **RHEL** (*Red Hat Enterprise Linux*) :

- des versions professionnelles destinées aux entreprises,
- des solutions du poste de travail au plus gros serveur,
- des architectures matérielles nombreuses,
- un support commercial,
- des mises à jour assurées pendant sept ans,
- 100% libre.

Vous vous doutez bien que même si l'installation d'une version RHEL AS (*Advanced Server*) est possible sur un PC de bureau elle n'a pas forcément d'intérêt pour un poste de travail ou un débutant. Bien que libre (ses sources sont intégralement disponibles librement) son coût avec le support est très élevé. Cependant si l'installation ne vous fait pas peur la distribution **CentOS** (*Community Enterprise Operating System*) est une copie exacte et téléchargeable de RHEL dont toute trace des noms et visuels Red Hat a été supprimée.

Quant au projet Fedora, il suit un cycle de développement rapide et reste destiné au grand public. Son installation est simple. Cependant l'ensemble manque un peu de cohérence (par exemple l'outil de partitionnement des disques n'est accessible que durant l'installation) ce qui en fait une distribution idéale pour tous ceux qui, amateurs éclairés, souhaitent rentrer un peu plus dans le détail.

4. Mandriva (ex-Mandrake)

Mandriva Linux (ex-Mandrake) est une distribution dérivée et longtemps entièrement compatible avec la distribution Red Hat. Elle a été créée par Gaël Duval afin d'intégrer à la distribution l'environnement de bureau graphique KDE contrairement à Red Hat qui intégrait l'environnement GNOME. Pendant plusieurs années Mandrake a été la distribution phare en forte compétition avec Red Hat. Mandrake était en effet (et est toujours) plus conviviale. Son processus d'installation est un modèle du genre et son utilisation des plus simples. Renommée Mandriva suite au rachat de la société Connectiva, la distribution est pourtant en perte d'audience depuis quelques temps. Les raisons sont multiples mais fortement liées aux aléas de la société Mandriva. Souffrant d'une image trop grand public, les solutions professionnelles n'arrivent pas à s'imposer. Enfin la distribution grand public si elle reste toujours au top techniquement souffre parfois de quelques problèmes d'instabilité.

Mandriva continue cependant d'innover fortement, notamment dans le poste de travail nomade avec des distributions clé en main bootables depuis des clés USB, et c'est généralement plus par habitude et ouïe dire qu'elle est bien souvent automatiquement conseillée aux débutants.

5. openSUSE

Se prononçant *sousse*, **openSUSE** est une distribution d'origine allemande datant de 1992. Le nom de l'entreprise lui-même était un hommage au célèbre **Konrad Zuse** l'inventeur des ordinateurs modernes.

La distribution est originellement basée sur la distribution Slackware. En 1996 SuSE se rapproche d'une distribution française appelée **Jurix** créée par Florian La Roche qui est utilisée comme base à la place de Slackware. Cette même année le développement de l'outil YaST est démarré et la version 4.2, en fait totalement nouvelle, sort. Au même moment SuSE utilise le nouveau gestionnaire de packages de Red Hat appelé RPM.

Début 1997 SuSE tente l'aventure américaine en installant de nouveaux bureaux à Oakland. Entre 1997 et 2003 la distribution SuSE ne cesse d'être améliorée pour devenir une référence en matière de simplicité d'installation, d'administration et d'utilisation.

Si l'avenir de la distribution était garanti, la société Novell rachète tout d'abord la société Ximian spécialisée dans le développement Open Source d'outils pour Linux dont un bureau Gnome, une messagerie appelée Evolution et une suite de configuration appelée Red Carpet. Novell annonce le rachat de la société SuSE en janvier 2004. Le développement est désormais communautaire avec le projet **openSUSE**. Le monde entier s'il le souhaite peut contribuer à l'amélioration du produit. En réponse, Novell s'engage à fournir à la communauté tous les six à huit mois une version stable, libre et gratuite.

6. Les autres

Il est impossible de nommer toutes les distributions tant elles sont nombreuses. Outre les grandes distributions que vous venez de rencontrer quelques autres noms sont à retenir. La distribution **Slackware** est l'une des plus anciennes distributions. Elle était même livrée sur disquette. Durant les toutes premières années la Slackware était la distribution de référence pour apprendre à utiliser Linux. Elle est extrêmement dépouillée. Son installateur est réduit à la plus simple expression et la plupart de la configuration doit être effectuée à la main. Son système de package est inexistant (il s'agit de simples archives de fichiers compressés). C'est donc l'idéal pour les bidouilleurs et les fondus de Unix. Cependant, ce n'est pas l'idéal pour les débutants.

La distribution **Gentoo** est très particulière. Plutôt que de vous livrer tous les logiciels déjà prêts à être utilisés, son installateur va avec votre aide déterminer exactement la configuration de votre machine et notamment votre modèle de microprocesseur en fonction de quoi il compilera (il transformera le programme source sous forme de langage compréhensible en langage machine) chaque composant logiciel que vous aurez sélectionné avec toutes les optimisations prévues pour votre matériel. C'est ce qu'on appelle une distribution source. Le résultat peut être intéressant : les performances de vos programmes peuvent être améliorées, étant en moyenne de 10% à 20% plus rapide. Mais à quel prix ! L'installation n'est pas forcément aisée pour les débutants et surtout elle est très longue : plusieurs heures (voire dizaines d'heures) selon vos choix de logiciels et la puissance de votre machine.

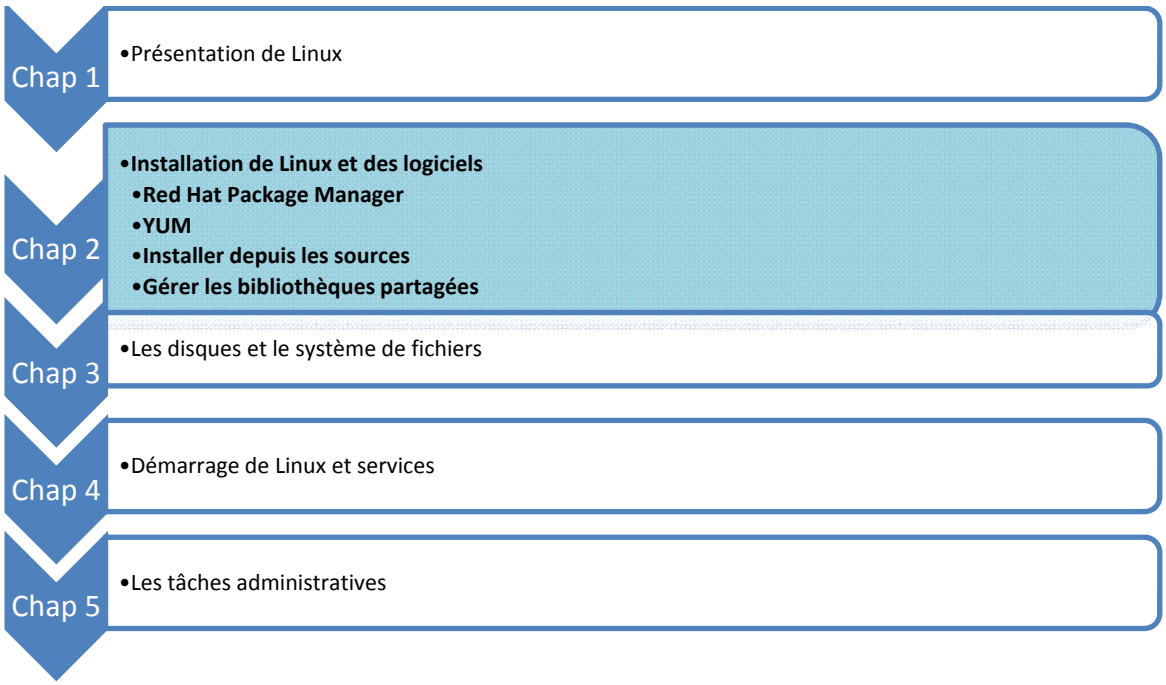
Une autre distribution surprenante est la **LFS** (*Linux From Scratch*). Ce n'est pas précisément une distribution mais plutôt un guide vous donnant une méthode pour construire votre propre configuration. Pas à pas, c'est à vous de choisir vos divers composants et la configuration de votre système. Ainsi vous êtes certain d'obtenir exactement la distribution que vous voulez, ni plus ni moins. Mais là encore les débutants, et même d'ailleurs les amateurs éclairés, passent leur chemin.

A côté de toutes ces distributions on trouve de nombreux dérivés. **Aurox Linux** dérive de Red Hat. **PCLinuxonline** dérive de Mandriva. **Kunbuntu** dérive de Ubuntu (ou plutôt est une distribution Ubuntu pleinement supportée mais intégrant l'environnement bureautique KDE) qui dérive de Debian. **CentOS** dérive de RHEL, et ainsi de suite. Encore à côté il y a les mini-distributions qui tiennent sur un mini cd ou une clé USB et c'est idéal pour dépanner un ordinateur.

7. Les LiveCD

Le LiveCD est une catégorie surprenante. Vous êtes certainement très nombreux à vouloir essayer Linux pour voir à quoi ça ressemble ou pour vérifier s'il fonctionne correctement avec votre matériel. Plutôt que de l'installer sur votre disque dur pensez d'abord à tester Linux sans l'installer. Le LiveCD sert principalement à ça : c'est une installation complète de Linux qui est fortement compressée et qui tient sur un seul cd ou dvd (dans ce cas on parle de liveDVD).

Pour utiliser un liveCD c'est très simple : insérez le CD ou le DVD dans votre lecteur et redémarrez votre ordinateur en ayant bien vérifié dans la configuration de votre machine (le setup du BIOS, voire le mode d'emploi de votre ordinateur) que votre lecteur CD ou DVD est le premier à démarrer. Après quelques secondes (ou minutes parfois) de chargement, voici que le bureau apparaît et tous les programmes les plus connus sont accessibles alors que strictement rien n'est installé sur votre disque dur. Le LiveCD le plus connu actuellement est **Knoppix**. Il est basé sur une distribution Debian, et qui plus est, s'il vous plaît un installateur est prévu pour le copier sur votre disque dur. Chaque nouvelle version de SUSE Linux arrive avec un LiveCD pour tester les dernières nouveautés sans l'installer.


Chap 1
•Présentation de Linux
Chap 2
•Installation de Linux et des logiciels
•Red Hat Package Manager
•YUM
•Installer depuis les sources
•Gérer les bibliothèques partagées
Chap 3
•Les disques et le système de fichiers
Chap 4
•Démarrage de Linux et services
Chap 5
•Les tâches administratives

Red Hat Package Manager

1. Notion de package

Contrairement à d'autres systèmes d'exploitation, il n'est pas courant sur Linux et Unix en général de disposer de logiciels fournis avec un programme l'installation interactif (pas de install.exe). Certains éditeurs proposent des scripts d'installation et bien souvent ceux-ci se contentent de décompresser et de désarchiver quelques fichiers.

Avec Linux, il est très classique de disposer des divers produits, outils, mises à jour, etc. sous forme de paquetages (packages). Un package est un fichier (parfois gros) qui contient le produit à installer et des règles. Ces règles peuvent être multiples :

- Gestion des dépendances : le produit ne pourra être installé que si les produits qu'il utilise lui-même sont déjà présents.
- Pré-installation : des actions sont à prévoir avant de pouvoir installer le produit (changer des droits, créer des répertoires, etc.).
- Post-installation : des actions sont à prévoir après l'installation du produit (paramétrage d'un fichier de configuration, compilation annexe, etc.).

Sur Red Hat, Fedora, SuSE, Mandriva et quelques autres distributions le format de package par défaut est le **RPM** (*Red Hat Package Manager*). Sous Debian, Knoppix, Kaella, Ubuntu, c'est le format **DPKG** (*Debian Package*). Outre le format, ce sont surtout les outils qui les différencient.

Le fait de disposer des informations de dépendances permet d'obtenir des outils performants qui peuvent seuls les résoudre en cascade. En installant un package, l'outil pourra installer toutes les dépendances nécessaires. On peut parfois spécifier plusieurs emplacements (repositories) pour ces packages, soit locaux (disque dur, CD-Rom, DVD, etc.) soit distants (http, ftp, etc.).

Il faut toujours utiliser un package prévu pour sa distribution quand il existe. Si ce n'est pas le cas, il est parfois possible d'utiliser un package d'un produit concurrent ou de recompiler le produit soi-même.

Les mises à jour d'un système Linux utilisant un système de packaging sont très simplifiées. Pour passer d'une version d'un produit à un autre, il suffit de récupérer le package du produit en version supérieure et de l'installer. Toutes les mises à jour sont sous cette forme. Depuis peu, il existe un format de **delta-rpm** qui ne fournit dans le package que les différences d'une version à une autre. Mais il est toujours possible d'utiliser un package complet.

2. Le gestionnaire RPM

RPM est un gestionnaire de packages inventé par Red Hat puis utilisé massivement par de nombreuses autres distributions. Il simplifie fortement la distribution, l'installation, la mise à jour et la suppression des logiciels. Il se base sur des commandes (ex : **rpm**), une base de données locale et des packages au format rpm (extension rpm).

La base de données est située dans **/var/lib/rpm**. Toutes les informations concernant les logiciels installés, leurs versions, leurs fichiers et droits, et leurs dépendances y sont précisées. Sauf gros problème, il ne faut JAMAIS modifier cette base à la main. Il faut utiliser les outils RPM.

Chaque logiciel est fourni sous forme de package au format RPM. Le rpm répond à une nomenclature précise.

```
nom-version-edition.architecture.rpm
```

par exemple :

```
php-4.1.2-2.1.8.i586.rpm
```

L'édition est un identifiant de version du package RPM propre à l'éditeur. Ici c'est la version 2.1.8 du package PHP version 4.1.2. L'architecture est i586 (Intel Pentium). On peut aussi trouver i386, i686, x86_64 (64 bits), ppc64, s390x ou noarch. Un package noarch ne contient pas de programmes ou bibliothèques binaires mais du code indépendant comme des scripts, de la documentation, des images, du son, de la vidéo, etc.

3. Installation, mise à jour et suppression

Vous installez un package rpm avec le paramètre `-i`.

```
rpm -i php-4.1.2-2.1.8.i586.rpm
```

Comme il est possible d'utiliser des caractères de substitution (**`rpm -i *.rpm`**), vous pouvez afficher le nom du package en cours d'installation avec le paramètre `-v`. Le paramètre `-h` affiche des caractères `#` pour indiquer la progression de l'installation. L'installation ne fonctionnera pas si les dépendances ne sont pas résolues.

La mise à jour d'un produit vers une version supérieure depuis un package se fait avec le paramètre `-U`. Dans ce cas tous les fichiers sont mis à jour par ceux de la nouvelle version : les anciens sont supprimés et remplacés par les nouveaux. Les anciens fichiers de configuration sont sauvés avec l'extension **`.rpmsave`**. Si le package n'était pas installé, la mise à jour joue le rôle d'installation. Attention avec ce paramètre : il installe le package même si une version précédente n'était pas installée.

```
rpm -Uvh php-4.1.3-1.i586.rpm
```

La mise à jour est aussi possible avec `-F`. Mais si le package n'était pas installé, il ne le sera pas non plus lors de la mise à jour contrairement à `-U`. Ainsi, si vous disposez de tous les packages de mise à jour du système et que vous ne souhaitez mettre à jour que ceux qui sont réellement installés, alors vous pouvez taper :

```
rpm -Fvh *.rpm
```

La suppression s'effectue avec le paramètre `-e`. Attention cependant, c'est le nom du package installé qui doit être passé en paramètre et pas le nom du fichier de package.

```
rpm -e php
```

Plusieurs options supplémentaires sont possibles :

- **`--force`** : en cas de conflit avec un autre package (le cas le plus courant est celui où deux packages proposent le même fichier au même endroit), cette option force tout de même l'installation.
- **`--nodeps`** : si le package refuse de s'installer à cause d'un problème de dépendances, cette option forcera l'installation. Il arrive parfois que cette erreur se produise quand la dépendance en question a été installée autrement que depuis un package rpm (ex : compilation, binaire copié à la main).

4. Cas du noyau

L'installation ou la mise à jour d'un noyau est un cas particulier. En effet, la mise à jour supprime l'ancienne version.
Le noyau est un composant très critique du système. S'il devait être avéré que le système ne fonctionne plus (ou mal) avec le noyau mis à jour, il faudrait donc réinstaller un ancien noyau depuis le support d'installation. Aussi la procédure est la suivante :

- Installation du nouveau noyau avec le paramètre `-i`, il sera rajouté au système.
- Redémarrage et test de vos logiciels et périphériques avec le nouveau noyau.
- S'il fonctionne correctement, suppression éventuelle de l'ancien noyau avec `-e`.
- Édition de **`/boot/grub/grub.conf`** et modification de la ligne **`Default`** pour démarrer par défaut sur le nouveau noyau.

5. Requêtes RPM

La base de données RPM peut être interrogée facilement avec le paramètre `-q` suivi de plusieurs options.
`-a` : liste de tous les packages installés.

`-i` : informations générales (le résumé) du package.

`-l` : liste des fichiers installés.

`-f nom` : trouve le package qui contient le fichier donné.

-p nom : la recherche s'effectue dans le fichier de package donné.

--requires : dépendances du package.

--provides : ce que fournit le package.

--scripts : scripts exécutés à l'installation et la suppression.

--changelog : l'historique du package.

6. Vérification des packages

Il est possible qu'après l'installation d'un package, un ou plusieurs des fichiers installés aient été altérés (changement de droit, de propriétaire, édition, suppression, etc.). Comme la base RPM contient toutes les informations nécessaires, on peut demander une vérification avec le paramètre `-V`.

- **S** : la taille du fichier a été modifiée.
- **5** : la somme MD5 ne correspond plus.
- **T** : la date de modification n'est plus la même.
- **U** : le propriétaire a été modifié.
- **G** : le groupe a été modifié.
- **L** : le lien symbolique a été modifié.
- **M** : les permissions ou le type du fichier ont été modifiés.
- **D** : le périphérique a été modifié (major/minor). Le **c** indique qu'il s'agit d'un fichier de configuration.

Les fichiers de packages RPM sont très souvent signés par l'éditeur de la distribution de manière à en garantir l'intégrité. On peut vérifier l'intégrité d'un package avec une clé publique GPG mais il faut par avance avoir déjà chargé cette clé publique sur le système.

7. Les dépendances

Si vous utilisez les outils graphiques fournis par votre distribution, ceux-ci tenteront de résoudre les dépendances à votre place. La commande **rpm** seule ne le fait pas par défaut. Des outils complémentaires « frontend » comme **yast**, **apt** ou **yum** le font à sa place. La distribution Red Hat fournissait jusqu'aux versions 4 (RHEL) un outil appelé **rpmdb-redhat** pour installer automatiquement les dépendances via rpm. Cela implique notamment le fait que tous les packages de la distribution doivent se trouver au même endroit (dans le même répertoire) et le système ne fonctionne qu'avec les packages officiels de Red Hat. On emploie le paramètre `--aid`.

```
$ rpm -ivh --aid libjpeg-6.2.0-738.i586.rpm
```

8. Mises à jour automatisées

Chaque distribution fournit maintenant un outil de mise à jour interactif ou automatisé. La openSUSE propose **YOU** (*Yast Online Update*), la Red Hat propose **up2date**. La version RHEL de Red Hat étant payante, l'accès aux mises à jour dépend d'un numéro de licence et d'une inscription à **RHN** (*Red Hat Network*). Les versions dérivées comme CentOS ne peuvent pas se mettre à jour via RHN mais proposent leur propre site distant de mise à jour.

YUM

YUM est un logiciel de gestion de packages. Il récupère les packages au sein de dépôts et gère les dépendances à votre place. YUM signifie *Yellow dog Updater Modified*. Il est principalement utilisé sur les distributions Redhat et Fedora.

Le fichier de configuration est **/etc/yum.conf**.

1. Configuration des dépôts

Les dépôts sont placés soit dans le fichier de configuration principal, soit dans le répertoire **/etc/yum.repos.d**. Le format est le suivant :

```
[rhel5]
name=ES5 baseurl=ftp:/ftp.server.com/redhat/x86/ES5u3/Server/ gpgcheck=1
enabled=1
gpgkey=ftp://ftp.server.com/x86/ES5u3/RPM-GPG-KEY-redhat-release
```

Le dépôt se nomme (nom court) `rhel5`.

- **name** : le nom long du dépôt, détaillé.
- **baseurl** : l'URL du dépôt.
- **gpgcheck** : demande une vérification de la signature GPG du dépôt.
- **enabled** : si absent ou à 1, le dépôt est actif.
- **gpgkey** : chemin de la clé publique GPG.

Les URL des dépôts peuvent être locales (file://) ou distantes (http:// ou ftp://). Elles doivent pointer sur un répertoire contenant les informations de dépôts qui sont dans le dossier `repodata`.

En tenant compte des valeurs par défaut, un simple dépôt peut être déclaré ainsi :

```
[updates-rhel5]
name=UPDATES-RHEL5
baseurl=ftp://ftp.server.com/RPMS.rhel5_updates_x86
```

Attention cependant car la configuration de YUM peut modifier les valeurs par défaut. La section `[main]` de **/etc/yum.conf** peut ainsi contenir la ligne :

```
gpgcheck=1
```

Dans ce cas, vous devrez modifier la valeur `gpgcheck` à 0 dans les dépôts ne nécessitant pas de signature.

2. Utilisation des dépôts

a. Rafraîchir le cache

À chaque commande, YUM tente de rafraîchir ses données si le délai d'expiration a été dépassé. Ce délai peut être réduit ou étendu en modifiant la ligne `metadata_expire` du fichier de configuration.

```
metadata_expire=1h
```

b. Lister les packages

Le paramètre `list` permet de lister les packages. Tous sont listés par défaut. Vous pouvez préciser une liste de packages, ou fournir des caractères jokers. Plusieurs options sont disponibles :

- **all** : c'est le cas par défaut : les packages installés sont listés en premier, puis les packages disponibles pour installation.
- **available** : les packages disponibles pour installation.
- **updates** : les packages pouvant être mis à jour.
- **installed** : les packages mis à jour.
- **obsoletes** : les packages du systèmes rendus obsolètes par des versions supérieures disponibles.
- **recent** : les derniers packages ajoutés dans les dépôts.

c. Installer des packages

Passez le paramètre install, suivi des noms des packages à installer. Voici un exemple d'installation du package mc :

```
[root@server /etc]# yum install mc
Setting up Install Process
Parsing package install arguments
Resolving Dependencies
--> Running transaction check
---> Package mc.i386 1:4.6.1a-35.el5 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package        Arch    Version           Repository   Size
================================================================================
Installing:
 mc             i386    1:4.6.1a-35.el5   rhel5        2.1 M

Transaction Summary
================================================================================
Install        1 Package(s)
Update         0 Package(s)
Remove         0 Package(s)

Total download size: 2.1 M Is this ok [y/N]: y
Downloading Packages:
mc-4.6.1a-35.el5.i386.rpm                          2.1 MB     00:00
Running rpm_check_debug Running Transaction Test
Finished Transaction Test Transaction Test Succeeded Running Transaction
Installing         : mc                            [1/1]

Installed: mc.i386 1:4.6.1a-35.el5
Complete!
```

d. Mises à jour

Vérifiez la présence de mises à jour avec le paramètre check-update :

```
[root@slyserver /etc]# yum check-update
```

Si rien n'est retourné, c'est qu'aucune mise à jour n'est disponible. Vous avez deux possibilités pour installer les mises à jour :

- **update** : mise à jour d'un package ou de tous si aucun package n'est précisé.
- **upgrade** : mise à niveau complète de la distribution : les packages vus comme obsolètes sont remplacés par ceux de la dernière version disponible.

Dans certains cas, pour le noyau par exemple, vous devrez éviter, lors d'une mise à jour, d'installer automatiquement certains packages. Dans ce cas, utilisez le paramètre --exclude :

```
[root@slyserver etc]# yum list -exclude=kernel\* update
```

Pour rendre permanente cette exclusion, mettez-la en dur dans le fichier de configuration en ajoutant une ligne comme ceci :

```
exclude=php* kernel*
```

e. Rechercher un package

Utilisez le paramètre search, suivi du ou des packages à rechercher dans les dépôts. Les caractères jockers sont autorisés.

```
root@slyserver etc]# yum search tomcat
============== Matched: tomcat ==========================
 jakarta-commons-collections-tomcat5.i386 : Jakarta Commons Collection dependency for Tomcat5
struts-webapps-tomcat5.i386 : Exemples d'applications Web struts pour tomcat5
tomcat5.i386 : Moteur Servlet/JSP Apache, RI pour Servlet 2.4/JSP
```

f. Supprimer un package

Pour supprimer un package, utilisez le paramètre remove :

```
[root@slyserver~]# yum remove mc Setting up Remove Process Resolving Dependencies
--> Running transaction check
---> Package mc.i386 1:4.6.1a-35.el5 set to be erased
--> Finished Dependency Resolution
```

Installer depuis les sources

1. Obtenir les sources

Il n'est parfois pas possible d'obtenir un logiciel ou une bibliothèque depuis un package pour sa distribution. Dans ce cas, il reste la solution de compiler et d'installer soi-même le produit depuis les sources.

Cela est possible pour une majorité de produits sous Linux, grâce aux avantages des logiciels libres et de la licence GPL. Tout logiciel libre est fourni avec ses sources. Il est donc possible de reconstruire soi-même le logiciel en le recompilant.

Une archive source est souvent récupérée sur divers sites Internet comme par exemple SourceForge. C'est une archive bien souvent compressée au format tgz (archive tar compressée avec gzip) ou tar.bz2 (archive tar compressée au format bzip2). Elle contient :

- le code source sous forme de fichiers **.c**, **.h**, **.cpp**, etc., selon le langage ;
- parfois un fichier **Makefile** permettant d'automatiser la compilation du produit ;
- souvent un fichier **.configure** permettant de générer le fichier Makefile en fonction de votre installation et de diverses options.

2. Pré-requis et dépendances

Pour compiler votre produit vous devez respecter quelques pré-requis :

- présence de l'outil make ;
- présence du ou des compilateurs nécessaires, notamment gcc ;
- présence des dépendances : bibliothèques, interpréteurs, etc.

Ce dernier point est très important. S'il manque une dépendance vous risquez divers problèmes :

- vous n'arriverez pas préparer les sources pour la compilation ;
- la compilation générera des erreurs ;
- le produit sera compilé mais avec des possibilités moindres ;
- le binaire résultant ne se lancera pas.

La commande **./configure** vous fournira les dépendances manquantes et leur version si c'est possible. Dans ce cas vous pouvez soit les installer depuis les packages de votre distribution, soit les installer depuis les sources.
Dans tous les cas, il n'y a pas besoin d'être root pour compiler votre logiciel. Cependant, selon la destination vous devrez passer root pour finaliser l'installation.

3. Exemple d'installation

Vous allez compiler et installer le produit PDFedit qui permet d'éditer et de créer des fichiers PDF.

- Téléchargez-le depuis le lien suivant. La version testée est la version 0.4.1 :

http://sourceforge.net/project/showfiles.php?group_id=177354

```
$ ls -l pdfedit-0.4.1.tar.bz2
-rw-r--r-- 1 seb users 2958137 mai 20 14:26
```

- Décompressez le fichier :

```
$ tar xvjf pdfedit-0.4.1.tar.bz2
```

- Déplacez-vous dans le dossier **pdfedit-0.4.1** créé par la décompression :

```
$ cd pdfedit-0.4.1/
```

- Remarquez la présence du fichier configure qui est exécutable.

```
$ ./configure --help
```

Une option importante de configure est **--prefix**. Elle définit l'emplacement de l'installation une fois le produit compilé. Par défaut le logiciel s'installe dans **/usr/local/**.

- Exécutez **./configure** seul. Il vous informera des dépendances manquantes le cas échéant.

```
$ ./configure
checking for g++... g++
checking for C++ compiler default output file name... a.out checking whether the C++ compiler
works... yes
...
creating src/utils/aconf.h config.status: creating src/xpdf/aconf.h
```

- Lancez la compilation avec la commande **make**. Il se peut que des avertissements apparaissent (lignes warning). Cela ne signifie pas forcément que le programme ne compilera pas ou ne marchera pas par la suite. De toute façon si la compilation produit des erreurs, elle s'arrêtera toute seule avec un message d'erreur du compilateur pouvant parfois (mais pas toujours) vous mettre sur la voie d'une solution.

```
$ make
```

- La compilation s'étant terminée sans erreur, finissez en installant le produit avec **make install**. Attention, le produit va s'installer dans /usr/local/ ce qui nécessite les droits de l'utilisateur root.

```
$ su -c "make install"
password  :
...
```

- Lancez le produit :

```
$ pdfedit
```

4. Désinstallation

La plupart des **Makefile**, en tout cas ceux générés par configure, permettent la désinstallation. Elle s'effectue par la commande **make uninstall**.

```
$ su -c "make uninstall"
password  :
...
```

Gérer les bibliothèques partagées

1. Principe

Une bibliothèque partagée est un fichier particulier qui contient une liste de fonctions, ou API, accessible à tout programme en ayant besoin sans avoir à les réécrire. À l'opposé de la bibliothèque statique, le programme accède dynamiquement aux fonctions qui sont placées dans un fichier à part. N programmes différents peuvent accéder aux fonctions proposées par la bibliothèque. Les bibliothèques regroupent des fonctions propres à un domaine ou un ensemble de domaines cohérents : traitement d'images, du son, de l'accès à une base de données, etc.

Un ensemble de fonctions proposées par une ou plusieurs bibliothèques partagées forme une **API**, *Application Programming Interface*, et sont parfois regroupées au sein d'un framework offrant une solution complète pour un domaine donné.

Un lien est établi entre le programme et une bibliothèque partagée lors de l'étape de l'édition des liens par l'éditeur de liens **ld**, lui-même appelé par le compilateur **gcc** avec l'option **-l<lib>**.

Une autre possibilité pour un programme est d'utiliser la fonction C **dlopen** qui ouvre une bibliothèque dynamique comme un fichier et qui accède aux fonctions qui y sont contenues avec des pointeurs de fonctions.

Si un programme dépend d'une bibliothèque partagée et que celle-ci est absente, le programme ne pourra plus fonctionner.

Sous Linux (et Unix en général) les bibliothèques partagées sont appelées des **Shared Objects** (so) dans le sens où il s'agit de fichiers objets sans bloc d'instruction **main**. Ils portent le suffixe **.so**.

Une bibliothèque peut disposer de plusieurs versions, pouvant être ou non compatibles, et la version peut être précisée lors de l'édition des liens, avec une version par défaut possible.

2. Lieu de stockage

Les bibliothèques partagées sont par convention placées dans des répertoires appelés lib :

- **/lib** : bibliothèques systèmes de base, vitales ;
- **/usr/lib** : bibliothèques utilisateur de base, non nécessaires au boot ;
- **/usr/local/lib** : bibliothèques locales aux produits pour la machine ;
- **/usr/X11R6/lib** : bibliothèques de l'environnement X Window ;
- **/opt/kde3/lib** : bibliothèques de KDE ...

```
$ ls -l /lib
total 6024
...
-rwxr-xr-x  1 root root 114636 oct 23 2007 ld-2.6.1.so
lrwxrwxrwx  1 root root 11 oct  5 2007 ld-linux.so.2 -> ld-2.6.1.so
lrwxrwxrwx  1 root root 13 oct  5 2007 ld-lsb.so.2 -> ld-linux.so.2
lrwxrwxrwx  1 root root 13 oct  5 2007 ld-lsb.so.3 -> ld-linux.so.2
```

La commande **ldd** permet de déterminer quelles sont les bibliothèques liées à un programme, et aussi si celles-ci sont présentes ou non.

```
$ ldd pdfedit
linux-gate.so.1 =>  (0xffffe000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0xb7ea5000)
```

3. Configurer le cache de l'éditeur de liens

L'édition des liens avec une bibliothèque partagée est dynamique et se fait au moment de l'exécution du programme par le système à l'aide de la bibliothèque ld.so. Le binaire fournit le nom des bibliothèques à lier à l'exécution, mais pas le chemin. Les fonctions de **ld.so** déterminent en fonction de son nom la bibliothèque à utiliser parmi les chemins qu'elles connaissent.
Tout programme est lié à la bibliothèque ld.so ou plutôt **ld-linux.so** (ld-linux.so.2).
Le chargeur de liens ld.so recherche les bibliothèques dans plusieurs endroits dont, et dans cet ordre :

- les chemins précisés dans la variable d'environnement **LD_LIBRARY_PATH**. Les chemins sont séparés, comme pour PATH, par des ":" ;
- le contenu du fichier **/etc/ld.so.cache** qui contient une liste compilée (format binaire) des bibliothèques trouvées dans les chemins prédéfinis ;
- les répertoires **/lib** et **/usr/lib**.

La recherche dans **/lib** et **/usr/lib** est implicite. De même, le fait de remplir la variable **LD_LIBRARY_PATH** n'empêche en rien la recherche des bibliothèques aux autres endroits si elle n'est pas dans un des chemins de la liste.

Pour éviter la mise en place d'une variable dont le contenu peut être difficile à manipuler, ld.so propose un cache que vous pouvez modifier vous-même. Le cache est construit depuis le contenu du fichier **/etc/ld.so.conf** et de la commande **ldconfig**.

Ce fichier contient la liste des répertoires contenant les bibliothèques partagées :

```
# cat /etc/ld.so.conf
/usr/X11R6/lib/Xaw3d
/usr/X11R6/lib
/usr/lib/Xaw3d
/usr/i386-suse-linux/lib
/usr/local/lib
/opt/kde3/lib
include /etc/ld.so.conf.d/*.conf
```

Plutôt que de modifier ce fichier, un package ou vous-même pouvez décider de rajouter un fichier dans **/etc/ld.so.conf.d** contenant le ou les chemins de vos nouvelles bibliothèques.

Il ne suffit pas de rajouter le chemin : vous devez régénérer le cache avec la commande **ldconfig**.

```
# ldconfig
```

La commande **ldconfig** :

- met à jour le cache pour les chemins définis dans /etc/ld.so.conf et associés, ainsi que pour /usr/lib et /lib ;
- met à jour les liens symboliques sur les bibliothèques ;
- permet aussi de lister les bibliothèques connues dans le cache. Les options suivantes sont acceptées :

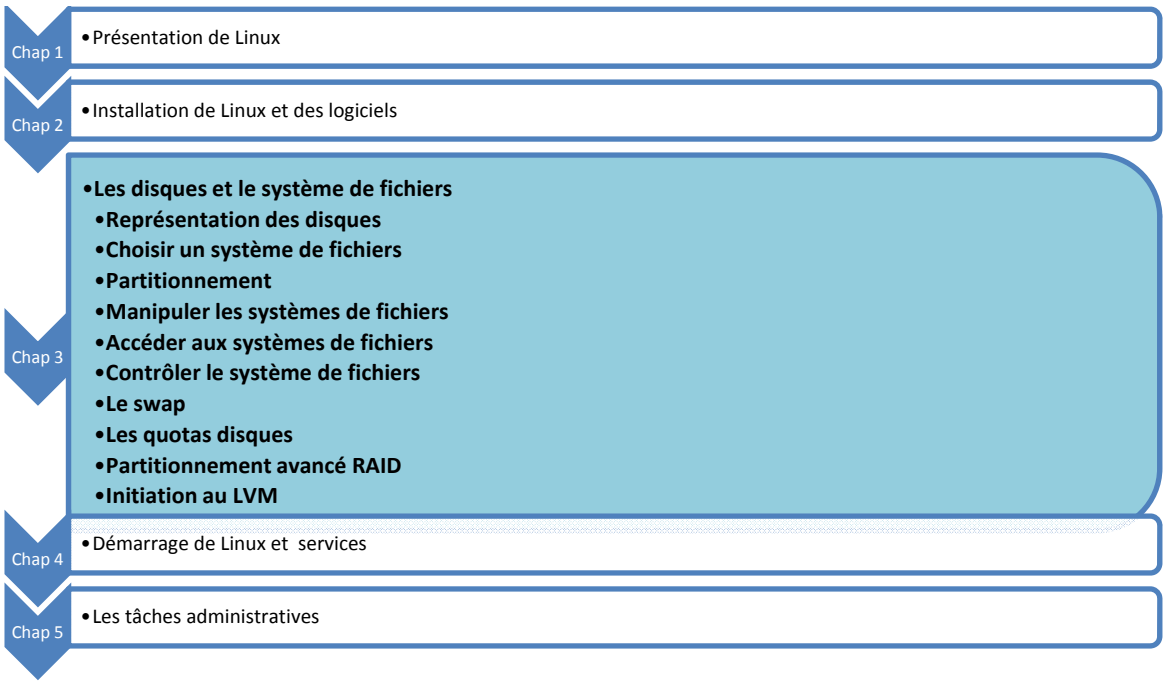
Option	Rôle
-v	Mode bavard : indique ce que ldconfig effectue
-N	Ne reconstruit pas le cache
-X	Ne met pas à jour les liens
-p	Liste le contenu du cache

Pour lister les bibliothèques connues de l'éditeur de liens :

```
# ldconfig -p
```

Enfin pour mettre à jour et voir le résultat :

```
# ldconfig -v
```


Chap 1
•Présentation de Linux
Chap 2
•Installation de Linux et des logiciels
Chap 3
•Les disques et le système de fichiers
•Représentation des disques
•Choisir un système de fichiers
•Partitionnement
•Manipuler les systèmes de fichiers
•Accéder aux systèmes de fichiers
•Contrôler le système de fichiers
•Le swap
•Les quotas disques
•Partitionnement avancé RAID
•Initiation au LVM
Chap 4
•Démarrage de Linux et services
Chap 5
•Les tâches administratives

Représentation des disques

1. Nomenclature

Suivant le type de contrôleur et d'interface sur lesquels les disques sont connectés, Linux donne des noms différents aux fichiers spéciaux des périphériques disques.

Chaque disque est représenté par un fichier spécial de type bloc. Chaque partition aussi.

a. IDE

Les disques reliés à des contrôleurs IDE (appelés aussi PATA, Parallel Ata, ou ATAPI) se nomment hdX :

- hda : IDE0, Master
- hdb : IDE0, Slave
- hdc : IDE1, Master
- hdd : IDE1, Slave
- etc.

Contrairement aux idées reçues, il n'y a pas de limites au nombre de contrôleurs IDE, sauf le nombre de ports d'extension de la machine (slots PCI). De nombreuses cartes additionnelles existent, de nombreuses cartes mères proposent jusqu'à quatre, six, huit connecteurs. Dans ce cas, les fichiers se nomment hde, hdf, hdg, etc.

Les lecteurs CD-Rom, DVD et graveurs sont vus comme des disques IDE et respectent cette nomenclature.

Les derniers noyaux Linux utilisent par défaut une API appelée libata pour accéder à l'ensemble des disques IDE, SCSI, USB, Firewire, etc. Si c'est votre cas (regardez les notes de version de la distribution), la nomenclature reprend celle des disques SCSI, abordée au point suivant.

b. SCSI, SATA, USB, FIREWIRE, etc.

Les disques reliés à des contrôleurs SCSI, SCA, SAS, FiberChannel, USB, Firewire (et probablement d'autres interfaces exotiques comme les lecteurs ZIP sur port parallèle) se nomment sdX. L'énumération des disques reprend l'ordre de détection des cartes SCSI et des adaptateurs (hosts) associés, puis l'ajout et la suppression manuelle via hotplug des autres.

- sda : premier disque SCSI
- sdb : deuxième disque SCSI
- sdc : troisième disque SCSI
- etc.

La norme SCSI fait une différence entre les divers supports. Aussi les lecteurs CD-Rom, DVD, HD-DVD, BlueRay et les graveurs associés n'ont pas le même nom. Les lecteurs et graveurs sont en srX (sr0, sr1, etc.). Vous pouvez aussi trouver scd0, scd1, etc. mais ce sont généralement des liens symboliques vers sr0, sr1, etc.

La commande **lsscsi** permet d'énumérer les périphériques SCSI.

Choisir un système de fichiers

1. Principe

a. Définition

L'action de « formater » un disque, une clé ou tout support de données consiste uniquement à créer sur un support de mémoire secondaire l'organisation logique permettant d'y placer des données. Le mot « formatage » n'est quasiment jamais utilisé sous Linux, sauf pour expliquer le principe aux personnes provenant d'autres horizons. On parle de système de fichiers qui est à la fois l'organisation logique des supports au niveau le plus bas comme au niveau de l'utilisateur.

Les informations ne sont pas écrites n'importe comment sur les disques. Une organisation est nécessaire pour y placer tant les informations sur les fichiers qui y sont stockés que les données. C'est le système de fichiers (et les pilotes associés) qui définit cette organisation. Si les principes de base sont souvent les mêmes entre les divers systèmes présents sous Linux, les implémentations et les organisations logiques des données sur le disque varient fortement. Aussi il n'existe pas un type de système de fichiers, mais plusieurs, au choix de l'utilisateur, administrateur ou ingénieur.

Le principe de base est d'associer un nom de fichier à son contenu et d'y permettre l'accès : création, modification, suppression, déplacement, ouverture, lecture, écriture, fermeture. Suivant ce principe, le système de fichiers doit gérer ce qui en découle : mécanismes de protection des accès (les permissions, les propriétaires), les accès concurrents, etc.

b. Représentation

Outre l'organisation et le stockage des informations et des données sur les fichiers, le système de fichiers doit fournir à l'utilisateur une vision structurée de ses données, permettant de les distinguer, de les retrouver, de les traiter et de les manipuler, par exemple sous forme de fichiers au sein d'une arborescence de répertoires avec les commandes associées. De même, chaque système de fichiers doit fournir le nécessaire pour que les programmes puissent y accéder.

Un système de fichiers Unix est organisé sous forme d'un arbre de répertoires et de sous-répertoires à partir d'une racine commune. C'est une arborescence. Chaque répertoire fait partie d'une organisation et propose lui-même une organisation : le système de fichiers dispose d'une hiérarchie ordonnée. L'arborescence elle-même peut être répartie sur plusieurs supports et systèmes de fichiers.

c. Les métadonnées

Un fichier est décrit par des propriétés appelées les métadonnées. Sous Linux, il s'agit de l'inode. Le contenu (les données) est placé dans d'autres blocs du support de stockage. Le contenu des métadonnées diffère d'un système de fichiers à un autre. Cependant on y retrouve sous Linux :

- les droits ;
- les dernières dates d'accès et de modification ;
- le propriétaire et le groupe ;
- la taille ;
- le nombre de blocs utilisés ;
- le type de fichiers ;
- le compteur de liens ;
- un arbre d'adresses de blocs de données.

d. Les noms des fichiers

Les noms peuvent avoir une longueur de 255 caractères. L'extension n'a pas d'existence en tant que composante
du système de fichiers contrairement à ce qui se passe sous Windows. Le type du fichier est déterminé par son contenu, notamment les premiers octets permettant de déterminer le type MIME. La commande **file** procède ainsi. L'extension n'est qu'une simple composante du nom du fichier, et incluse dans les 255 caractères. Elle est surtout utile pour que l'utilisateur différencie rapidement les fichiers entre eux.

Les noms des fichiers Unix ne sont pas placés dans les méta-données mais dans une table de catalogue. C'est pour ça qu'il est possible de donner plusieurs noms à un même fichier.

e. Le journal

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 (c'est un anglicisme). 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) et donc même si celui-ci est incomplet, la cohérence des données est assurée soit par complétion du journal quand c'est possible, soit par retour en arrière. La réparation est donc bien plus fiable et rapide.

2. Les filesystems sous Linux

a. ext2

Le « second extended filesystem » ext2 est considéré comme le système de fichiers historique de Linux, bien que celui-ci utilisait au tout début le MinixFS. La première mouture appelée ext (extended filesystem) bien que corrigeant les défauts de minix avait quelques limites qui n'en faisaient pas un véritable système de fichiers Unix. Ext2 est donc le premier système de fichiers développé spécifiquement pour Linux, d'un niveau de production et aux normes Unix (on parle de niveau de production pour indiquer un système quelconque répondant aux critères de mise en production (utilisation réelle) en entreprise). Prévu dès le début pour supporter les rajouts de fonctionnalités, il continue depuis 1993 à être utilisé et amélioré. Ext2 n'est pas journalisé.

Les fichiers peuvent avoir jusqu'à une taille de 2To (2048 Go), tandis qu'une partition peut atteindre 32 To, voire 128 To , selon la taille des blocs et l'architecture.

b. ext3

Le « third extended filesystem » ext3 est le successeur de ext2 depuis 1999. Il est journalisé. Surtout, il est entièrement compatible avec ext2. Le journal est une extension de ext2. Il est possible d'utiliser un système de fichiers ext3 comme étant ext2, avec les mêmes commandes, les mêmes manipulations. Il est possible de transformer en quelques secondes un système ext2 en ext3, et vice versa. C'est l'un des systèmes de fichiers de choix pour Linux, et le plus utilisé pour sa souplesse.

Comme pour ext2, la taille maximale des fichiers est de 2 To, et celle d'une partition de 32 To, suivant les mêmes restrictions.

c. reiserfs

reiserfs a été le premier système de fichiers intégré à Linux, avant même ext3. Sa force réside, outre dans son journal, dans l'organisation indexée des entrées des répertoires (les tables catalogues contenant les associations inodes/fichiers) et la manipulation des fichiers de petite taille. Ses performances sont exceptionnelles en présence de milliers de fichiers, de faible à moyen volume. Il est redimensionnable à chaud. Il devient plus lent sur des gros fichiers.

Les fichiers peuvent atteindre 8 To, et les partitions 16 To. Les noms de fichiers peuvent avoir 4032 caractères mais sont limités par Linux à 255 caractères (plus précisément par le **VFS**, *Virtual Filesystem Switch*).

reiserfs est moins utilisé malgré ses fortes qualités pour diverses raisons dont la principale est l'impossibilité de convertir un système de fichiers ext2/ext3 en reiserfs et vice versa, et ce à cause de la forte base de machines installées en ext2/ext3.

d. xfs

xfs est le plus ancien des systèmes de fichiers journalisés sous Unix, datant de 1993. Créé par Silicon Graphics (sgi) il a été porté sous Linux en 2000. Outre ses capacités de stockages encore inimaginables aujourd'hui, il a un système de journalisation très performant et des mécanismes avancés comme la défragmentation en ligne (à chaud et au fur et à mesure des écritures), la capacité d'effectuer des snapshots (figer l'état d'un filesystem à un instant t pour le restaurer plus tard), le dimensionnement à chaud, la réservation de bande passante pour les entrées et sorties, etc.

La taille maximale théorique (parce que personne n'en a créé de si gros) des fichiers est de 8 Eo (Exaoctets). Sachant que 1 Eo vaut 1024 Po (Petaoctet) donc 1048576 To, soit, rendu à une unité plus appréhendable, environ 1000 milliards de DVD. La partition peut atteindre 16 Eo, soit la capacité maximale d'un contrôleur sur 64 bits. En 32 bits, les tailles sont « limitées » à 16 To.

L'utilisation de xfs est encore peu étendue sous Linux peut-être à cause de sa prétendue complexité pour un paramétrage avancé, mais aussi parce que Red Hat n'a pas de support officiel pour ce système de fichiers dans ses RHEL.

e. vfat

vfat (*Virtual File Allocation Table*) est un terme générique regroupant les diverses versions de FAT supportant les noms de fichiers longs (255 caractères) sous Windows. Ces systèmes de fichiers sont conservés et continuent d'être utilisés pour des raisons à la fois historiques et pratiques. La plupart des supports amovibles, disques externes, clefs USB et lecteurs MP3 utilisent un système de fichiers de ce type. Les raisons sont :

- Un système de fichiers adapté aux petits volumes.
- Un système de fichiers simple à implémenter, idéal pour des lecteurs multimédias.
- Une compatibilité entre diverses plates-formes (Windows, Linux, BSD, MacOS, etc.). vfat souffre cependant de défauts inhérents à sa conception :
- L'ensemble des informations est stockée au sein d'une table unique, y compris le nom du fichier et chaque adresse et longueur des blocs (appelés clusters) composant les données du fichier.
- De ce fait, FAT tente de regrouper les données d'un fichier sur le plus de clusters contigus du support. En cas de nombreuses écritures (ajout, suppression, etc.), le système se retrouve fortement fragmenté.
- Toujours de ce fait, plus le support à une taille importante, plus FAT est lent, car il doit vérifier toute la table
- FAT pour trouver des clusters disponibles.
- La gestion des noms longs est considérée comme une bidouille par de nombreuses personnes car FAT doit continuer à assurer une compatibilité (encore aujourd'hui) avec les noms courts en 8.3.
- Contrairement aux systèmes de fichiers Unix, Linux ou Windows récents, FAT ne gère aucun attribut étendu, notamment aucune notion des droits et des propriétaires.
- FAT est limité à une taille de fichiers de 4 Go. Les fichiers plus gros (bases de données, archives, images ISO de DVD, etc.) doivent être découpés en conséquence et les logiciels (capture audio / vidéo, sgbd, etc.) doivent gérer cette limitation.

Linux gère parfaitement VFAT. Mais son utilisation sur des supports partagés entre Windows et Linux a de moins en moins de raison d'être car l'existence de ntfs3g permet d'utiliser des supports contenant un système de fichiers NTFS de manière native.

Partitionnement

1. Découpage logique

Pour la suite, le support de stockage sera considéré comme un support magnétique ou mémoire de type disque dur, SSD, carte mémoire, etc., c'est-à-dire tout ce qui peut être apparenté à un disque dur selon la vision classique : un espace de données pouvant être découpé en plusieurs entités logiques et indépendantes disposant chacune de leur propre système de fichiers.

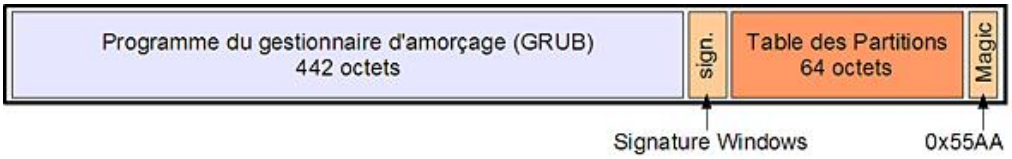
Un disque peut être vu comme une longue bande d'espace de stockage découpée en cases pouvant contenir une quantité donnée d'informations. Le disque peut être utilisé tel quel comme espace de stockage, rien n'empêche de créer un système de fichiers sur un disque sans passer par l'étape de partitionnement. Il est cependant important de donner une organisation logique à cet espace et aux systèmes de fichiers qu'il contiendra, ne serait ce qu'au nom de la séparation des données (les fichiers de données) et des traitements (les programmes les utilisant et le système).

Le 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.

2. Organisation d'un disque

a. 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 la 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.



Format d'un MBR

b. 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.

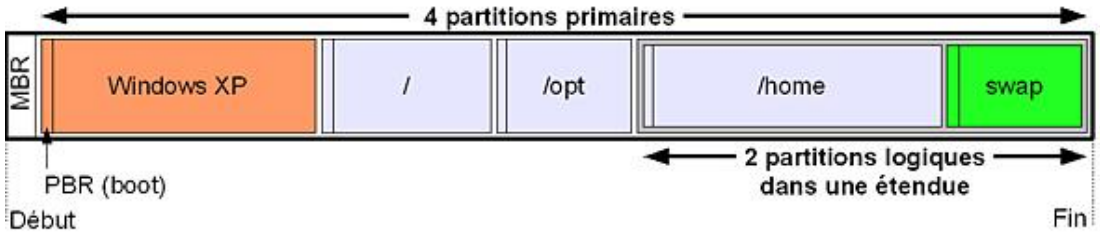
Un disque peut contenir jusqu'à 63 partitions en IDE, 15 en SCSI (c'est une limite de l'implémentation officielle du SCSI) ou via la libata. La limite actuelle est de 15 partitions pour tous les disques avec les derniers noyaux et l'API libata.

Notez bien qu'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 :

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

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.



Description schématique d'un disque

c. 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 indique effectivement la position et la taille d'une partition logique, tandis que le second 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.

- Les EBR forment une liste chaînée, la seconde entrée de partition pointant sur l'EBR suivant.
- Il n'y a qu'une seule partition logique décrite par EBR.

d. 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, voire 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.

e. Types 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. Il semble important d'en fournir une liste ici pour que vous compreniez bien la finalité de cet identifiant. Les valeurs les plus communes sont en gras.

Comme le type de partition devrait refléter 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 devrait contenir un système de fichiers Linux. Mais lequel ? Vous avez vu qu'il en existe plusieurs...

Notez la prédominance des types de partition pour Windows. Windows se base essentiellement sur ce type pour en déterminer le contenu. Rien n'empêche de créer une partition de type Linux et d'y placer un système de fichiers FAT32. Cependant si vous faites ceci, Windows ne reconnaîtra pas la partition (considérée de type inconnu) et vous ne pourrez pas accéder au contenu.

Linux reconnaît généralement (des exceptions sont possibles) le contenu d'une partition par le système de fichiers qui y réside. Vous pouvez créer un système de fichiers ext3 dans une partition de type 0x0e et constater que tout fonctionne. Le type 0x83 peut accueillir tout système de fichiers Linux : ext2, ext3, reiserfs, jfs, xfs... Cependant pour des raisons de compatibilité veillez à respecter l'association type de partition ó système de fichiers, à rester cohérent.

3. Manipuler les partitions

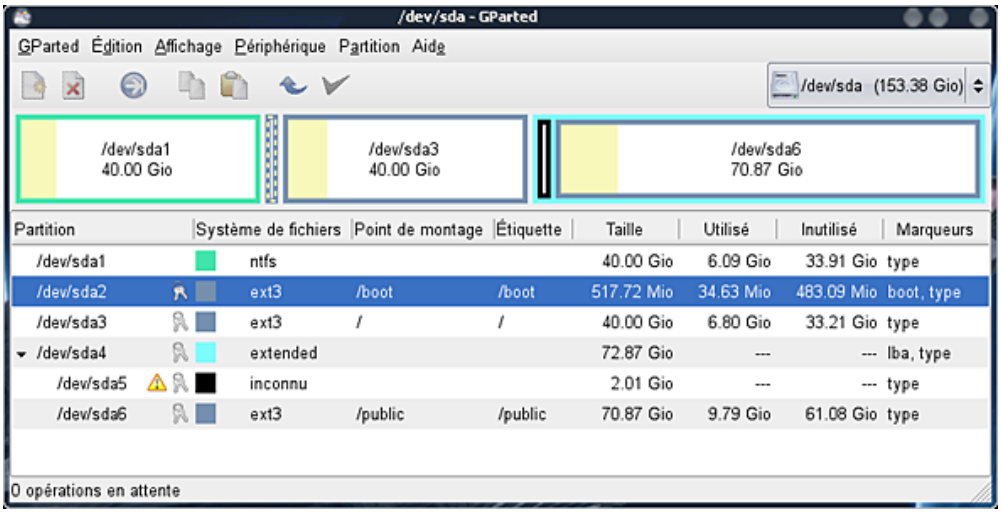
a. Outils disponibles

Les outils **fdisk**, **cfdisk**, **sfdisk** ou encore **parted** permettent de manipuler les partitions, sans compter les outils graphiques disponibles durant l'installation ou dans les panneaux de configuration.

- **fdisk** est le plus ancien et le plus utilisé des outils de partitionnement. Il n'a aucun rapport avec le fdisk de Microsoft. Il est à base de menus et raccourcis textuels.
- **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. Il existe des interfaces graphiques comme **qtparted** ou **gparted**.

Vous pouvez voir sur la capture **gparted** en action.



gparted, un éditeur de partitions graphique

b. Manipuler les 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]
```

Chaque paramètre est optionnel. Lancé tel quel fdisk se place sur le premier disque du système. Le paramètre `-l` permet de lister les partitions du disque donné, ou de tous les disques. Les informations obtenues sont les mêmes qu'en mode interactif avec l'entrée p (print) du menu.

```
# fdisk -l /dev/sda

Disque /dev/sda: 164.6 Go, 164696555520 octets
255 heads, 63 sectors/track, 20023 cylinders
Units = cylindres of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000c02ae

Périphérique Amorce      Début       Fin       Blocs     Id  Système
/dev/sda1                   1         5222     41945683+   7   HPFS/NTFS
/dev                 *      5         5           5       8    Linux
/dev                        5         1           4       8    Linux
/dev/sda                 1            2           7        f   W95 Etendu (LBA)
/dev/sda                 1            1           2        8   Linux swap /
/dev/sda6                10773       20023     74308626   83  Linux
```

Notez la partition sda4 qui est la partition étendue. Elle a comme type 0x0f, mais n'importe quel type étendu aurait fonctionné : les types 0x05 et 0x85 sont considérés comme identiques. Cependant Windows risque de ne pas reconnaître ces types et donc les partitions logiques qui y sont contenues notamment dans le cas d'un gros disque.

Liste des partitions

L'exemple suivant se base sur un disque reconnu par le système comme /dev/sdb, et ne contenant aucune partition. Le but est de créer trois partitions : une primaire, une étendue et une logique.

- Lancez **fdisk** avec le disque en argument, ne tenez pas compte des premières lignes affichées sauf si elles indiquent une erreur.

```
# fdisk /dev/sdb
...
Commande (m pour l'aide)  :
```

- Vérifiez tout d'abord l'existence de partitions avec la touche p (print) puis [Entrée].

```
Commande (m pour l'aide): p

Disque /dev/sdb: 4026 Mo, 4026531840 octets
64 heads, 62 sectors/track, 1981 cylinders
Units = cylindres of 3968 * 512 = 2031616 bytes
Disk identifier: 0x0003ed63

Périphérique Amorce      Début       Fin     Blocs   Id  Système
/dev/sdb1      *           1      1981    3930273   c  W95 FAT32 (LBA)
```

Supprimer

Pour supprimer une partition, 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.

```
Commande (m pour l'aide): d
Partition sélectionnée 1
```

Créer

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

```
Commande (m pour l'aide): n
Action de commande
e       étendue
p       partition primaire (1-4)
```

- Pour cette première partition, sélectionnez une partition primaire avec la touche p (qui veut dire primary cette fois).
- Comme le MBR contient quatre entrées vous pouvez choisir le numéro de partition à créer. Il est parfaitement possible de créer une partition sdb2 avant la sdb1. Ici, tapez **1**.
- Le premier cylindre correspond à la position de début de votre partition. Par défaut fdisk se place sur le premier cylindre disponible depuis le début du disque. Il est parfaitement possible de créer des partitions débutant au milieu d'un disque. Sélectionnez ici la valeur par défaut (1) en appuyant sur [Entrée].
- Enfin choisissez la taille de la partition. Il est préférable d'utiliser une unité lisible comme les Ko ou plutôt les Mo. Créez par exemple une partition de 1 Go, soit 1024 Mo, en saisissant **+1024M** et appuyez sur [Entrée]. La partition est maintenant définie.

```
Commande (m pour l'aide): n
Action de commande
e               étendue
p               partition primaire (1-4)
p
Numéro de partition (1-4): 1
Premier cylindre (1-1981, par défaut 1): Utilisation de la valeur par défaut 1
Dernier cylindre ou +taille or +tailleM ou +tailleK (1-1981, par défaut 1981): +1024M
```

- Vérifiez l'état de la partition (p).

```
Commande (m pour l'aide): p
...
Périphérique Amorce             Début       Fin      Blocs    Id  Système
/dev/sdb1                        1       505     1001889   83  Linux
```

Sauver

Quittez **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. Vous risquez d'obtenir des avertissements ici indiqué en gras.

```
Commande (m pour l'aide): w
La table de partitions a été altérée!

Appel de ioctl() pour relire la table de partitions.

AVERTISSEMENT: la re-lecture de la table de partitions a échoué avec l'erreur 16:
Périphérique ou ressource occupé.
Le kernel va continuer d'utiliser l'ancienne table.
La nouvelle table sera utilisé lors du prochain réamorçage. Synchronisation des disques.
```

Ce message signifie que, comme le périphérique disque est en cours d'accès ou d'utilisation, Linux ne voit pas encore la nouvelle table et donc les nouvelles partitions créées. Ceci peut être confirmé avec la commande suivante. Notez que la dernière ligne devrait vous montrer votre nouvelle partition, ce qui n'est pas le cas.

```
# cat /proc/partitions | grep sdb
8              16    3932160 sdb
```

Forcer la synchronisation

Pour corriger ce dernier problème et forcer le noyau à relire la table des partitions, vous avez le choix entre deux commandes. La première est **blockdev** avec le paramètre `--rereadpt` (re-read partition table).

```
# blockdev --rereadpt /dev/sdb
```

La seconde est **partprobe**, disponible seulement si parted est installé. Elle peut réussir si blockdev a échoué. Par défaut elle relit les tables de toutes les partitions, mais vous pouvez lui spécifier le disque en argument.

```
# partprobe /dev/sdb
```

Vérifiez si la partition est bien reconnue.

```
cat /proc/partitions | grep sdb
```

Vous pouvez maintenant utiliser votre nouvelle partition et y rajouter un système de fichiers. Créez maintenant une partition étendue, puis une partition logique. Remarquez qu'une partition étendue ou logique se crée de la même manière que les autres. Une partition étendue n'est pas forcément la dernière des primaires, elle peut être la seconde par exemple. Si vous n'avez pas utilisé toute la taille pour la créer vous pouvez compléter par la suite la création de partitions primaires.

Manipuler les systèmes de fichiers

1. Définitions de base

a. Bloc

Le bloc est l'unité de base, atomique, de stockage du système de fichiers. Un fichier occupe toujours un nombre entier de blocs. Ainsi 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 !

b. Superbloc

Chaque système de fichiers dispose d'au moins un superbloc. 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 métadonnées (autres superblocs, table d'inodes, etc.).

Linux tente en premier lieu de lire le superbloc primaire, le premier du disque. Il peut arriver que celui-ci soit corrompu suite à de mauvaises manipulations, un crash, une panne. Dans ce cas les données du disque ne sont plus accessibles (impossible de savoir par exemple où se trouve les inodes). Un système de fichiers Linux dispose de copies (backups) des superblocs à plusieurs endroits du disque. Les écritures sur les divers superblocs étant synchrones, ils sont tous identiques. En dernier recours si l'un deux est supprimé, il peut être recopié depuis un autre.

Vous verrez par la suite comment disposer de toutes les informations sur un système de fichiers ext2 ou ext3.

c. Table d'inodes

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 métadonnées.
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. Si le fichier titi porte le numéro d'inode 12345 sur un premier système de fichiers, et que toto porte le numéro d'inode 12345 sur une autre, ces fichiers n'ont aucun rapport entre eux.

Contenu

Le contenu d'un inode varie d'un système de fichiers à un autre, mais la norme POSIX impose que 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) ;
- sa date de dernier accès (access time) ;
- un compteur de hard links (liens physiques ou durs, voir plus loin). Un inode ne contient pas le nom du fichier.

Vous pouvez obtenir quelques informations sur un inode avec la commande **stat** :

```
# stat chapitre4.doc
```

Adresses

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) formant un arbre ou chacune des feuilles (terminaisons) pointe sur un bloc de données. On parle de blocs **d'indirection** (simple, double, triple).

Voici un exemple concret de calculs d'adresses sur un inode au sein d'un système de fichiers ext2.

Un inode ext2 contient dix champs pointant sur un bloc de données chacun, et trois champs d'indirection (pointant sur des adresses).

- Le premier en simple indirection simple pointe sur 256 adresses de blocs de données.
- Le deuxième en double indirection pointe sur 256 adresses dont chacune d'elle pointe sur 256 autres adresses pointant sur des blocs de données, soit 256^2 blocs de données.
- Le troisième en triple indirection pointe sur 256 adresses, elles-mêmes pointant sur 256 adresses, elles-mêmes pointant sur 256 adresses pointant enfin sur des blocs de données, soit 256^3 blocs de données.

Soit n la taille d'un bloc en octet, la taille maximale d'un fichier est donc de **$n*(10+256+256^2+256^3)$ octets**. Pour un bloc de 4096 octets, cela fait environ 64 Go !

d. Tables 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 une liste de fichier, et un fichier étant représenté par un inode, chaque nom de fichier est associé au sein du répertoire à son inode.

Vous pouvez vous représenter cette table comme un tableau à deux colonnes :

Table catalogue rep1 (répertoire rep1)	
Inode	**Nom**
12345	Document.txt
214579	Fichier.doc
47321	Musique.mp3
98542	Copie.odt
...	...

2. Créer un système de fichiers

a. mkfs, syntaxe générale

Les commandes de « formatage » telles que celles présentes sous Microsoft n'existent pas de manière identique sous Linux. Un formatage de type Microsoft est en fait la création et la vérification d'un système de fichiers sur une partition. La première étape est le remplissage des différents secteurs, blocs, et clusters de zéros (ou d'un autre motif binaire) avec une vérification du support, et la seconde l'écriture d'un système de fichiers. Cette seule dernière opération suffit à la création d'un système de fichiers vierge sur le disque ou la partition.

La commande pour créer un système de fichiers est **mkfs**. **mkfs** appelle d'autres programmes en fonction du type de système de fichiers sélectionné.

```
mkfs -t typefs options périphérique
```

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

Plutôt que d'utiliser mkfs, vous pouvez utiliser directement les programmes correspondant au type de système de fichiers à écrire.

b. Un premier exemple en ext2

Vous allez créer un système de fichiers de type ext2 sur la première partition précédemment créée, à savoir sdb1. Voici la commande de base :

```
# mkfs -t ext2 /dev/sdb1
```

c. ext2 et ext3

Les systèmes des fichiers ext2 et ext3 étant compatibles, ils partagent les mêmes paramètres, dont voici les plus courants :

Paramètre	Signification
-b	Taille des blocs en octet, multiple de 512. Si la taille n'est pas précisée, elle sera déterminée par la taille de la partition. Tout fichier créé sur le disque occupe au moins un bloc et donc si on manipule un grand nombre de petits fichiers il faut mettre une valeur basse (ex : 1024).
-c	Vérifie les mauvais blocs avant de créer le système de fichiers. On peut aussi utiliser la commande **badblocks**.
-i	Ratio octets/inode. La taille de la table des inodes est calculée en fonction de la taille totale du système de fichiers. Un inode occupe 128 octets. En mettre moins limite le nombre de fichiers possibles mais permet de gagner de la place. -i 4096 : un inode pour chaque 4 ko.
-m	Pourcentage réservé au super-utilisateur, par défaut 5%. Le mettre à zéro permet de gagner de la place et root pourra tout de même y travailler.
-L	Label, étiquette (nom) du système de fichiers, utile pour le montage.
-j	Crée un journal ext3, donc crée un système de fichiers ext3.

L'exemple suivant crée un système de fichiers journalisé ext3 (option -j) avec une taille de blocs de 2048 octets, et un inode pour chaque 16 Ko. La totalité du système est utilisable par les utilisateurs (aucun espace n'est réservé pour root). L'étiquette est DATA.

```
# mkfs -t ext2 -j -b 2048 -i 16384 -m 0 -L "DATA" /dev/sdb1
```

Notez que la ligne de commande suivante a exactement le même effet car le système de fichiers ext3 induit le paramètre -j :

```
# mkfs -t ext3 -b 2048 -i 16384 -m 0 -L "DATA" /dev/sdb1
```

ext2 vers ext3

Ext3 est un système de fichiers ext2 auquel on a rajouté un journal. Vous pouvez convertir un système de fichiers ext2 en ext3 en utilisant **tune2fs**.

```
# tune2fs -j /dev/sdb1
```

ext3 vers ext2

Pour revenir en ext2, il faut supprimer le journal avec tune2fs et le paramètre -O (grand O) :

```
# tune2fs -O ^has_journal /dev/sdb1
```

Vérifiez l'éventuelle présence d'un fichier **.journal** et supprimez-le. Enfin, effectuez une vérification avec fsck.

Label

Vous pouvez afficher et changer le label du système de fichiers en tapant e2label.

```
# e2label /dev/sdb1
DATA
# e2label /dev/sdb1 OLDDATA
# e2label /dev/sdb1
OLDDATA
```

Accéder aux systèmes de fichiers

1. mount

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).

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

a. Montage par périphérique

La partition sdb1 disposant de nouveau d'un système de fichiers ext3, la commande suivante rattache la racine du système de fichiers contenu dans sdb1 au répertoire /mnt/DATA.

```
# mount -t ext3 /dev/sdb1 /mnt/DATA
```

La commande **mount** utilisée seule donne tous les détails sur les systèmes de fichiers actuellement montés (périphériques, système de fichiers, point de montage, options) :

```
# mount
/dev/sda6 on / type ext3 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
udev on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/sda7 on /home type ext3 (rw,acl,user_xattr)
/dev/sda1 on /windows/C type fuseblk (rw,noexec,nosuid,nodev,noatime,
allow_other,default_permissions,blksize=4096)
securityfs on /sys/kernel/security type securityfs (rw)
/dev/sdb1 on /mnt/DATA type ext3 (rw)
```

Les mêmes informations sont accessibles en affichant le contenu du fichier **/etc/mtab**.

Montage par label

On peut souligner l'intérêt pratique d'utiliser des labels pour ses systèmes de fichiers. En cas de réorganisation des disques (déplacement dans une chaîne SCSI par exemple), l'ordonnancement des périphériques est modifié. L'utilisation des noms de périphériques oblige dans ce cas à modifier le fichier **/etc/fstab** à chaque modification. Ce n'est pas le cas avec les labels. Utilisez le paramètre `-L` de mount, suivi du nom du volume.

```
# mount -t ext3 -L DATA /mnt/DATA
# mount
...
/dev/sdb1 on /mnt/DATA type ext3 (rw)
```

La liste des labels actuellement connus de Linux peut être obtenue en listant le répertoire `/dev/disk/by-label`. Notez que le label est un lien symbolique vers le fichier périphérique correspondant :

```
# ls -l /dev/disk/by-label/
total 0
lrwxrwxrwx 1 root root 10 mar 18 14:00 DATA -> ../../sdb1
lrwxrwxrwx 1 root root 10 mar 18 14:00 DATAFAT -> ../../sdb5
```

Montage par UUID

Chaque système de fichiers dispose d'un identifiant unique appelé **UUID** : *Universal Unique Identifier*, généralement un nombre aléatoire codé sur assez de bits pour que sur un ou plusieurs systèmes donnés, ils soient tous différents. Ainsi si le disque change de position logique, l'UUID ne varie pas et mount retrouve le système de fichiers, alors qu'il est théoriquement bien plus possible que deux systèmes de fichiers portent le même label.

Il existe plusieurs moyens de connaître l'UUID d'une partition. Si udev est utilisé sur votre Linux, alors la commande **vol_id** est probablement disponible. Il se peut qu'elle ne soit pas présente dans le path, par exemple sur une fedora elle se situe dans : /dev/disk/by-uuid/.
Si votre système de fichiers est en ext2 ou ext3 la commande **dumpe2fs** retourne énormément d'informations dont l'UUID :

```
# dumpe2fs -h /dev/sdb1 | grep UUID
dumpe2fs 1.40.2 (12-Jul-2007)
Filesystem UUID:                  67f6e4b8-635c-4103-9a81-877fb7db29fe
```

Pour monter un système de fichiers par UUID, utilisez le paramètre -U de mount :

```
# mount -t ext3 -U 67f6e4b8-635c-4103-9a81-877fb7db29fe /mnt/DATA
# mount
/dev/sdb1 on /mnt/DATA type ext3 (rw)
```

Remonter un système de fichiers

Vous n'êtes pas obligé de démonter puis de remonter un système de fichiers si vous modifiez une option. Si vous modifiez une option de montage du système de fichiers (via le paramètre -o) vous pouvez passer l'option **remount** pour que la modification soit prise tout de suite en compte. Ne retapez pas la ligne de commande complète, mais seulement le périphérique ou le point de montage. Dans l'exemple suivant le système de fichiers est remonté en lecture seule :

```
# mount -o ro,remount /mnt/DATA
# mount
...
/dev/sdb1 on /mnt/DATA type ext3 (ro)
```

b. Options de montage

Option	Signification
Defaults	Souvent présente, l'option **defaults** reprend les options rw, suid, dev, exec, auto, nouser, et async.
sync/async	Active ou désactive les écritures synchrones. Avec async les écritures passent par un tampon qui diffère les écritures (plus performant) rendant la main plus vite. Il est préférable d'activer les écritures synchrones sur des supports externes (clés USB, disques USB/Firewire/eSATA, etc.).
exec/noexex	Permet l'exécution/ou non des fichiers binaires sur le support.
auto/noauto	Le système de fichiers est automatiquement monté/ne peut être monté que explicitement (voir fstab).
user/nouser	N'importe quel utilisateur peut monter le système de fichiers (implique noexec, nosuid, et nodev)/seul root a le droit de monter le système de fichiers (voir fstab).
remount	Remontage du système de fichiers pour la prise en compte de nouvelles options.
ro/rw	Montage en lecture seule ou lecture et écriture.
dev/nodev	Interpréter/Ne pas interpréter les fichiers spéciaux.
usrquota/grpquota	Ignoré par le système de fichiers lui-même mais utilisé par le sous-système de quotas.

Chaque système de fichiers accepte un certain nombre d'options de montage qui peuvent être spécifiées après le paramètre -o de **mount**. Les options sont séparées par des virgules. Sauf indication contraire les options suivantes fonctionnent avec ext2 et ext3.

c. umount

La commande **umount** détache le système de fichiers du point de montage.

```
# umount /mnt/DATA
```

Si un ou plusieurs fichiers du système de fichiers à démonter sont encore en cours d'utilisation, alors **umount** ne marchera pas. Vous devez vous assurer qu'aucun processus n'accède au système de fichiers.

```
# umount /mnt/DATA
umount: /mnt/DATA: périphérique occupé
```

La commande **lsof** vous aide à déterminer quel processus est actuellement en train d'utiliser un fichier du point de montage. Ici c'est le shell bash lancé par l'utilisateur seb qui y est présent (probablement que le répertoire courant est /mnt/DATA).

```
# lsof /mnt/DATA
COMMAND  PID USER     FD   TYPE DEVICE SIZE NODE NAME
bash      5366  seb  cwd    DIR   8,17 4096    2 /mnt/DATA
```

De manière très violente vous pouvez forcer l'arrêt des processus accédant au point de montage avec **fuser**. Il est fort probable que l'utilisateur concerné n'apprécie pas du tout (dans le cas présenté ici son shell sera arrêté et il sera déconnecté).

```
# fuser -km /mnt/DATA
```

d. /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.

```
périphérique point_de_montage typefs options dump fsck
```

Les champs sont séparés par des espaces ou des tabulations.

Champ	Description
périphérique	Le périphérique à monter. Il peut être spécifié en tant que chemin de périphérique (/dev/hda1 par exemple), que label de système de fichiers s'il existe (LABEL=/home), ou encore en tant que UUID (UUID=xxxx).
point de montage	Le répertoire d'accès au système de fichiers monté.
typefs	Le type (ext2, etx3, reiser, vfat, etc.) du système de fichiers.
options	Les options, séparées par des virgules, vues précédemment.
dump	Fréquence de dump pour les outils de dump ou de sauvegarde.
fsck	Fréquence de vérification du système de fichiers.0=ignorer. 1=en premier, 2 en second, etc. Les systèmes ayant le même numéro sont vérifiés en parallèle.

Voici un exemple tronqué (les systèmes de fichiers virtuels n'apparaissent pas) de fichier /etc/fstab :

```
/dev/sda3       /             ext3    defaults                1 1
/dev/sda2       /boot         ext3    acl,user_xattr          1 2
/dev/sdb1       /home         ext3    acl,user_xattr          1 2
/dev/sda6       /public       ext3    acl,user_xattr          1 2
```

Montage au boot

Lors de la séquence de démarrage le fichier **/etc/fstab** est balayé par l'un des scripts, presque au tout début du boot, entre le chargement du noyau et le démarrage des services. Tous les systèmes de fichiers ne possédant pas noauto comme option sont automatiquement montés (le auto est implicite). Le premier à l'être est le système de fichiers racine /. Puis viennent ensuite le swap et les autres systèmes de fichiers s'ils sont spécifiés (ex : /home, /usr, etc.) ainsi que les systèmes de fichiers virtuels /proc, /sys , /dev/pts, etc.

Montage manuel

Le contenu de /etc/fstab peut être utilisé après l'initialisation du système pour monter et démonter ponctuellement les systèmes de fichiers qui n'ont pas par exemple l'option **noauto**, ou les supports de masse comme les lecteurs CD/DVD. Dans ce cas vous utilisez simplement les labels, les points de montage ou le périphérique sans avoir à réécrire toute la ligne de commande. Mount va chercher ses renseignements dans /etc/fstab.

```
mount /home mount -L /u01
mount LABEL=/boot mount /dev/hda5
```

Tout monter

Si vous avez effectué des modifications importantes dans la fstab comme le rajout de plusieurs nouveaux points de montage, vous pouvez, au lieu de monter chaque système de fichiers un par un, tous les monter d'un coup avec le paramètre -a de **mount** : `# mount -a`

e. Cas des CD et images ISO

Les CD-Rom, DVD-Roms et autres supports de ce type se montent comme n'importe quel disque.
Les CD-Roms et certains DVD-Roms utilisent le système de fichiers **iso9660**.

```
# mount -t iso9660 /dev/sr0 /media/cdrom
```

La plupart des DVD-Roms utilisent plutôt le format **UDF** (*Universal Disk Format*).

```
# mount -t udf /dev/sr1 /media/dvd
```

Une image ISO est une image du contenu d'un CD ou d'un DVD. C'est un système de fichiers iso9660 ou udf dans un fichier. Il est possible d'utiliser cette image comme un périphérique, à l'aide des périphériques de loopback. L'image est rattachée à un périphérique de loopback, et les outils passent par ce périphérique comme s'il s'agissait d'un disque.

```
# mount -o loop -t iso9660 image.iso /mnt/iso
```

Contrôler le système de fichiers

1. Statistiques d'occupation

a. Par système de fichiers

La commande **df** permet d'obtenir des statistiques d'occupation de chaque système de fichiers monté. Sans argument, df fournit des informations sur tous les systèmes de fichiers. Vous pouvez passer comme argument un périphérique monté ou un point de montage. Si vous passez un répertoire quelconque, df donne des informations sur le système de fichiers qui contient ce répertoire.

```
# df
Sys. de fich.        1K-blocs       Occupé    Disponible Capacité Monté sur
/dev/sda3            41286828      6482952    32706592  17%       /
/dev/sda2      521780          27092     468184     6%      /boot
/dev/sdb1      153834852       49189572  96830864   34%     /home
/dev/sda6      73142560        19150372  50276760   28%     /public
/dev/sdc1               292890560 175894672 116995888  61%       /media/EXTERNE
```

L'unité par défaut est le kilo-octet (identique au paramètre `-k`) Vous pouvez modifier les paramètres pour demander le résultat en Mo (`-m`).

```
# df -m /home
Sys. de fich.             1M-blocs       Occupé Disponible Capacité Monté sur
/dev/sdb1                    150230     48043     94557          34% /home
```

Pour que ce soit plus lisible, rajoutez le paramètre `-H` (*Human readable*).

```
# df -H /home
Sys. de fich.               Tail.  Occ.  Disp. %Occ. Monté sur
/dev/sdb1                   158G    51G   100G  34% /home
```

Le `-T` rajoute l'affichage du type de système de fichiers.

```
# df -T /home
Sys. de fich. Type         1K-blocs     Occupé Disponible Capacité Monté sur
/dev/sdb1          ext3    153834852  49197688  96822748  34% /home
```

2. Vérifier, régler et réparer

a. fsck

La commande **fsck** permet de vérifier et de réparer un système de fichiers.

```
fsck -t typefs périphérique
```

Le système de fichiers à vérifier ou réparer ne devrait pas être monté, ou alors seulement en lecture seule. Tout comme mkfs, fsck appelle une autre commande selon le type de système de fichiers à vérifier : fsck.ext2, fsck.ext3, etc. Chacune peut prendre des options particulières. Si **fsck** ne reconnaît pas l'option qui lui est fournie, il la transmet au programme concerné. Si vous n'indiquez pas de type, fsck tente de le déterminer seul.

Pour cet exemple, le paramètre `-f` est passé à **fsck** pour forcer la vérification (il n'a pas été possible de produire une corruption) ainsi que le paramètre `-V` pour fournir tous les détails.

```
# fsck -fV /dev/sda2
```

Lorsque le système de fichiers est endommagé, fsck vous pose des questions à chaque action nécessaire. Vous pouvez passer le paramètre `-p` pour tenter une réparation automatique, ou encore `-y` pour forcer les réponses à oui.

Lors du démarrage du système, celui-ci vérifie depuis combien de temps, ou au bout de combien de montage, le système de fichiers n'a pas été vérifié. Si l'intervalle de temps est trop important alors il va exécuter un fsck sur le système de fichiers concerné. Les intervalles peuvent être modifiés par la commande **tune2fs**.

Le swap

1. Pourquoi créer un swap ?

Dans un environnement 32 bits un processus peut théoriquement accéder à 4 Go d'espace mémoire. Il dispose de 4 Go 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.

Que se passe-t-il si un processus n'a plus assez de mémoire pour traiter ses données ? Le système d'exploitation va décharger des segments de la mémoire physique dans une zone d'échange sur disque qui fera office de mémoire virtuelle tampon. Il y a donc un échange entre la mémoire physique et cette zone d'échange, appelé l'espace de swap. Ce processus permet d'utiliser plus de mémoire que l'ordinateur n'en dispose réellement, au prix d'un net ralentissement si le programme est très gourmand.

2. Taille optimale

Il n'y a pas de règles strictes sur la taille du swap. Cependant les quelques règles courantes suivantes sont valables dans la plupart des cas :

- Moins de 512 Mo de RAM : deux fois la RAM.
- 1 Go à 4 Go : la taille de la RAM.
- Plus de 4 Go : 4 Go, plus ou moins, selon l'utilisation des processus.

3. Créer une partition de swap

- Vous savez déjà créer une partition. Créez une partition avec fdisk de la taille souhaitée pour le swap, et donnez-lui le type **83**.
- Synchronisez la table des partitions avec **partprobe**.
- Utilisez la commande **mkswap** pour préparer la partition à recevoir du swap.

```
# mkswap /dev/sda5
```

4. Activer et désactiver le swap

a. Activation dynamique

Linux permet d'activer et de désactiver le swap, ou des morceaux de swap, directement sans avoir à redémarrer le système.

La commande **swapon** permet d'activer une partition de swap :

```
# swapon /dev/sda5
```

Le paramètre -p permet de modifier la priorité du swap. Plus la valeur, comprise entre 0 et 32767, est élevée, plus la priorité d'une zone de swap est élevée. Le système l'utilisera en priorité. Ce paramètre est utile si plusieurs partitions de swap existent sur des disques différents. Dans ce cas, privilégiez soit le disque le plus rapide, soit indiquez une priorité égale pour une meilleure répartition.

La commande **swapoff** désactive une zone de swap. Veillez à disposer de l'espace mémoire libre nécessaire, sinon la commande ne fonctionnera pas.

Le contenu de **/proc/swaps** reflète l'état actuel des zones de swap actives.

```
# cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/sda5                               partition       1461872 2012    -1
```

b. Dans /etc/fstab

Les zones de swap se placent dans le fichier **/etc/fstab**. Voici un exemple :

```
/dev/sda5 swap swap defaults 0 0
```

Les options **noauto** et **pri=X** peuvent être précisées. L'option **pri** permet de définir la priorité de la zone de swap.

Lors du démarrage, le système exécute **swapon -a** qui active toutes les partitions de swap présentes dans la fstab sauf si noauto est précisé.

Lors de l'arrêt, le système exécute **swapoff -a** qui désactive complètement le swap.

5. En cas d'urgence : fichier de swap

Si vous manquez d'espace de swap et qu'il n'est plus possible de créer une nouvelle partition il faut utiliser un fichier d'échange (comme Windows, voire celui de Windows). S'il reste de la place sur un de vos systèmes de fichiers vous pouvez créer dessus un fichier d'échange d'une taille prédéfinie. Ce swap sera moins performant qu'une partition de swap (problème de fragmentation, temps d'accès au système de fichiers).

Voici les manipulations pour un petit swap de 32 Mo :

```
# free | grep Swap
Swap:            2104472       4344    2100128
# dd if=/dev/zero of=/swap bs=1024 count=32768
32768+0 enregistrements lus
32768+0 enregistrements écrits
33554432 bytes (34 MB) copied, 0,35697 s, 94,0 MB/s slyserver:~ # mkswap /swap
Initialisation de la version de l'espace de swap 1, taille = 33550 kB
pas d'étiquette, UUID=b2e5e99e-09a1-4b2d-ac76-59f76526453a slyserver:~ # chmod 600 /swap
slyserver:~ # sync
slyserver:~ # swapon -v /swap swapon sur /swap
slyserver:~ # free | grep Swap
Swap:            2137232       4308    2132924
```

Modifiez éventuellement le fichier **/etc/fstab**, en espérant que le swap soit activé après le montage des systèmes de fichiers. Le swap est activé au boot, généralement après le montage de /. Mais s'il est ailleurs (autre point de montage) alors comme le swap est activé avant les autres points de montage, il en résultera une erreur. Il est donc préférable de créer le fichier dans le système de fichiers racine /.

```
/swap swap swap defaults 0 0
```

Les quotas disques

1. Définitions

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. Pour chaque utilisation (inode ou bloc), vous pouvez 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 but est qu'il revienne à terme sous la limite douce. Le délai dépassé, la limite douce devient la limite dure. Quoi qu'il arrive, l'utilisateur ne pourra jamais dépasser la limite dure.

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.

2. Mise en place

Vous allez mettre en place les quotas sur la partition /home en respectant les étapes suivantes :

- Modifiez les options de partition dans **/etc/fstab**. On rajoute dans les options usrquota (utilisateur) ou grpquota (groupe), ou les deux.

```
LABEL=/home /home ext3 defaults,usrquota 1 2
```

- Remontez le système de fichiers.

```
# mount -o remount /home
```

- Mettez à jour la base de données avec la commande **quotacheck**.

```
# quotacheck -c /home
```

- Démarrez (ou arrêter) les quotas. Cette opération n'est pas nécessaire après un redémarrage de Linux car la mise en place des quotas est comprise dans les scripts de démarrage. La commande **quotaon** démarre les quotas pour le système de fichiers indiqué (-a pour tous). La commande **quotaoff** stoppe les quotas.

```
# quotaon /home
```

- Éditez les quotas pour les utilisateurs ou les groupes. La commande **edquota** est utilisée. En pratique, si tous les utilisateurs doivent avoir les mêmes quotas ou avec quelques variantes, on crée un utilisateur lambda dont on recopiera les propriétés.

Établir les quotas pour elies :

```
# edquota roger # = edquota -u elies
```

Les quotas de anouar sont identiques à ceux de elies :

```
# edquota -p elies anouar
```

- Établissez le délai de grâce. Le délai accepte les unités « seconds », « minutes », « hours », « days », « weeks », « monthes ».

```
# edquota -t
```

- Vérifiez les quotas. Les utilisateurs peuvent vérifier l'état de leurs quotas avec la commande **quota**. L'administrateur peut générer un rapport avec **repquota**. Enfin la commande **warnquota** qui peut être exécutée via cron peut envoyer un mail aux utilisateurs pour les prévenir en cas de dépassement.

L'édition des quotas se fait avec l'éditeur par défaut du système qui est généralement vi. Les blocs de quotas sont des blocs de 1 Ko.

```
# edquota elies
Disk quotas for user elies (uid 502):
Filesystem     blocks        soft          hard          inodes  soft    hard
/dev/hda5      1695256       2500000       3000000       12345   0       0
```

Avec l'éditeur, on peut modifier les valeurs soft et hard qui correspondent aux limites douces et dures pour le nombre de blocs et le nombre d'inodes. Ci-dessus, il a été établi une limite douce à environ 2,4 Go (2500000 ko) et dure à environ 2,9 Go (3000000 ko) d'occupation du système de fichiers pour elies. Il n'y a pas de quotas sur le nombre d'inodes (valeur à 0).

Partitionnement avancé RAID

1. Définitions

Le **RAID** (*Redundant Array of Inexpansive Disks*) a été défini par l'université de Berkeley en 1987 dans le double but de réduire les coûts et d'augmenter la fiabilité du stockage des données. Le but est de combiner plusieurs petits disques physiques indépendants en une matrice (array : tableau, ensemble, rangée, matrice) de disques dont la capacité dépasse celle du SLED (*Single Large Expensive Drive*). Une matrice apparaît comme une unité logique de stockage unique.

Le **MTBF** (*Mean Time Between Failure* - temps moyen entre pannes) de l'ensemble est égal au MTBF d'un disque individuel divisé par le nombre de disques dans l'ensemble et donc théoriquement, une solution RAID peut être inadaptée pour des tâches critiques. Heureusement, le RAID peut être tolérant aux fautes en stockant de manière redondante ses informations selon plusieurs méthodes :

- **RAID-0 :** appelé **stripe mode** : deux disques au moins forment un seul volume. Les deux disques ont en principe la même taille. Chaque opération de lecture/écriture sera fractionnée et effectuée sur chacun des disques. Par exemple, 4 ko seront écrits sur le disque 0, 4 ko sur le disque 1, 4 ko sur le disque 2, puis 4 ko sur le disque 0, etc. Ainsi, les performances sont accrues puisque les opérations de lecture et d'écriture sont effectuées en parallèle sur les disques. Si N est le nombre de disques et P la vitesse de transfert, la vitesse de transfert du volume RAID est en principe proche de N*P mbps. Le RAID-0 n'a aucune redondance. En cas de panne d'un des disques, il est probable que l'ensemble des données soit perdu.
- **RAID-1 :** appelé **mirroring** : premier mode redondant. Il peut être utilisé à partir de deux disques ou plus avec d'éventuels disques de secours (*Spare Disk*). Chaque information écrite sur un disque est dupliquée sur les autres. Si N-1 disques du RAID viennent à tomber, les données restent intactes. Si un disque de secours est présent, en cas de panne, il est automatiquement reconstruit et prend la place du disque défaillant. Les performances en écriture peuvent être mauvaises : écriture sur N disques en même temps, risquant de saturer le contrôleur disque et le bus. Les performances en lecture sont bonnes, car RAID emploie un algorithme qui peut lire les données sur chaque disque (puisqu'ils sont identiques).
- **RAID-5 :** RAID avec bande de parité redistribuée. C'est le mode le plus utilisé car c'est celui qui offre le meilleur compromis entre le nombre de disques, l'espace disponible et la redondance. Il faut au moins trois disques avec d'éventuels disques de secours. La parité est présente sur chacun des disques. La taille finale est celle de N-1 disques. Le RAID-5 survit à une panne de disque. Dans ce cas, si un disque de secours est présent, il sera automatiquement reconstruit. Les performances en lecture sont équivalentes à celles du RAID-0 tandis qu'en écriture, elles dépendent de l'algorithme employé et de la mémoire de la machine.

2. Précautions et considérations d'usage

a. Disque de secours

Un disque de secours (*Spare Disk*) ne fait pas partie intégrante d'une matrice RAID tant qu'un disque ne tombe pas en panne. Si cela arrive, le disque est marqué défectueux et le premier disque Spare prend le relais. Quoi qu'il arrive, il faut tout de même, le plus vite possible, changer le disque défaillant et reconstruire le RAID.

b. Disque défectueux

Un disque défectueux (*Faulty Disk*) est un disque qui a été reconnu défaillant ou en panne par le RAID. Dans ce cas, RAID utilise le premier disque Spare pour reconstruire sa matrice. Les disques Faulty appartiennent toujours à la matrice mais sont désactivés.

c. Boot

La partition de boot (celle qui contient le noyau, la configuration du bootloader, les fichiers images de disques) ne doit pas être placée dans une matrice RAID : le chargeur de démarrage est incapable de monter des partitions RAID (la prochaine version de GRUB en sera capable).

d. Swap

Vous pouvez installer un swap sur du RAID mais ce n'est en principe pas utile dans les cas courants. En effet, Linux est capable d'équilibrer l'utilisation du swap sur plusieurs disques/partitions seuls. Dans ce cas, déclarez n swaps dans **/etc/fstab** avec la même priorité.

```
/dev/sda2    swap          swap        defaults,pri=     0 0
/dev/sdb2    swap          swap        defaults,pri=1    0 0
/dev/sdc2    swap          swap        defaults,pri=1    0 0
```

Cependant, en cas de besoin de haute disponibilité, le swap sur le RAID est possible.

e. Périphériques

Une matrice RAID est reconnue par le système comme un périphérique de type bloc, comme n'importe quel disque physique. Ainsi, un RAID peut être constitué avec des disques, des partitions (généralement, on crée une unique partition sur chaque disque). Le bus n'a aucune importance : vous pouvez construire une matrice RAID avec des disques SCSI et IDE mélangés. De même, on peut construire du RAID sur d'autres matrices RAID, par exemple du RAID-0+1 (2x2 disques en RAID-1, les deux matrices résultantes en formant une nouvelle en RAID-0). Les périphériques RAID sont sous la forme :

/dev/md0
/dev/md1

f. IDE

Si les disques IDE ont longtemps été le SCSI du pauvre (matériel de moins bonne qualité, lenteur, manque de fiabilité) ce n'est plus vraiment le cas. Les derniers modèles sont totalement identiques aux disques SCSI, contrôleur excepté. Vous pouvez donc monter pour un coût raisonnable des configurations RAID en IDE. Cependant une règle est à retenir :

UN SEUL DISQUE IDE PAR BUS IDE

En pratique, cela correspond à un disque par câble, sans rien d'autre. En effet, un bus IDE survit en principe à la déficience d'un disque mais il arrive régulièrement que le bus IDE devienne lui-même défectueux, entraînant la perte du second disque présent sur le bus et donc la perte de la matrice RAID. L'achat de cartes IDE supplémentaires (bas prix) permet de compenser le problème de fiabilité (deux disques par carte).

g. Hot Swap

- **IDE** : NE JAMAIS DEBRANCHER À CHAUD UN DISQUE IDE ! C'est le meilleur moyen de détruire le disque, si ce n'était pas encore le cas et de détruire le contrôleur IDE (et donc éventuellement la carte mère ou additionnelle). L'IDE n'est pas prévu pour.
- **SCSI** : les contrôleurs SCSI ne sont pas prévus pour le Hot Swap mais devraient en théorie tout de même fonctionner, si le disque est identique physiquement et logiquement.
- **SATA** : le SATA est reconnu comme du SCSI. La spécification SATA en version 2 supporte théoriquement le Hot Swap. Seulement, la plupart des contrôleurs actuels implémentent mal ou pas du tout cette possibilité, d'où les risques de plantages ou de griller son contrôleur. Référez-vous à la documentation du constructeur de votre carte mère (chipset).
- **SCA** : ce sont des disques SCSI spécifiques. Consultez le document « Software RAID Howto ».

3. RAID avec mdadm

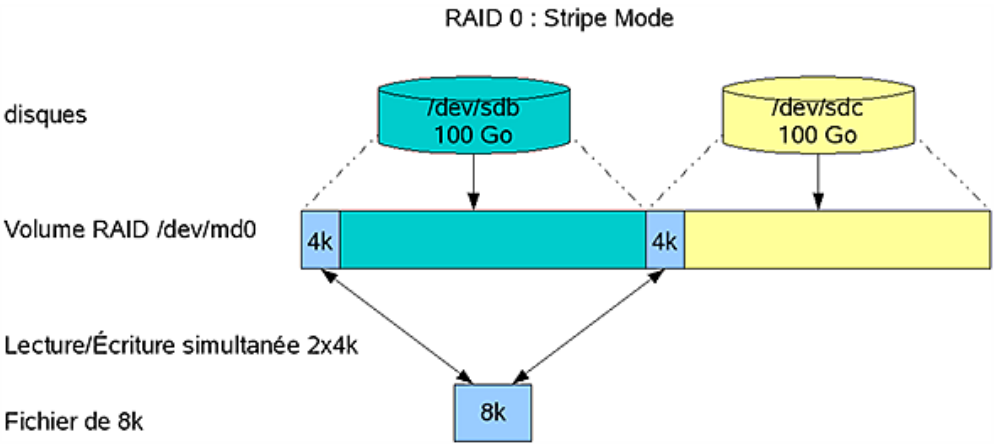
a. Préparation

L'outil **mdadm** remplace les outils raidtools des anciennes distributions Linux. Cet outil unique est plus simple et permet d'effectuer l'ensemble des opérations. Son fichier de configuration est **/etc/mdadm.conf**.

Afin de créer des matrices RAID, il faut que les partitions qui vont servir à créer la matrice soient de type **0xFD** (Linux RAID autodetect). Les partitions doivent être logiquement sur des disques différents, mais pour des tests, le support RAID autorise des partitions sur le même disque. Dans ce cas, vous veillerez à ce que les partitions disposent de la même taille.

b. Création

RAID-0



Principe du RAID 0 : Striping

Soient deux partitions **/dev/sdb1** et **/dev/sdc1**. Vous allez créer une partition RAID-0, assemblage de ces deux
partitions.

```
# mdadm --create /dev/md0 --level=raid0 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

`--create` : Créer un RAID.

`/dev/md0` : Nom du fichier périphérique de type bloc représentant la matrice RAID.

`--level` : Type de RAID à créer : `0`, `raid0` et `stripe` pour du RAID0.

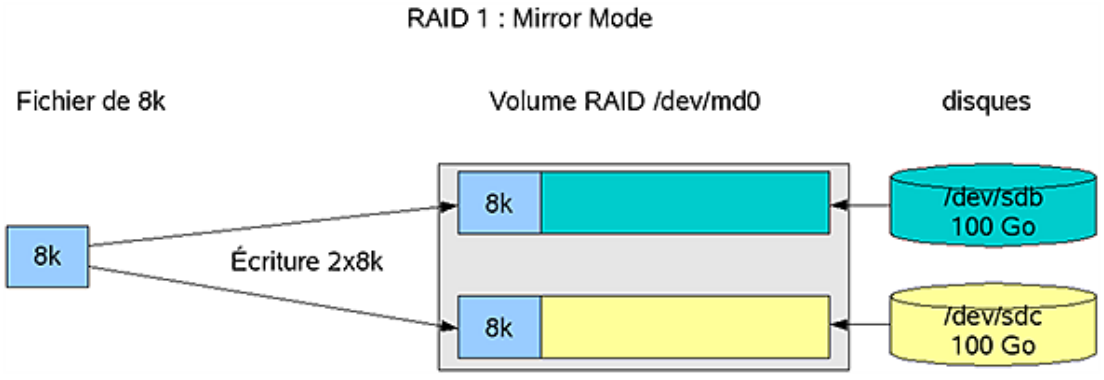
`--raid-devices` : Nombre de partitions utilisées pour créer la matrice.

`/dev/sdb1, /dev/sdc1` : Partitions constituant la matrice, suivant le nombre indiqué dans `--raid-devices`.

Il ne reste plus qu'à installer le système de fichiers sur le disque RAID :

```
# mkfs -t ext3 /dev/md0
```

RAID-1



C'est le même principe. Vous allez cette fois rajouter une partition de secours **/dev/sdd1**.

```
# mdadm --create /dev/md0 --level=raid0 --raid-devices=2 /dev/sdb1 /dev/sdc1
--spare-devices=1 /dev/sdd1
```

`--level 1`, `mirror` ou `raid1` sont de bonnes valeurs pour un RAID-1.

`--spare-devices` nombre de disques de secours à utiliser.

`/dev/sdd1` partitions constituant les disques de secours, suivant le nombre indiqué dans `-spare-devices`. Puis :

```
# mkfs -t ext3 /dev/md0
```

RAID-0+1

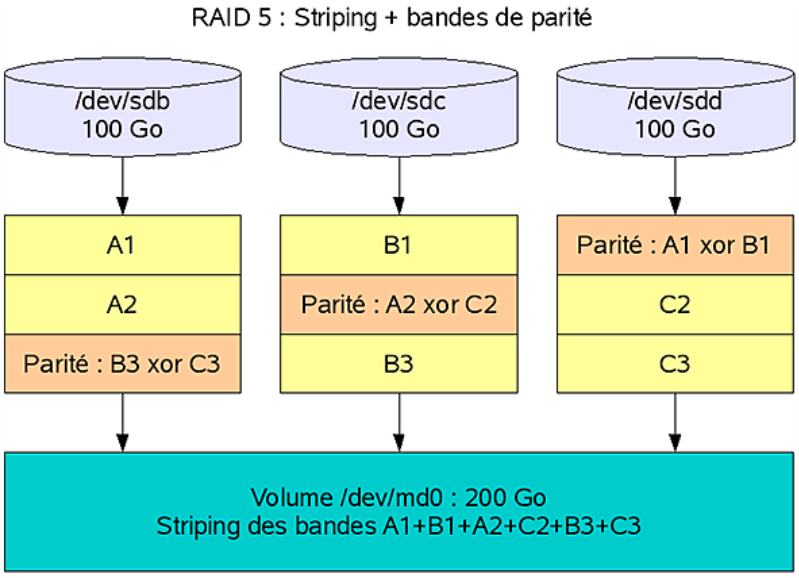
Il faut au moins quatre partitions. Vous devez créer deux matrices RAID-1 que vous allez regrouper en une matrice RAID-0.

```
# mdadm --create /dev/md0 --level=raid1 --raid-devices=2 /dev/sdb1 /dev/sdc1

# mdadm --create /dev/md1 --level=raid1 --raid-devices=2 /dev/sdd1 /dev/sde1

# mdadm --create /dev/md2 --level=raid0 --raid-devices=2 /dev/md0 /dev/md1
```

Puis :

```
# mkfs -t ext3 /dev/md2
```

RAID 5



Vous allez utiliser trois disques de données /dev/sdb1, /dev/sdc1, /dev/sdd1 et un disque de secours /dev/sde1.

```
# mdadm --create /dev/md0 --level=raid5 --raid-devices=3 /dev/sdb1 /dev/sdc1
/dev/sdd1 --spare-devices=1 /dev/sde1
```

Puis on formate :

```
# mkfs -t ext3 /dev/md2
```

4. État du RAID

Le fichier virtuel **/proc/mdstat** contient des informations sur le RAID. C'est ici que vous pouvez voir le détail d'un RAID, notamment si un des volumes de la matrice est défectueux (Faulty).

La commande **watch** permet de vérifier un état en continu :

```
# watch cat /proc/mdstat
```

Vous pouvez aussi utiliser **mdadm** avec le paramètre `--detail`

5. Simuler une panne

Vous allez simuler une panne sur **/dev/hda8** :

```
# mdadm /dev/md0 -f /dev/hda8

mdadm: set /dev/hda8 faulty in /dev/md0
```

Remarquez qu'un « (F) » est apparu près de hda8, indiquant un disque Faulty. On voit aussi que sur les deux disques, un est en panne et que le RAID reconstruit sa matrice avec le spare disk. Le RAID se reconstruit et fonctionne correctement.

```
# mdadm --detail /dev/md0
```

Le disque Faulty est bien **/dev/hda8** ; **/dev/hda10** a pris sa place en tant que disque de secours. Ainsi, le disque de secours devient un disque RAID de la matrice.

6. Remplacer un disque

Puisque **/dev/hda8** est en panne, vous allez le remplacer. Retirez-le avec `-r` (ou `--remove`) :

```
# mdadm /dev/md0 -r /dev/hda8
mdadm: hot removed /dev/hda8
```

Constatez que hda8 a disparu. Vous pouvez éteindre la machine puis remplacer le disque défaillant. Rallumez la machine, puis repartitionnez le disque correctement. Il n'y a plus qu'à rajouter le disque réparé dans la matrice RAID avec `-a` (`--add`) :

```
# mdadm /dev/md0 -a /dev/hda8
mdadm: hot added /dev/hda8
```

Le disque hda8 apparaît à nouveau. Voyez le détail :

```
# mdadm --detail /dev/md0
```

Le disque **/dev/hda8** a été remis et est devenu le nouveau disque de secours !

7. Arrêt et relance manuels

Vous pouvez arrêter une matrice RAID avec `-S` (`--stop`) APRÈS avoir démonté le périphérique :

```
# mdadm --stop /dev/md0
```

Vous redémarrez une matrice RAID avec `-As` (`--assemble -scan`). Cela implique que le fichier **/etc/mdadm.conf** est correctement renseigné (`--scan` recherche les informations dedans).

```
# mdadm --assemble --scan /dev/md0
```

Si le RAID ne redémarre pas, vous pouvez tenter avec `-R` (`--run`) : il est probable qu'il manque un disque ou qu'une reconstruction en cours n'est pas terminée :

```
# mdadm --run /dev/md0
```

Initiation au LVM

1. Principe

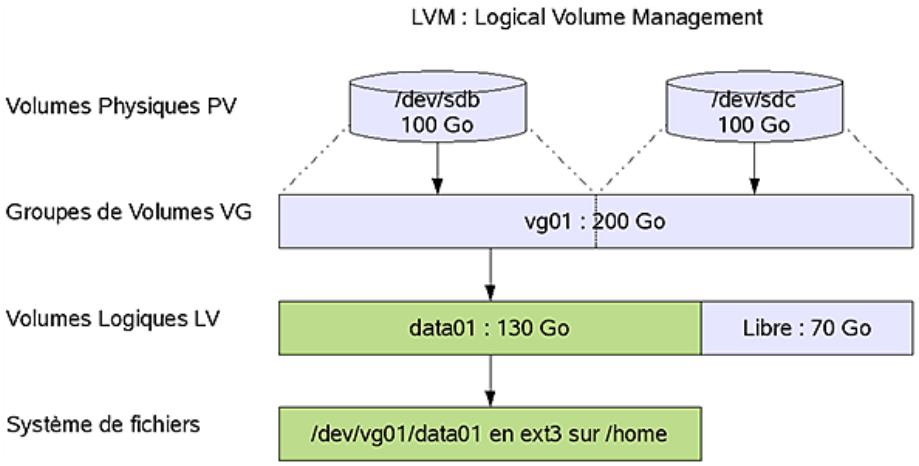
Le **Logical Volume Manager** est un système de gestion très perfectionné des supports de stockage. Le but est de dépasser la gestion physique des disques, et leur organisation logique basique (les partitions) pour étendre la capacité globale des supports, à l'aide d'une gestion entièrement logique de celle-ci.

Un LVM permet, tout comme le RAID 0 par exemple, de créer des espaces de données logiques sur plusieurs disques.
Il permet aussi de faire du mirroring, comme le RAID 1. Mais la comparaison s'arrête là. Le RAID logiciel se contente de créer une « partition » dans un espace de stockage défini par le RAID lui-même (par exemple une partition de 100 Go dans un RAID 0 de deux disques de 50 Go).

Le LVM regroupe les disques physiques, ou tout autre support de stockage dit physique (disque, RAID matériel, RAID logiciel, support de stockage en provenance d'un SAN), qu'il appelle des volumes physiques PV (Physical Volume) en un groupe de volumes VG (Volume Group). Ce groupe VG est vu par le LVM comme une sorte de métadisque, dans lequel vous allez créer des volumes logiques LV (Logical Volume) à volonté.

- Volume physique PV : un support de stockage de données dit physique : disque dur par exemple ;
- Groupe de volumes VG : un regroupement logique de 1 à n VG ;
- Volume logique LV : un découpage logique au sein d'un VG.

Un volume logique est vu comme une partition, et est utilisable comme telle. Il peut contenir des données, il suffit de créer un système de fichiers ordinaire (ext3 par exemple) et de le monter de manière tout à fait classique.



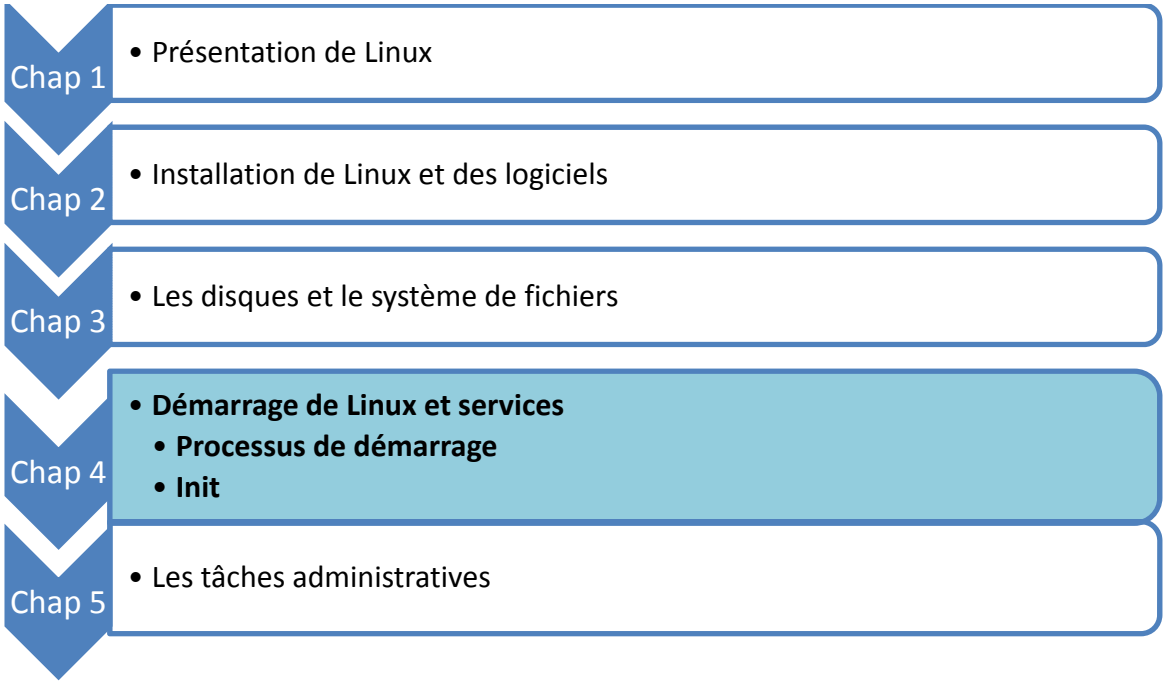
Contrairement au RAID 0 Logiciel où la partition de données doit occuper tout l'espace, il est possible de créer autant de volumes logiques de toute taille que souhaité. Mais cela va bien plus loin.

Le LVM est dynamique. Il est possible d'ajouter et de supprimer des volumes physiques d'un groupe de volumes. En ajoutant des volumes physiques, la capacité, et donc l'espace disponible du groupe augmente. Le nouvel espace disponible peut permettre de créer des nouveaux volumes logiques, mais aussi d'agrandir un volume logique existant.

Un volume logique est dynamique : il peut être agrandi ou réduit à volonté, ce qui implique qu'il faut aussi pouvoir agrandir un système de fichiers, ou le réduire.

Notez enfin qu'une matrice RAID peut être utilisée comme volume physique.

La configuration du LVM est située dans les fichiers et répertoires présents dans **`/etc/lvm`**. Le fichier **`/etc/lvm/lvm.conf`** contient la configuration globale. La configuration des différents volumes (physiques, groupes et logiques) ne se trouve pas dans un fichier, mais dans une structure présente au sein des périphériques eux-mêmes, dans les premiers blocs : ce sont les métadatas des volumes physiques.


Chap 1
• Présentation de Linux
Chap 2
• Installation de Linux et des logiciels
Chap 3
• Les disques et le système de fichiers
Chap 4
• Démarrage de Linux et services
• Processus de démarrage
• Init
Chap 5
• Les tâches administratives

Processus de démarrage

1. Le BIOS

a. Rôle

Le **BIOS** (*Basic Input Output System*) est l'interface logicielle entre le matériel et le logiciel à un niveau très basique. Il fournit l'ensemble des instructions de base utilisées par le système d'exploitation. Il fournit le niveau d'interface le plus bas aux pilotes et périphériques.

Le BIOS effectue un auto-test de l'allumage (POST) puis recherche les périphériques, notamment ceux utilisés pour démarrer. Les informations sur le matériel sont stockées de manière permanente dans une petite mémoire CMOS alimentée par une batterie. À la fin du processus, le périphérique de démarrage est sélectionné.

Le BIOS lit et exécute le premier secteur physique du média de démarrage. Il s'agit généralement des 512 premiers octets du premier disque dur (le MBR) ou de la partition active (le PBR).

2. Le chargeur de démarrage

Le BIOS active le chargeur de programme initial (*Initial Program Loader*, IPL) à partir des premiers 512 octets du support de démarrage. Sur Linux, le chargeur est décomposé en deux parties. Le chargeur initial des 512 octets ne contient pas assez de code pour proposer des menus et lancer le chargement d'un système d'exploitation. Il charge la seconde phase, basée sur un fichier de configuration.

La seconde phase fournit une interface pour lancer un système d'exploitation parmi un choix donné. Vous pouvez en profiter pour passer des paramètres au noyau Linux et au processus init.

Le BIOS n'intervient qu'au démarrage de la machine, à l'utilisation du chargeur de démarrage et aux toutes premières étapes du chargement du noyau. Ensuite, il devient inutile. Le noyau dispose de ses propres fonctions de détection bien qu'il s'appuie sur la configuration du BIOS.

3. GRUB

a. Configuration

Le chargeur par défaut sur la plupart des distributions Linux s'appelle **GRUB** (*Grand Unified Bootloader*). Il est hautement paramétrable, notamment en acceptant une protection par mot de passe crypté, un interpréteur de commandes ou encore des graphiques. Il est basé sur un fichier texte de configuration et il n'y a pas besoin de réinstaller GRUB à chaque modification.

Voici un exemple de configuration partant du principe que la première partition du premier disque est /boot, et que la seconde contient une installation de Windows.

```
timeout=10
default=0
title Red Hat
        root (hd0,0)
        kernel /vmlinuz-2.6.12-15 ro root=LABEL=/
        initrd /initrd-2.6.12-15.img
title Windows XP
        rootnoverify (hd0,1)
        chainloader +1
```

Voici la syntaxe générale d'un fichier GRUB :

Paramètre GRUB	**Signification**
timeout	Nombre de secondes avant le démarrage par défaut.
default n	Démarrage par défaut (0=1er titre, 1=2ème titre, etc.).
title xxxx	Début d'une section, entrée du menu de GRUB.

root(hdx,y)	Tous les accès fichiers spécifiés dessous le seront à partir de cette partition (cf. signification plus bas). Ici hd0,0 représente la première partition du premier disque détecté par le BIOS. C'est la partition /boot.
kernel	Le nom de l'image du noyau Linux, suivi de ses paramètres. Le / n'indique pas la racine du système de fichiers mais celle de (hd0,0), donc /boot/vmlinuz...
initrd	Initial ramdisk. Le noyau va charger ce fichier comme disque en mémoire pour y trouver une configuration et des pilotes initiaux.
rootnoverify	La racine spécifiée, à ne pas monter par GRUB (il ne supporte pas NTFS).
chainloader +1	Démarrer le premier secteur de la racine spécifiée ci-dessus.

Voici la signification des noms de périphériques sous GRUB.

- (fd0) : premier lecteur de disquettes détecté par le BIOS (/dev/fd0 sous Linux).
- (hd0,0) : première partition sur le premier disque dur détecté par le BIOS que ce soit IDE ou SCSI (/dev/hda1 ou /dev/sda1 suivant le cas).
- (hd1,4) : cinquième partition sur le second disque dur détecté par le BIOS (/dev/hdb5 ou /dev/sda5).

b. Installation

La configuration de **GRUB** réside dans **`/etc/grub.conf`** ou **`/boot/grub/menu.lst`** (le premier est un lien sur l'autre). GRUB peut s'installer sur un **MBR** (*Master Boot Record*, les 512 premiers octets d'un disque) ou un **PBR** (*Partition Boot Record*, les 512 premiers octets d'une partition).
Pour installer ou réinstaller GRUB en cas de MBR corrompu, par exemple sur **`/dev/sda`** utilisez la commande **grub-install** :

```
# /sbin/grub-install /dev/sda
```

4. Initialisation du noyau

Au chargement du noyau une multitude d'informations défile sur l'écran. Vous ne pouvez pas figer ces informations à ce moment là. Par contre juste après le passage à l'étape suivante (init) toutes les traces du noyau sont placées dans le fichier **`/var/log/dmesg`**.

- Le matériel est détecté et initialisé.
- initrd est chargé, les modules présents éventuellement chargés.
- Le noyau monte le système de fichiers racine en lecture seule.
- Il crée la première console.
- Le premier processus est lancé (normalement init).

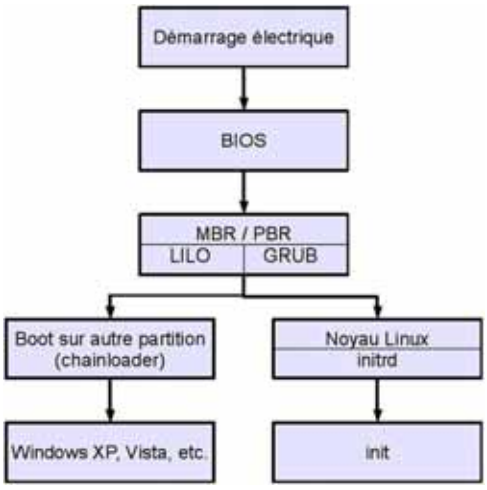


Schéma de la séquence de démarrage

init

1. Rôle

Le programme init, premier démarré et dernier stoppé au sein du système, est celui qui lance toutes les autres tâches. Le rôle initial de init est de démarrer et d'arrêter tous les services. C'est init qui va exécuter les diverses tâches initiales nécessaires au bon fonctionnement de Linux via l'exécution de plusieurs commandes et scripts.

Une fois le système démarré et les services lancés, init reste actif pour gérer les changements d'état des processus qu'il contrôle et des niveaux d'exécution.

Le processus **init** est le père de tous les processus. Il a toujours le PID 1. Sa configuration est présente dans le fichier **/etc/inittab**. Si ce fichier est corrompu et inutilisable, il faudra démarrer en mode single (S, s, 1, Single) et le réparer, ou au pire démarrer depuis un support externe ou un disque de secours. C'est un fichier central du système d'exploitation.

2. Niveaux d'exécution

Un niveau d'exécution, ou runlevel, est un état dans lequel se trouve Unix/Linux. Cet état est contrôlé par init. Chaque état dispose de sa propre configuration (soit par inittab, soit par des scripts appelés initscripts). Un niveau d'exécution peut par exemple être utilisé pour lancer Unix en mono-utilisateur, en multi-utilisateurs, avec ou sans réseau, avec ou sans mode graphique. Tous les niveaux sont personnalisables par l'administrateur. Ces niveaux sont généralement définis comme ceci par convention sur les distributions Red Hat/Fedora, Mandriva, openSUSE et associées :

Niveau	**Effet**
0	Halt : stoppe le système d'exploitation, éteint la machine.
1	Mode mono-utilisateur utilisé pour la maintenance, mode console.
2	Multi-utilisateur, sans réseau, console.
3	Multi-utilisateur, avec réseau, console.
4	Idem que le 3, laissé à la convenance de l'administrateur.
5	Multi-utilisateur, avec réseau, avec environnement graphique X Window.
6	Reboot : redémarrage de la machine.
S,s	Single user mode, le mode le plus bas en cas de soucis.

Les niveaux 7 à 9 sont parfaitement valides mais pas utilisés par défaut. Le niveau d'exécution par défaut est positionné dans **/etc/inittab** sur la ligne initdefault.

```
id:5:initdefault:
```

Remplacez 5 par le niveau souhaité au démarrage.

3. /etc/inittab

Le comportement du processus init et des runlevels est défini dans le fichier **/etc/inittab**. La syntaxe d'une ligne est la suivante :

```
Id:[niveaux]:action:commande
```

Champ	Description
Id	Identifiant de ligne sur quatre caractères, juste indicatif (sous Linux avec getty/mingetty : numéro de terminal)
Niveaux	Indique si la commande doit être prise en compte pour le niveau demandé, c'est la liste des niveaux sans séparateur.
Action	Type d'action à effectuer selon les circonstances pour cette ligne.
Commande	La commande à exécuter avec ses paramètres et les redirections.

L'action est très importante car elle définit les activités de init lors du démarrage et du changement de niveau. Voici les principales :

Action	Signification
initdefault	Définit le niveau par défaut lors du boot et du lancement d'init.
sysinit	Exécuté une seule et unique fois lors du démarrage du système.
wait	Idem, mais init attend la fin de l'exécution avant de continuer.
respawn	La commande est lancée pour les niveaux concernés. Si le processus se termine, il est automatiquement relancé. C'est le cas pour les terminaux si un utilisateur s'en déconnecte.
powerfail	Idem, mais sans attente de la fin d'exécution de la commande.
powerokwait	La commande est lancée lorsque le courant est rétabli.
ctrlaltdel	Init reçoit un signal SIGINT issu d'une séquence [Alt][Ctrl][Suppr].

4. Changement de niveau

Vous pouvez changer de niveau à la volée après le démarrage de la machine avec la commande **/sbin/init** ou **/sbin/telinit**, cette dernière étant un simple lien symbolique vers init. La commande suivante passe au niveau 5.

```
# telinit 5
```

Les valeurs q, u ou -t peuvent être précisées :

- Q ou q : init relit le fichier **/etc/inittab**, s'il a été modifié, en corrigeant ses tables internes.
- U ou u : init se relance sans relire inittab et sans changer de niveau. Si des services ont été rajoutés ou supprimés du niveau en cours, init prend en compte la modification.
- -t : quand init a terminé l'arrêt des services (ou plutôt quand le script rc l'a fait, voir un peu plus loin), init envoie le signal SIGTERM à tous les processus restants, leur demandant de se terminer proprement, attend le nombre de secondes spécifié (5 par défaut), puis envoie SIGKILL.

Le niveau d'exécution actuel est visible avec la commande **/sbin/runlevel**. La première valeur retournée est le niveau précédent le niveau actuel. Un N signifie qu'il n'y a pas de précédent niveau. La seconde valeur est le niveau actuel.

```
# runlevel
N 5
```

5. Paramétrage système de base

Quel que soit le niveau d'exécution précisé par défaut, init lance toujours la commande associée aux actions sysinit, bootwait ou boot de **/etc/inittab** lors du démarrage du système, l'action sysinit étant la première.

- Sous Red Hat : `si::sysinit:`**`/etc/rc.d/rc.sysinit`**

Sous Red Hat, c'est un seul script monolithique qui s'occupe de toute la configuration de base.

Dans tous les cas, les tâches suivantes sont exécutées à peu près dans cet ordre :

- Configuration des paramètres du noyau présents dans **/etc/sysctl.conf**.
- Mise en place des fichiers périphériques (/dev via udev par exemple).
- Configuration de l'horloge du système.
- Chargement des tables de caractères du clavier.
- Activation des partitions d'échange SWAP.
- Définition du nom d'hôte.
- Contrôle et montage du système de fichiers racine (en lecture-écriture cette fois).
- Ajout des périphériques RAID et/ou LVM. Ceci peut déjà être mis en place lors du chargement de inittab.
- Activation des quotas de disque.
- Contrôle et montage des autres systèmes de fichiers.
- Nettoyage des verrous (stale locks) et des fichiers PID en cas d'arrêt brusque.

Il est possible avec certaines distributions de passer en mode interactif. Au début du boot, après le lancement d'init, il peut vous être demandé de taper sur la lettre i puis de répondre par oui ou par non aux différentes actions.

6. Niveaux d'exécution System V

a. rc

Le script **/etc/init.d/rc** prend comme paramètre le niveau d'exécution par défaut selon la ligne **initdefault** de **/etc/inittab** ou celui spécifié lors de l'appel manuel des commandes **init** ou **telinit**. Le script rc initialise le niveau d'exécution voulu et est responsable du démarrage et de l'arrêt des services associés quand le niveau d'exécution change.

```
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
```

Les services sont analysés à chaque niveau d'exécution. Lors du passage d'un niveau à un autre, et quel que soit l'ordre (du 2 au 5, du 5 au 3, etc.) le script rc compare les services qui doivent être arrêtés ou démarrés entre l'ancien et le nouveau niveau. Si un service est commun aux deux niveaux, il est maintenu. Si un nouveau service doit être lancé dans le nouveau niveau, il le lance. Si un service doit être arrêté car il est absent du nouveau niveau, il l'arrête.

7. Gestion des niveaux et des services

a. Services dans init.d

Le niveau d'exécution définit les services à démarrer pour ce niveau. C'est le script rc qui charge les services. Les services sont contrôlés (démarrage, arrêt, relance, status, etc.) à l'aide de scripts présents dans **/etc/init.d**.

Pour chaque niveau d'exécution *n*, il existe un répertoire rc*n*.d qui contient des liens symboliques (raccourcis) vers les services présents dans **/etc/init.d** à lancer ou arrêter. Ce répertoire peut être à différents endroits selon la distribution : **/etc/rc.d/rcn.d** avec des liens sur **/etc/rcn.d**

Le préfixe du nom de chaque lien définit son ordre de lancement ou son ordre d'arrêt. Le nom est sous la forme suivante :

```
[SK]nnservice
```

- **S** : start.
- **K** : kill (stop).
- **nn** : ordre numérique de démarrage ou d'arrêt. (00=premier, 99=dernier).
- **service** : nom du service.

Par exemple le lien S10network indique que le service network, responsable de la mise en place du réseau, sera démarré en ordre 10, après les S01, S05, etc. mais avant les S11, S15, S20, etc.

```
# ls -1 S* S00microcode_ctl S01sysstat S02lvm2-monitor
```

b. Contrôle manuel des services

Via le script

Les services peuvent être lancés dans tous les cas individuellement, ou à l'aide d'outils selon la distribution. Chaque service présent dans /etc/init.d accepte au moins deux paramètres :

- **start** : le service démarre.
- **stop** : le service s'arrête.

Si vous souhaitez démarrer et arrêter le service sshd (serveur ssh) à la main :

```
# /etc/init.d/sshd start
Starting SSH daemon                                                    done
# /etc/init.d/sshd stop
Shutting down SSH daemon                                               done
```

Certains services peuvent accepter d'autres paramètres :

```
# /etc/init.d/sshd
Usage: /etc/init.d/sshd {start|stop|status|try- restart|restart|force-reload|reload|probe}
```

- **status** : fournit l'état du service (démarré ou non). Selon les services des informations supplémentaires peuvent être fournies.
- **reload / forcereload** : indique au service de relire sa configuration (via un signal 1).
- **restart** : arrête et relance le service, quelle que soit l'issue de l'arrêt.

Via la commande service

La commande **service** est disponible sous Red Hat et openSUSE. Elle permet simplement de se passer du chemin vers le script de lancement du service et d'utiliser simplement son nom :

```
# service sshd stop
Shutting down SSH daemon                                          done
# service sshd start
Starting SSH daemon                                               done
```

c. Modification des niveaux d'exécution

La commande **chkconfig** permet d'ajouter, de supprimer, d'activer ou de désactiver des scripts, par niveau d'exécution. Cette commande est très pratique pour configurer les services parce qu'elle sait gérer tant les services System V que les services xinetd.

```
chkconfig [option] [service]
```

Voici le début du script de lancement de service sshd sous Red Hat. Le script démarre et s'arrête dans les niveaux 2, 3, 4 et 5. Il démarre en position 55 (S55sshd) et s'arrête en position 25 (K25sshd).

```
# chkconfig: 2345 55 25
# description: OpenSSH server daemon
```

Voici la liste des options de chkconfig :

- **--list** : liste de l'ensemble de la configuration.
- **--list** service : la configuration d'un service donné.
- **--add** service : ajoute le service indiqué dans la configuration System V.
- **--del** service : supprime le service de la configuration System V.
- **--level xxx** service **on/off** : active ou désactive le service pour les niveaux d'exécution indiqués.

8. Consoles virtuelles

Les consoles virtuelles permettent d'obtenir des terminaux virtuels sur une machine. Elles sont définies dans **/etc/inittab**. Elles sont disponibles via les périphériques **/dev/ttyn** où n est le numéro de console.

Les consoles sont lancées par inittab et par les processus **getty** ou **mingetty**. Ce sont les seules entrées de inittab ou le label a son importance : il correspond au numéro de la console. Notez l'utilisation de respawn. Lorsque le shell se termine, un processus mingetty est automatiquement relancé pour accepter une nouvelle connexion.

```
1:2345:respawn:/sbin/mingetty tty1
...
6:2345:respawn:/sbin/mingetty tty6
```

9. Arrêt

Plusieurs méthodes permettent d'arrêter proprement une machine sous Linux. Tout d'abord les arrêts sont aussi gérés par init avec les niveaux 0 et 6.
C'est ainsi que la commande suivante éteint l'ordinateur : `# init 0`
Et que celle-ci le reboote : `# init 6`

Cependant la commande la plus correcte, la plus propre et la plus sécuritaire pour arrêter le système est **shutdown**. Shutdown appelle init, mais accepte des paramètres supplémentaires. Sa syntaxe base est `shutdown <param> <délai> <message>`

Les paramètres sont :

Paramètre	Action
-k	N'effectue pas le shutdown mais envoie le message à tout le monde.
-r	C'est un reboot.
-h	(halt) c'est un arrêt.
-f	Empêche l'exécution de fsck au boot.
-F	Force l'exécution de fsck au boot.
-c	Annule le shutdown sans délai, mais un message est possible.

Le délai peut être spécifié de différentes manières :

- **hh:mm** : une heure précise.
- **+m** : dans m minutes.
- **now** : un alias pour +0, c'est-à-dire tout de suite.

L'exemple suivant programme un reboot pour dans 10 minutes avec un message d'avertissement.

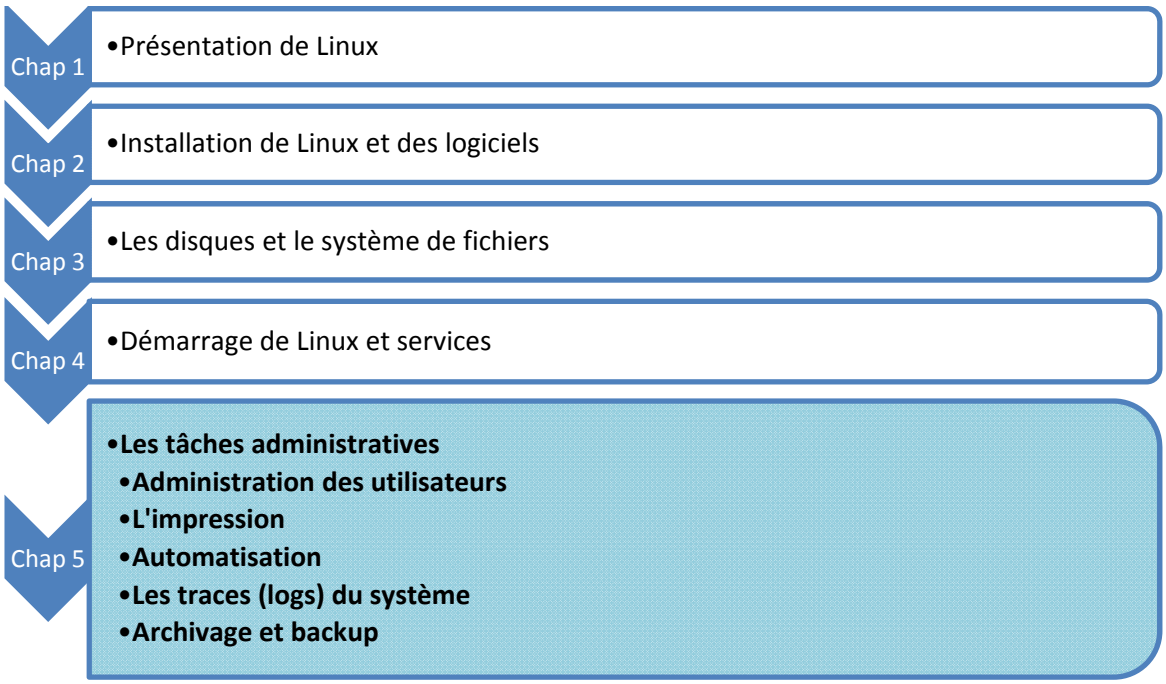
```
# shutdown -r +10 "Reboot pour maintenance dans 10 minutes"
```

L'exemple suivant annule le reboot.

```
# shutdown -c "Maintenance annulée"
```

Les commandes **reboot** et **halt** sont appelées en fin d'init 6 et 0, respectivement. Si elles sont appelées dans un autre niveau que 6 ou 0, elles sont l'équivalent d'un appel à shutdown :

- **halt** : shutdown -h
- **reboot** : shutdown -r


Chap 1
•Présentation de Linux
Chap 2
•Installation de Linux et des logiciels
Chap 3
•Les disques et le système de fichiers
Chap 4
•Démarrage de Linux et services
Chap 5
•Les tâches administratives
•Administration des utilisateurs
•L'impression
•Automatisation
•Les traces (logs) du système
•Archivage et backup

Administration des utilisateurs

1. Principe

a. Identification et authentification

L'**identification**, c'est savoir qui est connecté, afin de déterminer les droits de la personne qui se connecte. Un utilisateur est identifié par un login.

L'**authentification**, c'est apporter la preuve de qui on est, par exemple via un secret partagé entre l'utilisateur et le système, et connus d'eux seuls. L'utilisateur est authentifié par un mot de passe.

b. Les utilisateurs

Un utilisateur est l'association d'un nom de connexion, le login, à un UID et au moins un GID.

- **UID** : User ID.
- **GID** : Group ID.

Les UID et les GID sont en principe uniques. Le login est unique.

L'UID identifie l'utilisateur (ou le compte applicatif) tout au long de sa connexion. Il est utilisé pour le contrôle de ses droits et de ceux des processus qu'il a lancé. Ce sont les UID et GID qui sont stockés au sein de la table des inodes, dans la table des processus, etc., et non les logins.

L'utilisateur dispose des attributs de base suivants :

- un nom de connexion appelé le login ;
- un mot de passe ;
- un UID ;
- un GID correspondant à son groupe principal ;
- un descriptif ;
- un répertoire de connexion ;
- une commande de connexion ;

Les UID d'une valeur inférieure à 100 sont en principe associés à des comptes spéciaux avec des droits étendus. Ainsi l'UID de root, l'administrateur, est 0. Selon les distributions, à partir de 100, 500 ou 1000, et ce jusqu'à environ 60000, ce sont les UID des utilisateurs sans pouvoirs particuliers.

Un login a en principe une taille de 8 caractères. En fait Linux et d'autres systèmes acceptent une taille plus grande, mais avec la plupart des commandes l'affichage, voire la gestion des logins, est limité à 8 caractères.

Un login accepte la plupart des caractères. Il ne doit pas commencer par un chiffre. Il est possible de modifier la liste des caractères autorisés et de forcer la longueur et la complexité via les mécanismes d'authentification PAM et le fichier **`/etc/login.defs`**.

c. Les groupes

Chaque utilisateur fait partie d'au moins un groupe. Un groupe regroupe des utilisateurs. Comme pour les logins, le GID du groupe accompagne toujours l'utilisateur pour le contrôle de ses droits. Un utilisateur peut faire partie de plusieurs groupes, auquel cas il faut distinguer son groupe primaire des groupes secondaires.

Les groupes sont aussi des numéros. Il existe des groupes spécifiques pour la gestion de certaines propriétés du système et notamment l'accès à certains périphériques.

Le groupe primaire est toujours appliqué à la création d'un fichier. Si l'utilisateur elies a pour groupe primaire users, alors les fichiers créés par elies auront comme groupe d'appartenance users.

Un utilisateur dispose de tous les droits associés à ses groupes secondaires. Si elies a comme groupe secondaire video et qu'un fichier dispose des droits d'écriture pour ce groupe, alors elies aura le droit de modifier son contenu.

La commande **id** donne les informations sur un utilisateur : uid, gid, groupes secondaires.

```
$ id seb
uid=1000(seb) gid=100(users) groupes=100(users),16(dialout),3(sys),
33(video)
```

Un fichier créé par elies. est propriété de elies et son groupe est le groupe principal de elies : users.

```
$ touch test
$ ls -l test
-rw-r--r-- 1 elies users 0 avr 10 14:30 test
```

d. Les mots de passe

Les mots de passe permettent d'authentifier les utilisateurs. Ils doivent être assez complexes pour ne pas être découverts facilement, mais assez intuitifs pour qu'ils s'en souviennent. Les mots de passe sont cryptés (MD5, DES par exemple) et ne sont pas directement lisibles sous leur forme cryptée par l'utilisateur afin que personne ne puisse tenter de le décrypter via un quelconque traitement.

Un utilisateur devrait changer régulièrement son mot de passe, ne jamais l'écrire quelque part ni le conserver sur lui.

2. Les fichiers

a. /etc/passwd

Le fichier **/etc/passwd** contient la liste des utilisateurs du système local. Il est lisible par tout le monde. Les informations qu'il contient sont publiques et utiles tant pour le système que pour les utilisateurs. Chaque ligne représente un utilisateur et est composée de sept champs.

```
Login:password:UID:GID:comment:homedir:shell
```

- Champ 1 : le login ou nom d'utilisateur.
- Champ 2 : sur les vieilles versions, le mot de passe crypté. Si un x est présent, le mot de passe est placé dans /etc/shadow. Si c'est un point d'exclamation le compte est verrouillé.
- Champ 3 : le User ID.
- Champ 4 : le GID, c'est-à-dire le groupe principal.
- Champ 5 : un commentaire ou descriptif. C'est un champ d'information.
- Champ 6 : le répertoire de travail, personnel, de l'utilisateur. C'est le répertoire dans lequel il arrive lorsqu'il se connecte.
- Champ 7 : le shell par défaut de l'utilisateur. Mais ce peut être toute autre commande, y compris une commande interdisant la connexion.

b. /etc/group

Le fichier **/etc/group** contient la définition des groupes d'utilisateurs et pour chacun la liste des utilisateurs dont il est le groupe secondaire. Chaque ligne est composée de quatre champs :

```
Group:password:GID:user1,user2,...
```

- Champ 1 : le nom du groupe.
- Champ 2 : le mot de passe associé. Voyez l'explication juste en dessous.
- Champ 3 : le Group Id.
- Champ 4 : la liste des utilisateurs appartenant à ce groupe.

Il est inutile de replacer dans le quatrième champ les utilisateurs ayant ce groupe pour groupe principal, c'est induit. Le champ de mot de passe pour les groupes est très rarement utilisé dans la pratique. Un utilisateur a le droit de changer de groupe afin de prendre, temporairement un groupe secondaire comme groupe principal avec la commande **newgrp**.

Dans ce cas, l'administrateur peut mettre en place un mot de passe sur le groupe pour protéger l'accès à ce groupe en tant que groupe principal.

c. /etc/shadow

Le fichier **/etc/shadow** accompagne le fichier **/etc/passwd**. C'est là qu'est stocké, entre autres, le mot de passe crypté des utilisateurs. Il contient toutes les informations sur le mot de passe et sa validité dans le temps. Chaque ligne est composée de 9 champs séparés par des ':'

```
elies:$2a$10$AjADxPEfE5iUJcltzYA4wOZO.f2UZ0qP/8EnOFY.P.m10HifS7J8i:13913:0:99999:7:::
```

- Champ 1 : le login.
- Champ 2 : le mot de passé crypté. Le xx initial indique le type de cryptage.
- Champ 3 : nombre de jours depuis le 1er janvier 1970 du dernier changement de mot de passe.
- Champ 4 : nombre de jours avant lesquels le mot de passe ne peut pas être changé (0 : il peut être changé n'importe quand).
- Champ 5 : nombre de jours après lesquels le mot de passe doit être changé.
- Champ 6 : nombre de jours avant l'expiration du mot de passe durant lesquels l'utilisateur doit être prévenu.
- Champ 7 : nombre de jours après l'expiration du mot de passe après lesquels le compte est désactivé.
- Champs 8 : nombre de jours depuis le 1er janvier 1970 à partir du moment où le compte a été désactivé.
- Champ 9 : réservé.

Dans l'exemple de la ligne elies, le mot de passe a été changé 13913 jours après le 01/01/1970. Le mot de passe doit être changé avant 0 jours mais il est toujours valide car le champ suivant indique qu'il faut le changer au bout de 99999 jours (273 ans) et le champ 5 est vide (pas d'obligation de changement de mot de passe). Le compte est désactivé après 7 jours.
Pour connaître la date en fonction du 01/01/1970 utilisez la commande :

```
$ date --date "1 jan 1970 +13984 days"
mar avr 15 00:00:00 CEST 2008
```

3. Gestion des utilisateurs

a. Ajout

La création d'un utilisateur pourrait être entièrement effectuée à la main.
La création d'un utilisateur consiste à :

- rajouter une ligne dans **/etc/passwd**,
- rajouter d'une ligne dans **/etc/shadow**,
- rajouter d'éventuelles informations dans **/etc/group**,
- créer le répertoire personnel et mettre à jour son contenu avec **/etc/skel**,
- changer les permissions et le propriétaire du répertoire personnel,
- changer le mot de passe (encodé).

Vous pouvez créer directement un compte en éditant les fichiers avec un éditeur, utilisez la commande **vipw** qui va mettre à jour les divers caches associés à la gestion des comptes.

La commande **vipw** admet trois arguments :

- -p : édition de **/etc/passwd**.
- -g : édition de **/etc/group**.
- -s : édition de **/etc/shadow**.

Tout ceci peut être effectué avec la commande **useradd**. Elle ajoute un nouveau compte et effectue les principales opérations :

- création de l'utilisateur et remplissage des fichiers,
- création d'un groupe privé d'utilisateur (de même nom que celui-ci),
- création du répertoire personnel, remplissage et modification des droits.

```
useradd <options> login
```

Si aucune option n'est précisée, les valeurs par défaut sont récupérées au sein du fichier **/etc/defaults/useradd**. Les options principales suivantes sont acceptées :

Option	Rôle
-m	Crée aussi le répertoire personnel. Elle est parfois comprise par défaut, mais il vaut mieux vérifier si le répertoire personnel est présent après l'utilisation de la commande si vous n'utilisez pas cette option.
-u	Précise l'UID numérique de l'utilisateur, pour le forcer. Autrement l'UID est calculé selon les règles du fichier login.defs et les UID existants.

-g	Précise le groupe principal de l'utilisateur, par GID ou par son nom (variable GROUP).
-G	Précise les groupes additionnels (secondaires, de l'utilisateur) séparés par des virgules (variable GROUPS).
-d	Chemin du répertoire personnel. Généralement /home/<login>, mais n'importe quel chemin peut être précisé (variable HOME/<login>).
-c	Un commentaire associé au compte. Il peut être quelconque mais est parfois utilisé par certaines commandes comme **finger**. Son contenu peut être modifié par l'utilisateur avec la commande **chfn**.
-k	Chemin du répertoire contenant le squelette de l'arborescence du répertoire utilisateur. C'est généralement /etc/skell (variable SKEL).
-s	Shell (commande de connexion) par défaut de l'utilisateur (variable SHELL). L'utilisateur peut le changer via la commande **chsh**.

b. Sécurité des mots de passe

Changer de mot de passe

La commande **passwd** permet de gérer les mots de passe mais aussi les autorisations de connexion et la plupart des champs présents dans **/etc/shadow**.

Tout utilisateur a le droit de changer son mot de passe, dans le délai précisé par le champ 4 de /etc/shadow. L'action par défaut est de changer le mot de passe de l'utilisateur courant. L'ancien mot de passe est demandé par sécurité. La saisie est masquée.
L'utilisateur root a le droit de modifier les mots de passe de tous les utilisateurs du système, sans avoir à connaître le précédent mot de passe.

```
# passwd elies
Changing password for elies. Nouveau mot de passe :
Mot de passe incorrect : basé sur un mot du dictionnaire
Retaper le nouveau mot de passe : Mot de passe changé.
```

Gérer les informations de validité

Tous les champs de **/etc/shadow** peuvent être modifiés par la commande **passwd**. Voici quelques options disponibles.

Option	Rôle
-l	Lock : verrouille le compte en rajoutant un ! devant le mot de passe crypté.
-u	Unlock : déverrouille le compte. Il n'est pas possible de déverrouiller un compte qui n'a pas de mot de passe, il faut utiliser en plus f pour cela.
-d	(root) Supprime le mot de passe du compte.
-n <j>	(root) Durée de vie minimale en jours du mot de passe.
-x <j>	(root) Durée de vie maximale en jours du mot de passe.
-w <j>	(root) Nombre de jours avant avertissement.
-i <j>	(root) Délai de grâce avant désactivation si le mot de passe est expiré.
-S	(root) Statut du compte.

Dans l'exemple suivant le compte elies est modifié comme ceci :

- Il doit attendre 5 jours après saisie d'un nouveau mot de passe pour pouvoir le changer,
- Son mot de passe est valide 45 jours,
- Il est prévenu 7 jours avant qu'il doit changer de mot de passe,
- S'il ne change pas de mot de passe après 45 jours, il aura encore 5 jours avant d'être désactivé.

```
# passwd -n 5 -x 45 -w 7 -i 5 elies
```

Voici la ligne de **/etc/shadow** associée.

```
elies:$2a$10$dwbUGrC75bs3l52V5DHxZefkZyB6VTHsLH5ndjsNe/vF/HAzHOcR2:13984:5:45:7:5::
```

La commande **chage** permet de faire à peu près la même chose. Elle n'est accessible que par root. Lancée avec argument le login de l'utilisateur, elle est interactive. Notez la possibilité de modifier la date du dernier changement du mot de passe et une date fixe d'expiration du mot de passe (champ 8) :

```
# chage elies
Changing aging information for elies.
Minimum Password Age [7]:
Maximum Password Age [40]:
Password Expiration Warning [10]:
Password Inactive [5]:
Last Password Change (YYYY-MM-DD) [2008-04-10]:
Account Expiration Date (YYYY-MM-DD) [1969-12-31]: 2010-01-01
Aging information changed.
```

Voici la ligne /etc/shadow résultante :

```
elies:$2a$10$dwbUGrC75bs3l52V5DHxZefkZyB6VTHsLH5ndjsNe/vF/HAzHOcR2:13979:7:40:10:5:14610:
```

Les paramètres suivants sont acceptés :

Option	Rôle
-m	Mindays : équivaut à passwd -n.
-M	Maxdays : équivaut à passwd -x.
-d	Date de dernière modification du mot de passe (depuis le 01/01/1970).
-E	Date d'expiration du mot de passe (depuis le 01/01/1970).
-I	Inactive : équivaut à passwd -i.
-W	Warndays : équivaut à passwd -w.
-l	List : affiche tous les détails.

Les détails sont bien plus lisibles avec chage qu'avec passwd :

```
# passwd -S elies
elies PS 04/10/2008 7 40 10 5
# chage -l elies
Minimum:          7
Maximum:          40
Warning:          10
Inactive:         5
Last Change:              avr 10, 2008
Password Expires:         mai 20, 2008
Password Inactive:        mai 25, 2008
Account Expires:          jan 01, 2010
```

c. Modification

Utilisez la commande **usermod** pour modifier un compte. Elle prend la même syntaxe et options que useradd mais dispose aussi d'une syntaxe complémentaire.

Option	Rôle
-L	Lock du compte, comme passwd -l.
-U	Unlock du compte, comme passwd -u.
-e <n>	Expire : le mot de passe expire n jours après le 01/01/1970.
-u <UID>	Modifie l'UID associé au login. Le propriétaire des fichiers appartenant à l'ancien UID au sein du répertoire personnel est modifié en conséquence.
-l <login>	Modifie le nom de login.
-m	Move : implique la présence de -d pour préciser un nouveau répertoire personnel. Le contenu de l'ancien répertoire est déplacé dans le nouveau.

d. Suppression

Supprimez un utilisateur avec la commande **userdel**. Par défaut le répertoire personnel n'est pas supprimé. Vous devez pour ceci passer l'option -r.

```
# userdel -r bean
```

4. Gestion des groupes

a. Ajout

Vous pouvez créer un groupe directement dans le fichier **/etc/group** ou bien passer par les commandes associées. Si vous éditez le fichier à la main, utilisez la commande **vigr** (ou vipw g).

La commande **groupadd** permet de créer un groupe. Sa syntaxe accepte l'argument -g pour un GID.

```
# grep iset /etc/group iset:!:1234:
```

b. Modification

La commande **groupmod** permet de modifier un groupe. Ses paramètres sont les suivants :

Option	Rôle
-n <nom>	Renomme le groupe.
-g <GID>	Modifie le GID. Attention, le groupe d'appartenance des fichiers concernés n'est pas modifié.
-A <user>	Ajoute l'utilisateur spécifié dans le groupe (groupe secondaire).
-R <user>	Supprime l'utilisateur spécifié du groupe.

c. Suppression

La commande **groupdel** supprime un groupe. La commande vérifie d'abord si le groupe que vous voulez supprimer est le groupe principal d'un utilisateur, alors le groupe ne peut pas être supprimé.

Par contre aucune action autre que celle consistant à supprimer la ligne correspondant dans /etc/group n'est effectuée : c'est à vous de vérifier le système de fichiers (et la configuration des applications si besoin) pour supprimer toute trace de ce groupe.

```
# groupdel iset
```

d. Vérifier la cohérence

Il peut être utile de lancer des outils de vérification de la cohérence des fichiers des groupes et des mots de passe. Si vous avez l'habitude de modifier ces fichiers à la main, rien ne garantit que tout est correct. La commande .**pwck** vérifie les fichiers **/etc/passwd** et **/etc/shadow** et reporte les erreurs.
La commande **grpck** fait la même chose pour les groupes.

c. Vérifier les connexions

Vous pouvez tracer les connexions sur votre machine à l'aide de deux commandes. La commande **lastlog** se base sur le contenu de /var/log/lastlog. Elle accepte les paramètres -u (précision d'un utilisateur) et -t pour rechercher les connexions des n derniers jours.
La commande **last** fait à peu près la même chose, mais se base sur /var/log/wtmp qui fournit des informations supplémentaires comme l'origine de la connexion (IP, nom de la console, etc.) et les dates de connexion et de déconnexion, ainsi que la durée de connexion et si l'utilisateur est encore connecté.

5. Configuration avancée

a. /etc/default/useradd

Le fichier **/etc/default/useradd** contient un certain nombre de variables définissant les règles par défaut à appliquer à la création d'un utilisateur :

- son groupe,
- la racine de son répertoire personnel (là où celui-ci sera situé),
- s'il est actif ou non,
- le shell,
- son ou ses groupes secondaires,
- l'endroit où est situé le squelette des comptes (structure de base d'un répertoire utilisateur),
- la création ou non d'un spool (dépôt) de courrier,
- etc.

b. /etc/default/passwd

Le fichier **/etc/default/passwd** contient quelques règles utilisées par la commande **passwd** pour le cryptage des mots de passe.

c. /etc/default/su

Le fichier **/etc/default/su** permet de configurer le fonctionnement de la commande **su**. Par défaut **su** avec le paramètre **-** met en place un nouveau PATH car il charge l'environnement de l'utilisateur ciblé. Vous pouvez modifier ceci et mettre en place votre propre PATH, ou conserver l'ancien.

d. /etc/login.defs

Le fichier **/etc/login.defs** est utilisé par de nombreuses commandes comme **login**, **useradd**, **groupadd**, **passwd** pour définir quelques valeurs par défaut et la validité des logins. Son contenu peut varier suivant les distributions. Il peut contenir :

- une règle de validité des comptes (caractères autorisés, longueur, etc.),
- les UID min et max lors de la création d'un utilisateur,
- les GID min et max lors de la création d'un groupe,
- les commandes à appeler pour les ajouts/modifications/créations d'utilisateur,
- les règles par défaut pour la validité des mots de passe,
- la création ou non d'un répertoire personnel,
- etc.

6. Notifications à l'utilisateur

a. /etc/issue

Lorsqu'un utilisateur se connecte depuis la console, un message est généralement affiché juste avant l'invite de saisie de son login. Ce message est contenu dans le fichier **/etc/issue**. C'est un message d'accueil. Par défaut, il contient le nom de la distribution Linux et la version du noyau.

b. /etc/issue.net

Le message d'accueil peut être différent lorsqu'un utilisateur se connecte depuis une console distante (telnet, ssh, etc.). C'est souvent le même mais sans les caractères de contrôles liés à un shell donné.

c. /etc/motd

Motd signifie Message of the day, le message du jour. Une fois l'utilisateur connecté depuis une console (locale ou distante), un message peut être affiché. Par défaut il est vide. Vous pouvez par exemple modifier ce fichier pour prévenir vos utilisateurs d'une perturbation sur le réseau pour maintenance aura lieu tel jour à telle heure.

7. L'environnement utilisateur

a. /etc/skel

À la création d'un utilisateur et de son répertoire personnel, l'environnement de l'utilisateur est mis en place. L'environnement contient les variables d'environnement, les alias, l'exécution de divers scripts. Il est contenu dans des fichiers chargés au démarrage de l'interpréteur de commandes (shell).

Lors de la création d'un compte, les divers fichiers de configuration sont copiés depuis le contenu du répertoire /etc/skel (skeleton) vers le répertoire personnel. Si vous souhaitez modifier les environnements de façon globale AVANT la création des utilisateurs, vous pouvez placer dans /etc/skel tous les fichiers que vous souhaitez et les modifier selon votre convenance.

b. Scripts de configuration

À la connexion d'un utilisateur, les scripts suivants sont exécutés dans cet ordre :

- **/etc/profile** : définit les variables d'environnement importantes comme PATH, LOGNAME, USER, HOSTNAME, HISTSIZE, MAIL et INPUTRC.
- **/etc/profile.d/*** : /etc/profile appelle tous les scripts présents dans ce répertoire. Ces scripts peuvent compléter la configuration globale en ajoutant par exemple la configuration des paramètres linguistiques, des alias globaux, etc.
- **~/.bash_profile** : si le shell est bash, c'est le script suivant à être exécuté. Il est dans le répertoire utilisateur et appelle un autre script : **~/.bashrc** qui appelle luimême **/etc/bashrc**. Vous pouvez définir dans .bash_profile des variables supplémentaires, alors que vous aurez tendance à définir dans **~/.bashrc** des alias et des fonctions. Il n'y a pas de règles strictes.
- **/etc/bashrc** est utilisé pour définir les fonctions et alias pour tout le système et tous les utilisateurs sous bash.

L'impression

1. Principe

Il existe trois standards d'impression sous Unix, un sous System V, un autre sous BSD et un dernier fédérateur CUPS.

Quel que soit le standard, le principe est le même. À chaque imprimante déclarée (généralement dans /etc/printcap) correspond une file d'attente (**queue**). L'ensemble de ces files d'attente est géré par un service indépendant. Ces deux principes permettent une impression multiutilisateur (les travaux d'impression sont en file d'attente, **job queues**), et en réseau (le service peut être utilisé depuis une autre machine distante).

En règle générale toutes les imprimantes savent directement imprimer du texte brut ASCII en 80 colonnes. Pour imprimer des documents formatés ou des images, on peut utiliser un pilote. On parle en fait de **filtre d'impression**. Le filtre peut être un script ou un binaire qui récupère le flux entrant (texte, image, document, postscript...), l'identifie et à l'aide de traitements associés le transforme en langage compréhensible par l'imprimante (Postscript, PCL, Canon, Epson, WPS...).
Linux accepte les commandes issues des Unix de type System V et BSD. Pendant longtemps le sous-système d'impression était basé sur les services BSD et le démon **lpd**. Depuis quelques années, toutes les distributions se basent sur CUPS, rétro-compatible avec les anciens systèmes d'impression.

2. System V

Les commandes de gestion des files d'attente et des impressions sous System V sont les suivantes :

- `lp [-dImprimante] [-nChiffre] fic1` : imprime le contenu du fichier fic1. L'option `-d` permet de choisir l'imprimante, `-n` le nombre d'exemplaires.
- `lpstat [-d] [-s] [-t] [-p]` : informations sur l'impression. L'option `-d` affiche le nom de l'imprimante par défaut, `-s` un état résumé des imprimantes, `-t` la totalité des informations (statut, travaux...), `-p [liste]` uniquement les informations sur les imprimantes incluses dans la liste.
- `cancel [ids] [printers] [-a] [-e]` : supprime les tâches d'impression ids des imprimantes printers. L'option `-a` supprime tous les travaux de l'utilisateur, `-e` tous les travaux (seulement pour l'administrateur).
- On peut trouver les commandes **enable** et **disable** qui prennent comme paramètre le nom de la file d'attente, permettant d'en activer ou désactiver l'accès.

Le démon (ou daemon) s'appelle généralement **lpd** (*line printer daemon*).

- **lpadmin** permet d'administrer les services d'impression comme les files d'attentes associées à une imprimante et la file d'attente par défaut. Ex : `lpadmin -p queue1 -v imprimante-m modèle`.
- `lpadmin -x file` : suppression de la file d'attente.
- `lpadmin -d file` : définir la file d'attente par défaut.
- `lpadmin -p file -u allow:liste` : autorisation d'imprimer pour les utilisateurs précisés.
- `lpadmin -p file -u deny:liste` : interdiction d'imprimer pour les utilisateurs précisés.
- `lpshut` arrête le service d'impression. Au redémarrage du démon, les impressions en cours au moment de l'arrêt sont reprises.
- accept et reject permettent de valider une file d'attente pour l'impression ou de la fermer.
- `lpmove` permet de transférer des requêtes d'impression d'une file d'attente vers une autre.

3. BSD

- `lpr [-Pimprimante] [-#copies] fic1` : imprime le contenu du fichier fic1. L'option `-P` permet de spécifier l'imprimante, `-#` le nombre de copies.
- `lpq [-Pimprimante]` : indique l'état et la liste des travaux pour l'imprimante éventuellement spécifiée par l'option `-P`.
- `lprm [-Pimprimante] [-] [ids]` : permet de supprimer un travail de l'imprimante spécifiée par l'option `-P`, l'option - supprime tous les travaux de l'utilisateur, ids représente une liste de travaux à supprimer.
- La commande **lpc** est une sorte de petit shell permettant de contrôler les imprimantes et les travaux. Le service s'appelle généralement **lpd**.

4. CUPS

a. Présentation

CUPS (*Common Unix Printing System*) est un système d'impression Unix, orienté réseau :

- Basé sur le protocole **IPP** (*Internet Printing Protocol*) basé luimême sur le protocole **HTTP/1.1**.
- Simple d'utilisation, notamment grâce à une configuration et une administration centralisée depuis une interface HTTP, des règles de conversion basées sur les types MIME, et des fichiers de description d'imprimante standards (**PPD**, *PostScript Printer Description*).
- CUPS reprend les commandes System V et BSD déjà abordées pour plus de simplicité.
- Les traces des impressions sont disponibles au format **CLF** (*Common Log Format*) de serveur Web et exploitables par les mêmes outils.
- CUPS est capable d'interagir avec les serveurs d'impression LPD pour garder une compatibilité ascendante.
- CUPS dispose de sa propre API permettant de créer des interfaces utilisateur pouvant s'intégrer dans des environnement graphiques ou des interfaces d'administration.
- Les pools d'impression permettent la redirection automatique des tâches.
- L'authentification est possible par utilisateur, hôte ou certificat numérique. Le service d'impression se nomme **cupsd**.

```
$ ps -ef |grep cupsd
root             3128     1  0 Apr29 ?        00:00:01 /usr/sbin/cupsd
```

Il n'y a pas besoin d'outils graphiques pour administrer un serveur CUPS bien que pour des raisons de facilité la plupart des distributions, voire des environnements graphiques, en proposent. CUPS propose une interface d'administration WEB directement accessible depuis le port 631 du serveur. L'interface fonctionne avec n'importe quel navigateur HTTP.

```
http://localhost:631
```

Le fichier de configuration est **/etc/cups/cupsd.conf**. L'équivalent du fichier **/etc/printcap** est présent dans **/etc/cups/printers.conf**.

```
<DefaultPrinter lj2100> Info Laserjet 2100
DeviceURI socket://192.168.1.10:9100
State Idle
StateTime 1203806079
Accepting Yes
Shared Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
OpPolicy default
ErrorPolicy stop-printer
</Printer>
```

b. Ajout d'une imprimante

Vous avez deux solutions pour ajouter une imprimante :

- Éditer les fichiers à la main.
- Passer par l'interface Web ou un outil de votre distribution.

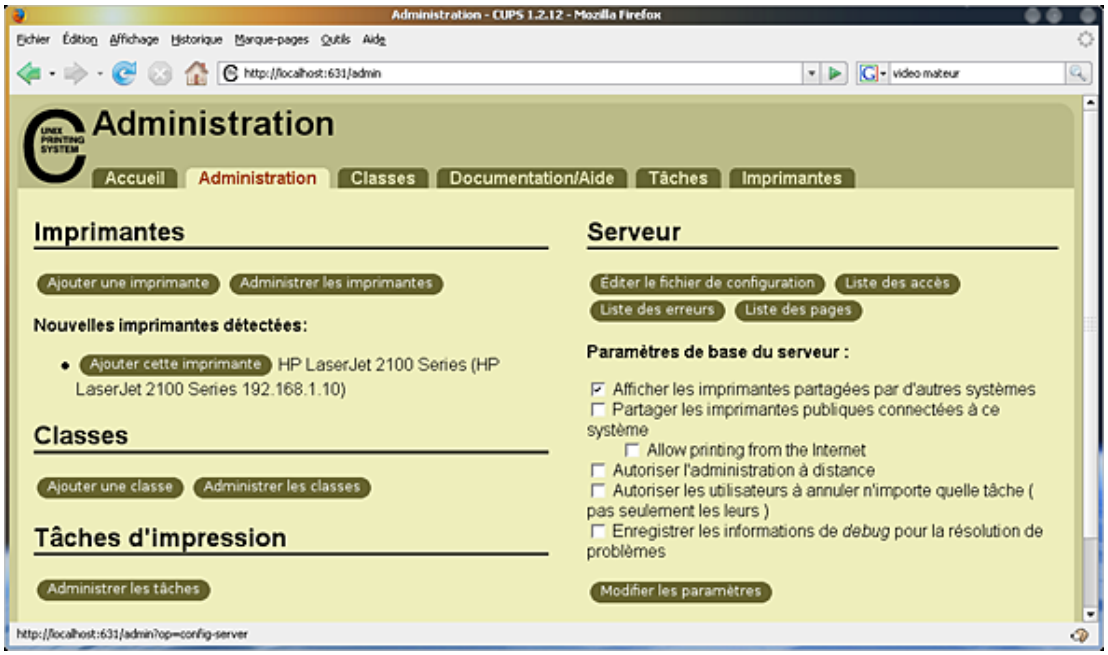
Dans le premier cas, vous devez modifier le fichier printers.conf pour ajouter une section pour votre imprimante puis copier dans /etc/cups/ppd le fichier PPD correspondant à votre imprimante et le renommer en utilisant le nom de section (ex : lj2100.ppd) du fichier printers.conf. Enfin vous devez rechercher la configuration de CUPS. Sous Red Hat ou openSUSE par exemple :

```
# service cups reload
```

Comme l'interface Web est généralement activée par défaut, vous pouvez passer par un navigateur Web. Il est possible que lors de l'accès aux pages d'administration des identifiants vous soient demandés. En principe ceux de root suffisent mais vous pouvez créer des comptes (ou rajouter des utilisateurs) chargés de la gestion des impressions avec la commande **lppasswd**.

```
# lppasswd -a elies
Enter password:
Enter password again:
```

- Sur la page principale cliquez sur l'onglet **Administration**.



Accueil du frontend Web de CUPS

Automatisation

1. Avec cron

a. Présentation

Le service **cron** permet la programmation d'événements à répétition. Il fonctionne à l'aide d'une table, appelée une **crontab**. C'est un fichier texte, éditable avec un simple éditeur, par exemple vi. Pour modifier votre crontab personnelle utilisez la commande **crontab** pour éditer la table, avec le paramètre -e.

Les fichiers crontabs sont sauvés dans /var/spool/cron.

Le service **cron** doit tourner pour que les crontabs soient actives.

b. Formalisme

Le format d'un enregistrement de crontab est le suivant :

Minutes	Heures	Jour du mois	Mois	Jour semaine	Commande
1	2	3	4	5	6

Utilisez le format suivant pour les valeurs périodiques :

- Une valeur pour indiquer quand il faut exécuter la commande. Ex : la valeur 15 dans le champ minute signifie la quinzième minute.
- Une liste de valeurs séparées par des virgules. Ex : 1,4,7,10 dans le champ mois pour janvier, avril, juillet, octobre.
- Un intervalle de valeurs. Ex : 15 dans le champ jour de la semaine indique du lundi (1) au vendredi (5). Le 0 est le dimanche et le 6 le samedi.
- Le caractère * pour toutes les valeurs possibles. Ex : * dans le champ jour du mois indique tous les jours du ou des mois.

c. Exemples

Exécution de df tous les jours, toute l'année, tous les quarts d'heure :

```
0,15,30,45 * * * * df > /tmp/libre
```

Exécution d'une commande tous les jours ouvrables à 17 heures :

```
0 17 * * 1-5 fin_travail.sh
```

Lister les crontabs actives :

```
$ crontab -l
```

Supprimer la crontab active :

```
$ crontab -r
```

Éditer la crontab d'un utilisateur particulier :

```
# crontab -u user
```

d. crontab système

La configuration crontab générale pour le système est dans /etc/crontab.

```
SHELL=/bin/bash PATH=/sbin:/bin:/usr/sbin:/usr/bin MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Ici tous les jours à 4h02 du matin **run-parts /etc/cron.daily** est exécuté. Le script **run-parts** accepte en paramètre un répertoire et exécute tous les programmes présents dans ce répertoire.
Enfin, le répertoire /etc/cron.d contient des crontabs supplémentaires.

e. Contrôle d'accès

Vous pouvez contrôler l'accès à la commande **crontab** par utilisateur avec les fichiers **/etc/cron.allow** et **/etc/cron.deny**.

- Si cron.allow est présent, seuls les utilisateurs qui y sont explicitement indiqués peuvent utiliser **cron**).
- Si cron.allow est absent, cron vérifie la présence d'un fichier cron.deny. Tous les utilisateurs n'y étant pas sont autorisés à utiliser cron. S'il est vide la commande **cron** est autorisée pour tout le monde.
- Si les deux fichiers sont absents, seul root peut utiliser cron.

2. Avec at

a. Présentation

La commande **at** et les commandes associées permettent une gestion des traitements batchs. Contrairement à la crontab les modifications sont volatiles : elles sont perdues lorsque la session est terminée. C'est à vous de placer la liste des commandes dans un éventuel fichier et de le charger au besoin via les scripts de votre profil. Pour que at fonctionne le service **atd** (*at daemon*) doit fonctionner.

b. Formalisme

Pour simplifier, il y a deux moyens d'utiliser at :

- en lui passant de manière interactive une ligne de commande,
- en lui passant un fichier exécutable contenant les commandes à exécuter.

Dans les deux cas, vous devez fournir à at une heure d'exécution. Le formalisme de cette heure est assez souple. Pour programmer l'exécution d'une ligne de commande à 21h20 de manière interactive :

```
$ at 21:20
warning: commands will be executed using /bin/sh at> echo salut
at> <EOT>
job 4 at 2008-05-08 21:20
```

Après avoir saisi la ou les commandes à exécuter à 21h20, appuyez sur [Entrée] et sur une ligne vide appuyez sur [Ctrl] **D** (fin de saisie). La commande **at** confirme la programmation de la commande. Pour programmer l'exécution d'une commande (script ou binaire) à 21h25 :

```
$ at -f /home/elies/test.sh 21:25
warning: commands will be executed using /bin/sh job 6 at 2008-05-08 21:25
```

Heure

L'heure peut être formatée ainsi :

- HHMM ou HH:MM.
- L'heure peut être au format 12 ou 24h. Au format 12 heures, vous pouvez préciser AM (matin) ou PM (après-midi).
- Midnight (minuit), noon (midi), teatime (16h00, typiquement anglais).
- MMJJAA, MM/JJ/AA ou JJ.MM.AA pour une date précise.
- Now : maintenant.
- + n minutes/hours/days/weeks : l'heure courante plus n minutes/heures/jours/semaines

Si l'heure précisée est inférieure à l'heure actuelle, la commande est exécutée le lendemain.

```
$ at 21:30 09.05.2008
warning: commands will be executed using /bin/sh
at> echo salut !
at> <EOT>
job 9 at 2008-05-09 21:30
$ at now + 2 days
warning: commands will be executed using /bin/sh
at> echo dans deux jours
at> <EOT>
job 10 at 2008-05-10 21:29
```

c. Contrôle des tâches

La commande **atq** (*at queue*) permet de lister les tâches programmées :

```
$ atq
10        2008-05-10 21:29 a elies
9         2008-05-09 21:30 a elies
```

Les jobs (tâches) sont placées dans le répertoire /var/spool/atjobs, à raison de un exécutable par tâche.

```
# ls -l /var/spool/atjobs/
-rwx------ 1 elies users 5620 mai  8 21:29 a000090133cf92
-rwx------ 1 elies users 5628 mai  8 21:30 a0000a0133d531
```

Si vous regardez le contenu de l'exécutable, vous voyez que votre commande n'est pas seule. Elle est située à la fin, mais le script positionne tout l'environnement lors de la création de l'entrée at.

```
#cat a0000a0133d531
#!/bin/sh
# atrun uid=1000 gid=100
# mail        elies 0
umask 22
LESSKEY=/etc/lesskey.bin; export LESSKEY NNTPSERVER=news; export NNTPSERVER
INFODIR=/usr/local/info:/usr/share/info:/usr/info; export INFODIR
MANPATH=/usr/local/man:/usr/share/man:/opt/gnome/share/man; export MANPATH
KDE_MULTIHEAD=false; export KDE_MULTIHEAD
... (environ 80 lignes) ... cd /home/elies || {
echo 'Execution directory inaccessible' >&2
exit 1
}
echo salut !
```

La commande **atrm** permet de supprimer une tâche :

```
$ atrm 10
$ atrm 9
$ atq
```

d. Contrôle d'accès

Vous pouvez contrôler l'accès à la commande **at** par utilisateur avec les fichiers **/etc/at.allow** et **/etc/at.deny**.

- Si at.allow est présent, seuls les utilisateurs qui y sont explicitement indiqués peuvent utiliser at.
- Si at.allow est absent, at vérifie la présence d'un fichier at.deny. Tous les utilisateurs n'y étant pas sont autorisés à utiliser at. S'il est vide la commande at est autorisée pour tout le monde.
- Si les deux fichiers sont absents, seul root peut utiliser at.

Les traces (logs) du système

1. Principe

Lorsque le système démarre, fonctionne et effectue tout type d'action, ses actions et celles de la plupart de ses services sont tracées dans divers fichiers. Deux services sont spécialisés dans la réception des messages à écrire dans ces fichiers :

- **klogd** : *kernel log daemon*, chargé de la gestion des informations émises par le noyau.
- **syslogd** : *system log daemon*, chargé de la gestion des informations émises par tout type de service et éventuellement le noyau.

Les messages importants émis par un composant du système devraient passer par le service syslogd. Ceci n'empêche pas, au contraire, qu'un service puisse gérer ses propres traces dans ses propres fichiers. Les traces applicatives ne devraient pas être placées dans les traces de système. Les traces d'accès aux pages Web d'un serveur Apache n'ont rien à y faire. Par contre les traces de connexion au système (via la console, ssh, telnet, etc.) ont un intérêt important et doivent être présentes dans les fichiers logs du système.

2. Les messages

Le service **klogd** gère les messages émis par le noyau. Il dispose de deux sources d'accès aux messages :

- le système de fichiers virtuel /proc, utilisé par défaut s'il est présent, et notamment /proc/kmsg ;
- les appels systèmes via l'API du noyau, notamment sys_syslog, si /proc est absent ou si le paramètre -s a été passé à klogd.

Les messages du noyau ont des niveaux de priorité différents, étagés de 0 (haute priorité) à 7 (message de débogage).

Le service **syslogd** (ou syslog-ng) reçoit les messages issus des services mais aussi de klogd. Il les dispatche ensuite selon l'émetteur, la sévérité, dans des fichiers, des consoles, sous forme de mails aux utilisateurs du système (root par exemple), etc.

Les actions les plus courantes sont l'écriture des logs dans des fichiers, la redirection de messages sur une console ou l'envoi de messages à root.

3. Configuration de syslog

Le fichier de configuration **`/etc/syslog.conf`** permet de définir l'origine, l'importance et la destination de chaque message, sous forme de deux champs.

- L'origine définit en fait un ensemble de **systèmes** et de **sous-systèmes** (noyau, services). La liste, extensible, est composée à l'origine des éléments suivants. L'étoile définit l'ensemble des sous-systèmes.

Sous-système	**Signification**
auth/authpriv	Service de sécurité et d'authentification.
cron	Service cron.
daemon	Les démons du système.
kern	Le noyau.
lpr	Le service d'impression.
mail	La messagerie.
news	Le réseau.
syslog	Syslog luimême.
user	Messages des processus utilisateurs.
uucp	Unix to Unix CoPy.
local0->7	Messages issus de klogd, le chiffre représente le niveau.

- L'importance ou **niveau** définit le niveau de sévérité du message. L'étoile définit l'ensemble de tous les niveaux. Il y a équivalence entre les niveaux émis par klogd et syslogd.

Niveau	Signification
emerg	Le système est inutilisable.
alert	Une intervention immédiate est indispensable.
crit	Erreur critique pour le soussystème.
err	Erreur de fonctionnement.
warning	Avertissement.
notice	Évènement normal méritant d'être signalé.
info	Pour information seulement.
debug	Message envoyé pour la mise au point.
none	Ignorer les messages.

- La destination ou **action** peut être un fichier, un message à un utilisateur, la console, une liste d'utilisateurs... L'étoile indique tout le monde.

Les messages syslog sont inscrits dans les fichiers **/var/log/messages** et **/var/log/syslog** ou dans tout autre fichier paramétré dans **/etc/syslog.conf**.

Vous pouvez vous-même, directement ou dans vos scripts, envoyer des messages à **syslogd** par la commande **logger**.

Archivage et backup

1. Les outils de sauvegarde

La sauvegarde est un travail important de l'administrateur puisqu'en cas de gros problème, on passe généralement par une restauration du système depuis une sauvegarde, ou une image du système lorsque celuici était encore intègre (bon fonctionnement, pas de corruption). Chaque Unix est fourni avec des commandes et des procédures de sauvegarde qui lui sont propres. On distingue tout de même quelques outils communs.

a. Commandes, plans, scripts

- Pour la sauvegarde de fichiers et d'arborescences, utilisez les commandes **tar** et **cpio**. Ces commandes sauvent une arborescence, et pas un système de fichiers.
- Pour la sauvegarde physique de disques et de systèmes de fichiers (des dumps), utilisez la commande **dd**.

Une sauvegarde incrémentale consiste à sauvegarder une première fois la totalité des données, puis ensuite uniquement les fichiers modifiés. On trouve aussi sous forme de logiciels libres ou dans le commerce des solutions plus pointues de sauvegarde.

L'administrateur aura parfois à définir des scripts de sauvegarde et de restauration adaptés au cas par cas (partition système, données applicatives...) et à automatiser quand c'est possible l'exécution de ceux-ci en fonction de la date, l'heure ou la charge de la machine.

Il sera aussi très important de définir un plan de sauvegarde, en se posant les bonnes questions :

- Que faut-il sauvegarder ?
- Avec quelle fréquence ?
- Combien de temps conservera-t-on les sauvegardes, à quel endroit, en combien d'exemplaires ?
- À quel endroit sera stocké l'historique des sauvegardes ?
- Quel est le support le plus approprié ?
- Quels sont les besoins, en capacité, du support de sauvegarde ?
- Combien de temps prévoit-on pour sauvegarder un fichier, un système de fichiers et est-ce raisonnable ?
- La sauvegarde doit-elle être automatique ou manuelle ?
- Quelle est la méthode de sauvegarde la plus appropriée ?

b. Voici quelques autres commandes

- **mt** : contrôle d'une bande magnétique.
- **touch** : met la date de dernière modification à l'heure actuelle, pour forcer une sauvegarde incrémentale.
- **find** : sélectionne les fichiers à sauvegarder.
- **gzip**, **gunzip**, **zcat**, compression et décompression au format GnuZip.
- **bzip2**, **bunzip2**, compression et décompression au format bzip2.

2. Tar

La commande **tar** est simple et efficace. Elle crée des archives des fichiers, y compris l'arborescence de fichiers, sur tout type de support y compris dans une autre fichier (archive à l'extension .tar). L'archive ainsi créée peut s'étendre sur plusieurs volumes : quand la bande ou la disquette est pleine, c'est à l'utilisateur d'en insérer une nouvelle et la sauvegarde/restitution continue.

a. Archiver

La syntaxe est la suivante :

```
tar cvf nom_archive Fichier(s)
```

Par exemple pour placer dans une archive tar le répertoire Desktop :

```
$ tar cvf desktop.tar Desktop/ Desktop/
```

Les paramètres sont les suivants :

- c : création d' archive,
- v : mode bavard, tar indique ce qu'il fait,
- f : le paramètre suivant est le nom de l'archive.

b. Lister

La syntaxe est :

```
tar tvf nom_archive
```

Pour lister le contenu de l'archive précédente :

```
$ tar tvf desktop.tar
```
Le paramètre t liste le contenu de l'archive.

c. Restauration

Pour restaurer le contenu d'une archive la syntaxe est :

```
tar xvf nom_archive fichiers
```

Pour restaurer l'archive précédente :

```
tar xvf desktop.tar
```
Le paramètre x permet l'extraction de l'ensemble des fichiers de l'archive, ou du ou des fichiers spécifiés à la suite
du nom de l'archive.

d. Autres paramètres

La commande **tar** de gnu permet de gérer les formats de compression directement :

- z : l'archive est compressée au format gzip.
- Z : l'archive est compressée au format compress.
- j : l'archive est compressée au format bzip2.

Ainsi les commandes précédentes pour le format de compression gzip deviennent :

```
$ tar cvzf desktop.tar.gz Desktop/ Desktop/
$ ls -l desktop.tar*
-rw-r--r-- 1 seb users 30720 mai  9 11:16 desktop.tar
-rw-r--r-- 1 seb users  7556 mai  9 11:22 desktop.tar.gz
```

Notez la différence de taille. Les options de compression peuvent être utilisées avec c, t et x. Il peut être préférable de ne pas spécifier d'option de compression si vous sauvez sur une bande dont le lecteur gère lui-même la compression.

Si votre archive est compressée et qu'elle est à destination d'un autre système, ou que vous souhaitez garder une compatibilité avec les paramètres par défaut de tar, vous pouvez procéder comme ceci :

```
$ gzip -cd desktop.tar.gz | tar xvf - Desktop/
```
Le paramètre -d précise à **gzip** de décompresser le fichier, tandis que -c passe le résultat par la sortie standard. Le - final indique à tar de récupérer le flux par l'entrée standard.

3. cpio

La commande **cpio** sauvegarde sur la sortie standard les fichiers dont on saisit les noms sur l'entrée standard, par défaut l'écran et le clavier. Vous devez donc utiliser les redirections. Cpio ne compresse pas les archives. C'est à vous de le faire.

a. Archiver

La syntaxe générale est :

```
cpio -oL
```

Les paramètres les plus utilisés sont :

- -o : output, création de la sauvegarde en sortie.
- -L : sauve les fichiers liés et pas les liens symboliques.
- -v : mode bavard « verbose », informations détaillées.
- -c : sauvegarde des attributs des fichiers sous forme ASCII (pour l'échange entre divers OS).

Voici comment archiver et compresser le répertoire Desktop :

```
find Desktop -print| cpio -ocv | gzip > archive.cpio.gz
> ls -l archive.cpio.gz
-rw-r--r-- 1 seb users 7377 mai  9 11:33 archive.cpio.gz
```

b. Lister

La syntaxe générale est :

```
cpio -it archive
```

Les paramètres sont :

- `-i` : lecture de l'archive en entrée.
- `-t` : comme pour tar, liste le contenu de l'archive.

```
$ cat archive.cpio.gz | gzip -cd | cpio -it
```

c. Restaurer

La syntaxe générale est :

```
cpio -i[umd]
```

- `-u` : restauration inconditionnelle, avec écrasement des fichiers qui existent déjà. Par défaut les fichiers ne sont pas restaurés si ceux présents sur le disque sont plus récents ou du même âge.
- `-m` : les fichiers restaurés conservent leur dernière date de modification.
- `-d` : cpio reconstruit l'arborescence des répertoires et sous-répertoires manquants. Pour restaurer l'archive précédente :

```
$ cat archive.cpio.gz | gzip -cd | cpio -iuvd
```

4. dd

La commande **dd** (*device to device*) est destinée à la copie physique, bloc par bloc, d'un fichier périphérique vers un fichier périphérique ou quelconque. À l'origine elle était utilisée pour la lecture et l'écriture sur bande magnétique, mais elle peut être employée avec n'importe quel fichier. La commande **dd** permet de réaliser des copies physiques de disques et de systèmes de fichiers.

Argument	Rôle
if=fichier	Nom du fichier en entrée (celui à copier).
of=fichier	Nom du fichier en sortie.
bs=n	Taille du bloc en octets.
count=n	Nombre de blocs à copier.

Ici vous allez placer le secteur de boot de la partition (où est installé lilo ou grub) dans un fichier. Le fichier ainsi créé pourra être utilisé avec le chargeur de NT/2000/XP pour démarrer sous Linux.

```
# dd if=/dev/sda1 of=boot.lnx bs=442 count=1
```

Pour créer un fichier vide d'une taille de 10Mo :

```
$ dd if=/dev/zero of=vide bs=1M count=10
```