



|   |           |
|---|-----------|
| <b>COURS 1 : ADMINISTRATION SYSTEME LINUX .....</b>                                 | <b>2</b>  |
| <b>INTRODUCTION GENERALE.....</b>   | <b>2</b>  |
| <b>I. GENERALITE LINUX .....</b>  | <b>3</b>  |
| 1. Historique : Unix à Linux.....   | 3         |
| 2. Distributions Linux.....   | 4         |
| 3. Licence GPL.....   | 4         |
| <b>II. ARCHITECTURE LINUX.....</b>  | <b>6</b>  |
| 1. Hiérarchie des dossiers .....  | 6         |
| 2. Root .....   | 7         |
| <b>III. COMPILATION DU NOYAU .....</b>  | <b>8</b>  |
| 1. Notion du noyau .....  | 8         |
| 2. Compilation .....  | 8         |
| <b>IV. INTERPRETEUR DE COMMANDES : Shell .....</b>                                  | <b>11</b> |
| 1. Définition.....  | 11        |
| 2. Types de Shells.....   | 11        |
| <b>V. COMMANDES DE BASE.....</b>  | <b>12</b> |
| 1. Définition.....  | 12        |
| 2. Commandes .....  | 12        |
| <b>VI. COMMANDES COMPLEXES.....</b>   | <b>16</b> |
| 1. Manipulations d'archives et compressions .....                                   | 16        |
| 2. Gestion des disques/points de montage .....                                      | 16        |
| 3. Manipulation de texte .....  | 16        |
| 4. Permissions.....   | 16        |
| 5. Processus .....  | 16        |
| 6. Services et démarrage.....   | 16        |
| 7. Réseaux .....  | 17        |
| 8. Utilisateurs.....  | 17        |
| <b>COURS 2 : ADMINISTRATION RESEAU LINUX .....</b>                                  | <b>18</b> |
| <b>I. GENERALITE .....</b>  | <b>18</b> |
| 1. Introduction .....   | 18        |
| 2. Activités récurrentes de l'administrateur système.....                           | 18        |
| <b>II. GESTIONNAIRES DE PAQUETS .....</b>   | <b>19</b> |
| 1. Définition.....  | 19        |
| 2. Types de gestionnaires de paquets .....  | 19        |
| 3. Gestion des paquets .....  | 19        |
| <b>III. GESTIONS DES RESSOURCES .....</b>   | <b>23</b> |
| 1. Gestion des utilisateurs : utilisateurs, groupes, profils .....                  | 23        |
| 2. Gestion des fichiers .....   | 29        |
| 3. Gestion des disques : partitions, formatage et montage de disques, profils ..... | 31        |
| 4. Gestions d'amorçage : LILO et Grub.....  | 32        |
| 5. Gestion des processus : planification de tâches, les signaux .....               | 32        |
| <b>IV. CONFIGURATION DES SERVICES .....</b>   | <b>34</b> |
| 1. Réseau .....   | 34        |
| 2. DNS : Bind .....   | 39        |
| 3. Serveur web (HTTP) .....   | 44        |
| <b>V. OUTILS D'ADMINISTRATION .....</b>   | <b>50</b> |
| 1. Webmin .....   | 50        |
| 2. Poste de pilotage.....   | 51        |
| 3. Shorewall.....   | 52        |
| 4. Nagios.....  | 53        |
| 5. phpMyAdmin .....   | 54        |
| <b>CONCLUSION GENERALE.....</b>   | <b>55</b> |

# COURS 1 : ADMINISTRATION SYSTEME LINUX

## INTRODUCTION GENERALE

**GNU/Linux** (GNU : GNU is Not Unix) est un système d'exploitation entièrement constitué de **logiciel libre**. Il a été lancé en 1983 et a été développé par beaucoup de gens travaillant ensemble pour donner à tous les utilisateurs de logiciels GNU, la liberté de contrôler leur informatique.

Sur le plan technique, GNU est d'une manière générale similaire à Unix. Mais contrairement à Unix, il est « **Open Source** ».

Un logiciel Open Source est un code conçu pour être accessible au public : n'importe qui peut voir, modifier et distribuer le code à sa convenance.

Ce type de logiciel est développé de manière collaborative par une communauté, et repose sur l'examen par les pairs.

Linux est un système d'exploitation Open Source et gratuit, distribué sous **licence publique générale GNU**.

Le système d'exploitation Linux a été créé du système d'exploitation **MINIX**, qui s'appuyait lui-même sur les principes et la conception d'**Unix**.

La publication de Linux sous une licence Open Source écarte toute restriction d'utilisation du logiciel.

Par conséquent, tout le monde peut utiliser, étudier, modifier et redistribuer le code source, ou même vendre des copies du code modifié, tant que la licence reste la même.

Il existe alors de multiples raisons qui peuvent expliquer la décision d'opter pour un logiciel Open Source plutôt qu'un logiciel propriétaire.

Voici les plus courantes :

- ✓ **Examen par les pairs,**
- ✓ **Transparence,**
- ✓ **Fiabilité,**
- ✓ **Flexibilité,**
- ✓ **Coûts inférieurs,**
- ✓ **Pas de dépendance vis-à-vis d'un fournisseur,**
- ✓ **Collaboration ouverte.**

## I. GENERALITE LINUX

### 1. Historique : Unix à Linux

**UNIX** désigne une famille de systèmes d'exploitation dont le premier a été conçu en 1969 aux **laboratoires Bell** (Bell Laboratories) par des ingénieurs, puis réécrit en langage C, puis porté sur de nombreuses architectures matérielles, avec une importante contribution de l'**université de Berkeley**.

C'est un système qui est assez vieux, utilisé tant pour les gros ordinateurs que pour les plus petits.

Les principales versions actuelles sont :

- ✓ **System VR4,**
- ✓ **OSF/1,**
- ✓ **SUN Solaris.**

Il est retrouvé sur :

- ✓ **Les super-ordinateurs (Cray),**
- ✓ **Les ordinateurs centraux,**
- ✓ **Les minis (VAX, HP),**
- ✓ **Les postes de travail (HP, Apollo, Sun, SGI, ...).**

Par contre, **Linux** est un système d'exploitation proche des systèmes UNIX pouvant être exécuté sur différentes plates-formes matérielles : **x86**

**Linus B.Torvalds** est à l'origine du **noyau** du système d'exploitation « **Linux** » entièrement libre au début des années 90.

En 1991, l'étudiant finlandais Linus Torvalds, indisposé par la **faible disponibilité du serveur informatique UNIX** de l'université d'Helsinki, entreprend le développement d'un noyau de système d'exploitation, qui prendra le nom de « **noyau Linux** » ou « **kernel** ».

C'est en mars 1992 qu'a été diffusée la première version ne comportant quasiment aucun bug.

Bien que Linux ait été initialement conçu pour fonctionner sur plateforme PC, il a désormais été adapté à d'autres plateformes, telles que :

- ✓ Macintosh,
- ✓ Stations SPARC,
- ✓ Stations DEC Alpha,
- ✓ Personnel Computer (PC),
- ✓ Assistants personnels (PDA),
- ✓ Consoles de jeu vidéo.

## 2. Distributions Linux

Une distribution Linux est un système d'exploitation créé à partir d'une **collection de logiciels** utilisant le noyau Linux/GNU. Dans la plupart des distributions, les logiciels disponibles sont libres, développés en langage open source.










La plupart des distributions proposent également une installation graphique qui leur est propre ainsi qu'un système de gestion de paquets (**RPM, APT GET**) permettant d'installer automatiquement des logiciels en gérant les dépendances (les logiciels sous Linux sont parfois liés à des bibliothèques externes ou s'appuient sur d'autres logiciels).

Chaque distribution possède ses avantages et ses inconvénients.

En effet si certaines sont plus adaptées à des débutants et proposent des interfaces graphiques évoluées, d'autres privilégient la sécurité ou l'évolutivité.

Les distributions les plus connues sont :

**Tableau N°1 : Distributions linux**

| Ordre | Distributions | Logos   |
|-------|---------------|---|
| 1     | RedHat        |  redhat.   |
| 2     | Ubuntu        |  ubuntu    |
| 3     | Debian        |  debian   |
| 4     | SuSe          |  SUSE     |
| 5     | Knoppix       |           |
| 6     | Slackware     |           |
| 7     | Mandriva      |  Mandriva |

### 3. Licence GPL

Le mouvement du logiciel libre est apparu en 1983 avec le **projet GNU**, créé par **Richard Stallman**.

Ce mouvement s'est organisé autour de l'idée des libertés des utilisateurs.

À l'opposé du logiciel libre, on trouve le **logiciel propriétaire** ou à « code source fermé ». Un logiciel fermé est strictement protégé et seuls les propriétaires du code source sont en droit d'accéder à ce code.

Linux est un noyau à « **code source ouvert** », d'où accessible gratuitement.

Afin de permettre la distribution de logiciels libres exempts de droits, la **FSF** (Free Software Foundation) (littéralement « **Fondation pour le logiciel libre** »), une organisation américaine à but non lucratif fondée par Richard Stallman, le 4 octobre 1985, a reçu comme mission la promotion du logiciel libre et la défense des utilisateurs.

Les utilitaires GNU sont alors soumis aux termes de la licence d'utilisation **GPL** (General Public License).

L'objectif de la licence GPL est de garantir à l'utilisateur les droits suivants (appelés libertés) sur un logiciel libre :

- ✓ **Liberté 0** : La liberté d'exécuter le logiciel, pour n'importe quel usage ;
- ✓ **Liberté 1** : La liberté d'étudier le fonctionnement d'un programme et de l'adapter à ses besoins ;
- ✓ **Liberté 2** : La liberté de redistribuer des copies ;
- ✓ **Liberté 3** : L'obligation de faire bénéficier la communauté des versions modifiées.

## II. ARCHITECTURE LINUX

### 1. Hiérarchie des dossiers

Contrairement à MS-DOS, Unix voit ses disques comme une unique arborescence.

Une partition contient la racine du système de fichier, qu'on note /

La structure "standard" des dossiers est décrite par le **FHS** (Filesystem Hierarchy Standard) auquel la plupart des distributions Linux essaient de se conformer.

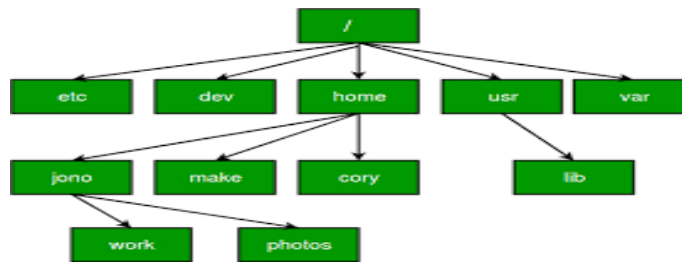


Figure 1

Les **principaux dossiers** sont :

**/root** : contient le dossier de l'administrateur système.

**/etc** : contient des fichiers de configuration.

**/dev** : contient les fichiers spéciaux correspondant aux périphériques.

**/home** : contient les dossiers personnels des utilisateurs.

**/usr** : contient les logiciels installés avec le système.

**/usr/bin** :

les exécutables.

**/usr/sbin** :

les serveurs réseau principalement.

**/usr/src** :

les sources de certains logiciels, principalement le noyau de Linux.

**/var** : contient des données fréquemment réécrites.

**/var/lib** :

les bases de données, des fichiers de config...

**/var/log** :

le journal du système.

**/bin** : contient les commandes de base.

**/sbin** : contient les commandes de base nécessaires à l'administration système.

**/boot** : contient les informations nécessaires au démarrage de la machine.

**/lib** : contient les principales bibliothèques partagées (équivalent DLL de Windows).

**/proc** : contient des infos sur l'état du système et des processus en cours d'exécution.

**/tmp** : contient les fichiers temporaires.

## 2. Root

Le root est le nom donné à un compte disposant des **droits d'administrateur** sur un système d'exploitation Linux. Ce type d'accès permet de réaliser des modifications à la racine du système et ainsi de définir des réglages sensibles.

**/root: #**

En anglais, « root » signifie d'ailleurs racine.

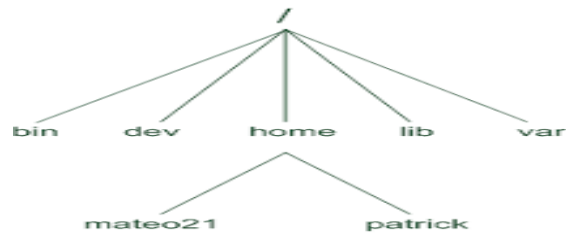


Figure 2

### III. COMPILATION DU NOYAU

#### 1. Notion du noyau

Linux est architecturé autour d'un **noyau** (en anglais **kernel**) chargé de prendre en charge le matériel.

On appelle **distribution** l'assemblage de logiciels autour d'un noyau Linux afin de fournir un système tout fonctionnel.

Le noyau d'une distribution peut être **mis à jour** afin de permettre la prise en compte de matériels récents, toutefois cette manipulation consistant à **recompiler le noyau** est délicate car elle nécessite un certain niveau de connaissance du système et du matériel.

La recompilation du noyau est à réserver aux spécialistes ou bien aux utilisateurs prêts à rendre inutilisable leur système dans le but d'apprendre.

**Tableau N°1 : Noyau de linux (Kernel)**

| Créateur         | Linus Torvalds           |
|------------------|--------------------------|
| Première version | 0.01 (17 septembre 1991) |
| Dernière version | 6.5.6 (6 octobre 2023)   |

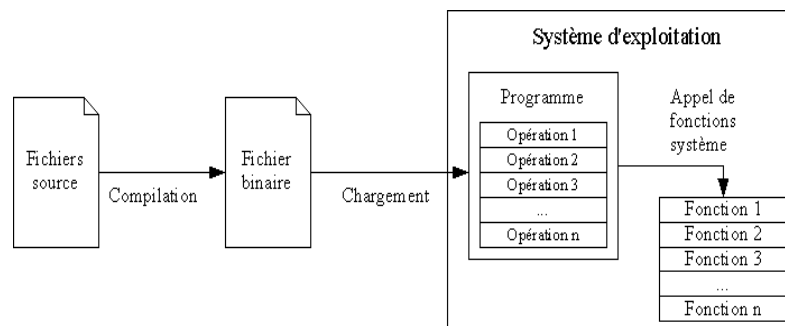
#### 2. Compilation

Les programmes sont en général écrits dans un certain nombre de fichiers, appelés **fichiers sources**, du fait d'être à l'origine du programme. Le texte de ces fichiers est appelé le **code source**, ou plus simplement le code.

Pour exécuter un programme à partir de son code source, il n'y a que deux solutions :

- ✓ La **compilation** : disposer d'un **compilateur** capable de traduire le code source en langage binaire, qui sera alors directement exécutable par l'ordinateur.
- ✓ L'**interprétation** : disposer d'un **interpréteur** capable de lire le code source et d'effectuer les opérations décrites dans le code source.

Les programmes compilés sont beaucoup plus rapides à l'exécution, puisque la phase d'analyse du code source a été réalisée au préalable et se fait en dehors de la phase d'exécution.



**Figure 3 : Phase de compilation**

Les **programmes compilés** sont notamment le noyau lui-même, le shell, les commandes de base et les applications.



En général, au cours de la compilation, le processus génère un fichier binaire pour chaque fichier source. Ces fichiers binaires sont nommés **fichiers objets**, et porte de ce fait l'extension « .o » (ou « .obj » dans les systèmes Microsoft).

Comme un programme peut être constitué de plusieurs fichiers sources, il faut regrouper les fichiers objets pour générer **le fichier exécutable** du programme, fichier que l'on appelle également le **fichier image** en raison du fait que c'est le contenu de ce fichier qui sera chargé en mémoire par le système pour l'exécuter.

**L'opération de regroupement** des fichiers objets pour constituer le fichier image s'appelle **l'édition de liens**, et elle est réalisée par un programme nommé le **linker** (éditeur de liens en français).

Certains fichiers objets nécessaires pour **tous les programmes** et définissant les fonctions de base sont regroupés dans des **bibliothèques (librairies)**.

Les bibliothèques portent souvent l'extension « .a » (ou « .lib » dans les systèmes Microsoft).

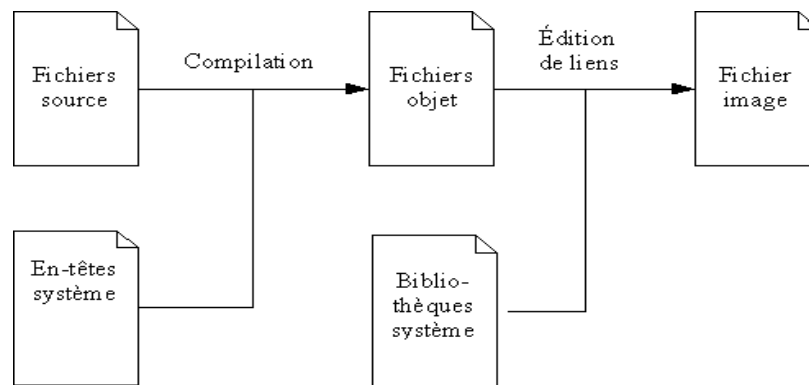


Figure 4 : Phase de l'édition de liens (Bibliothèque)

Malheureusement, la solution consistant à stocker dans des bibliothèques les fonctions les plus utilisées souffre de la duplication du code contenu dans ces bibliothèques dans tous les programmes, d'où une perte de place considérable.

C'est pour cela que les **bibliothèques dynamiques** ont été créés : une bibliothèque dynamique n'est pas incluse dans les fichiers des exécutables qui l'utilisent, mais reste dans un fichier séparé.

Les bibliothèques sont regroupées dans un dossier bien défini du système de fichiers, ce qui permet de les partager entre différents programmes.

Les bibliothèques dynamiques portent l'extension « .so » (pour « Shared Object »), ou « .dll » dans les systèmes Microsoft (pour « Dynamic Link Library »).

Pour compiler un programme, on a réellement besoin que de trois types de fichiers :

- ✓ Les fichiers de **déclaration du système** ;
- ✓ Les fichiers des **bibliothèques de base** ;
- ✓ Les **fichiers sources** (déclaration et définition) du programme à compiler.

En général, la compilation d'un programme passe par les étapes suivantes :

- ✓ Récupération des **fichiers sources** du programme ;
- ✓ **Configuration** du programme pour l'environnement courant ;
- ✓ Appel à **make** pour la compilation ;
- ✓ Appel à **make install** pour l'installation.

La **première étape** est élémentaire et va de soi.

La **deuxième étape** se fait souvent en appelant un script dans le dossier d'installation des fichiers sources.

Ce script se nomme souvent configure, et peut être appelé avec la ligne de commande suivante :

**./configure**

à partir du dossier où se trouvent les fichiers sources. Ce script effectue tous les tests sur le système et l'environnement de développement, et génère le fichier « **makefile** ».

Le programme configure est en général fourni avec les logiciels GNU. Il permet de déterminer l'environnement logiciel et système et de générer le fichier makefile et des fichiers d'en-têtes contenant la définition de quelques options du programme.

La **troisième étape** est très simple aussi. Il suffit de taper la commande suivante :

**Make**

toujours dans le dossier où se trouvent les fichiers sources.

Enfin, la dernière étape se fait en tapant la commande suivante :

**make install**

Bien entendu, ces différentes étapes varient parfois avec le logiciel à installer.

Cependant, il existe quasiment toujours un fichier texte « **readme** », indiquant comment effectuer ces opérations dans le dossier des fichiers sources.

## IV. INTERPRETEUR DE COMMANDES : Shell

### 1. Définition

Un shell est un interpréteur de commandes qui permet à un utilisateur de lancer des commandes de façon interactive. C'est le programme chargé d'assurer le dialogue avec l'utilisateur et gère l'invite de commandes.

Sous linux, on dispose de plusieurs shells, la tendance actuelle est à l'utilisation du BASH (Bourne Again Shell).

### 2. Types de Shells

Les autres shells connus sont :

- ✓ **sh** : Bourne **Shell**. L'ancêtre de tous les **shells**.
- ✓ **bash** : Bourne Again **Shell**. Une amélioration du Bourne **Shell**.
- ✓ **ksh** : Korn **Shell**.
- ✓ **csh** : C **Shell**.
- ✓ **tcsh** : Tenex C **Shell**.
- ✓ **zsh** : Z **Shell**.

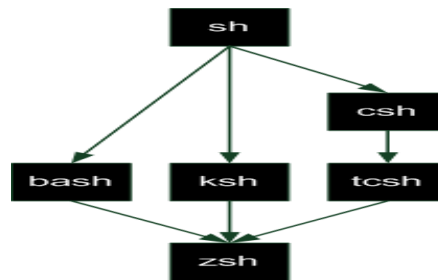


Figure 5: Shells

Tous ces shells sont capables d'assurer l'exécution d'une suite de commandes, ce que l'on appelle un script. Tous ces shells sont de véritables langages de programmation.

## V. COMMANDES DE BASE

### 1. Définition

Le système GNU Linux de base contient un noyau, des outils et des fichiers de données et de configuration.

Les commandes Linux s'utilisent pour des opérations sur des dossiers ; elles permettent ainsi de créer, supprimer et gérer des dossiers sur un système via un terminal mais aussi de naviguer dans l'arborescence des dossiers.

Les systèmes d'exploitation de type GNU Linux offrent à leurs utilisateurs des **centaines de commandes** qui font de la console un outil pratique et extrêmement puissant. Certaines d'entre elles sont fournies directement par le shell, alors que d'autres sont des exécutables situés dans **/bin**, **/usr/bin**, **/usr/local/bin**.



Figure 6 : Commandes de base

### 2. Commandes

L'objectif est de présenter quelques commandes de base pour mettre le pied à l'étrier avec votre terminal.

Chaque utilisateur connecté au système d'exploitation est capable de diriger la machine en exécutant une commande dans un terminal :

✓ La commande **pwd**

Cette commande permet d'afficher l'emplacement où on se situe actuellement dans la hiérarchie FHS.

```
[root@centos ~]# pwd
/root
```

✓ La commande **cd**

Cette commande permet de changer de dossier courant et de se situer sur un autre

```
[root@centos ~]# cd /home/
[root@centos home]# pwd
/home
```

✓ La commande **ls**

Permet de lister les fichiers disponibles dans un dossier, si appelé sans arguments, ls liste les fichiers du dossier courant.

```
[root@centos home]# ls
```

```
TIENDREBEOGO
```

```
DIALLO
```

=> Ainsi, dans le dossier home, deux fichiers existent et sont : user1 et user2

✓ La commande **mkdir**

Cette commande permet de créer un dossier.

```
[root@centos home]# cd /tmp
```

```
[root@centos tmp]# pwd
```

```
/tmp
```

```
[root@centos tmp]# mkdir UTM
```

```
[root@centos tmp]# ls
```

```
UTM
```

```
fichiertemp
```

=> Dans cet exemple, on s'est déplacé sous le dossier tmp disponible sous la racine /, afficher notre emplacement grâce à la commande pwd, puis créer un dossier nommé repertoiredest et le visualisé avec la commande ls.

✓ La commande **rmdir**

Cette commande permet de supprimer un dossier.

```
[root@centos tmp]# rmdir UTM
```

```
[root@centos tmp]# ls
```

```
fichiertemp
```

=> Avec cette commande, on vient de supprimer le dossier qu'on vient de créer et la commande ls nous l'a confirmé.

✓ La commande **touch**

Cette commande permet de changer la date du dernier accès ou modification d'un fichier, mais permet également de créer un fichier vide.

```
[root@centos tmp]# touch devoir
```

```
[root@centos tmp]# ls
```

```
devoir
```

```
fichiertemp
```

=> On a créé un fichier nommé fichier.

✓ La commande **cp**

Cette commande permet de copier un fichier ou un dossier.

```
[root@centos tmp]# cp devoir classe
```

```
[root@centos tmp]# ls
```

```
devoir classe
```

```
fichiertemp
```

=> Avec la commande cp, on a copié le fichier « **devoir** » en le copiant dans **classe**.

✓ La commande **rm**

Cette commande permet de supprimer un fichier ou un dossier. Cette commande est à utiliser avec précaution car avec l'option **-f** ou **-rf**, cette commande peut endommager voir supprimer tout votre système de fichiers de manière irréversible.

```
[root@centos tmp]# rm -f devoir classe
[root@centos tmp]# ls
fichier temp
```

=> La commande **rm** nous a permis dans cet exemple de supprimer les deux fichiers précédemment créés.

✓ La commande **mv**

Cette commande sert à renommer ou déplacer un fichier ou un dossier.

Dans l'exemple suivant, on va créer un fichier nommé "**etudiant**", et à l'aide de la commande **mv**, on va le renommer en "**connaissance**" et le déplacer dans le dossier **/home** en une seule fois.

```
[root@centos tmp]# touch etudiant
[root@centos tmp]# mv etudiant /home/connaissance
[root@centos tmp]# ls /home/
connaissance TIENDREBEOGO DIALLO
```

✓ La commande **cat**

Cette commande permet d'afficher le contenu d'un fichier.

```
[root@centos tmp]# cat /home/connaissance
[root@centos tmp]#
```

Le fichier "**connaissance**" est vide, donc la commande "cat" ne renverra aucun résultat, dans un autre cas on pourrait avoir :

```
[root@centos tmp]# cat /etc/resolv.conf

# Generated by NetworkManager
nameserver 8.8.8.8
```

Le fichier **resolv.conf** détient les informations DNS : **nameserver 8.8.8.8**

✓ La commande **echo**

Cette commande permet d'afficher une ligne.

```
[root@centos tmp]# echo "tutoriel Linux de sitedetout"
tutoriel Linux de sitedetout
```

=> **echo** permet aussi **d'écrire** du contenu dans un fichier moyennant le signe ">" pour **écraser** le contenu du fichier ou ">>" pour suffixer le contenu du fichier.

✓ La commande **man**

Cette commande permet d'afficher le manuel d'aide de n'importe quelle autre commande disponible,  
ex : **man ls**

**man ls**

Tapez la lettre q pour fermer la page du manuel.

✓ La commande **ln**

Cette commande est utilisée pour créer un lien matériel ou symbolique (raccourci) vers un fichier ou un dossier.

En parcourant le manuel utilisateur de la commande **ln** (**man ln**), on peut voir que l'option **-s** sert à créer un lien symbolique à la place des liens physiques.

**man ln**

A suivre le cas suivant :

```
[root@centos tmp]# touch fichiersource
[root@centos tmp]# echo "contenu" > fichiersource
[root@centos tmp]# cat fichiersource
contenu
[root@centos tmp]# ln -s fichiersource fichiercible
[root@centos tmp]# cat fichiercible
contenu
```

✓ La commande **which**

Cette commande permet de visualiser l'emplacement d'une commande en effectuant une recherche dans différents dossiers.

```
[root@centos tmp]# which pwd
/bin/pwd
```

✓ La commande **more**

La commande **more** permet de visualiser le contenu d'un fichier page à page.

```
[root@centos tmp]# touch fichier
[root@centos tmp]# echo "contenu" >> fichier
[root@centos tmp]# more fichier
contenu
```

✓ La commande **tail**

La commande **tail** permet d'afficher la fin d'un fichier (par défaut, les 2 dernières lignes)

```
[root@centos tmp]# tail -n fichier2
ligne 11
ligne 12
```

✓ La commande **find**

La commande **find** permet de chercher un fichier sous Linux.

## VI. COMMANDES COMPLEXES

### 1. Manipulations d'archives et compressions

- ✓ **cpio** : copie de fichiers à partir de ou vers une archive cpio/tar
- ✓ **gzip/gunzip** : compresse et décompresse des fichiers. (Gnu ZIPper)
- ✓ **tar** : manipulation d'archives, capable de fonctionner avec bzip2 ou gzip.
- ✓ **zip/unzip** : compresse et décompresse des fichiers. (ZIP)

### 2. Gestion des disques/points de montage

- ✓ **df** : affiche l'espace libre sur les partitions (Disk Free).
- ✓ **dump/restore** : sauvegarde et restauration d'un système de fichiers.
- ✓ **mount** : attache un système de fichiers sur un point de montage.
- ✓ **umount** : détache un système de fichiers. (UnMOUNT). « unmount » reste autorisé.

### 3. Manipulation de texte

- ✓ **cat** : concatène des fichiers texte. Peut aussi servir à afficher ou lire un fichier.
- ✓ **cut** : supprime une partie des lignes d'un fichier selon un critère.
- ✓ **grep** : affiche les lignes qui contiennent une expression régulière donnée
- ✓ **head** : affiche les premières lignes d'un fichier. (opposé de tail)
- ✓ **nl** : permet de numéroté les lignes d'un fichier.
- ✓ **read** : lit une chaîne de caractères à partir de l'entrée standard.
- ✓ **sort** : trie les lignes d'un texte selon l'ordre alphabétique (numérique avec l'option -n).
- ✓ **tail** : affiche les dernières lignes d'un fichier. (opposé de head)

### 4. Permissions

- ✓ **chgrp** : change le groupe propriétaire d'un fichier. (CHange GRouP)
- ✓ **chmod** : change les permissions en lecture, écriture et/ou exécution d'un fichier. (CHange MODes)
- ✓ **chown** : change le propriétaire, et éventuellement le groupe propriétaire d'un fichier. (CHange OWNer)

### 5. Processus

- ✓ **cron** : permet de programmer l'exécution d'un programme de façon cyclique.
- ✓ **free** : affiche des informations sur l'utilisation de la mémoire.
- ✓ **sleep** : suspend l'exécution d'un processus pendant un intervalle de temps.
- ✓ **fuser** : affiche quel processus utilise le fichier donné en paramètre.
- ✓ **kill** : envoyer un signal à un processus donné, généralement pour y mettre fin
- ✓ **killall** : tue tous les processus d'un certain type, ou leur envoie un signal donné.
- ✓ **ps** : affiche les processus en cours d'exécution (Process Status)

### 6. Services et démarrage

- ✓ **init** : change le niveau d'exécution (runlevel) du système.
- ✓ **runlevel** : donne le niveau d'exécution en cours
- ✓ **service** : démarre, arrête ou redémarre un service, sur Red Hat et Debian
- ✓ **who am I** : affiche l'identifiant (login) initial
- ✓ **whoami** affiche l'identifiant (login) avec lequel on est connecté



## 7. Réseaux

- ✓ **ftp** : client FTP en ligne de commande.
- ✓ **host** : affiche le nom d'hôte ou l'adresse IP de la machine en argument.
- ✓ **hostname** : affiche ou modifie le nom d'hôte de la machine.
- ✓ **ifconfig** : configurer et afficher les paramètres d'une interface réseau
- ✓ **iwconfig** : pour les réseaux sans fil (affichage et configuration)2
- ✓ **links** : navigateur web en mode texte
- ✓ **lynx** : navigateur web en mode texte
- ✓ **mail** : lire ou envoyer des courriels.
- ✓ **nslookup** : permet de connaître l'adresse IP d'un ordinateur, via un serveur DNS.
- ✓ **ping** : effectue un ping sur une machine distante, utilise ICMP.
- ✓ **route** : manipulation des tables de routage.
- ✓ **rsync** : synchronise un dossier entre deux machines distantes.
- ✓ **scp** : copie distante sécurisée (Secure Copy).
- ✓ **ssh** : client SSH (Secure SHell).
- ✓ **traceroute** : trace la route vers une machine distante routeur par routeur.
- ✓ **wget** : (Www GET) télécharge des fichiers via les protocoles HTTP, HTTPS et FTP.
- ✓ **whois** : informations sur les enregistrements de noms de domaine ou d'adresses IP

## 8. Utilisateurs

- ✓ **finger** : donne des renseignements sur l'utilisateur
- ✓ **groupadd** : ajoute un groupe d'utilisateurs.
- ✓ **groupmod** : modifier les paramètres d'un groupe utilisateur déjà créé par la commande groupadd. Cette commande a les mêmes options que la commande groupadd.
- ✓ **groupdel** : supprime un groupe d'utilisateurs.
- ✓ **su** : commence un nouveau shell ou une autre commande en changeant l'utilisateur.
- ✓ **id** : affiche l'identité de l'utilisateur.
- ✓ **sudo** : exécute un processus avec les droits d'un autre utilisateur selon les règles définies dans le fichier /etc/sudoers
- ✓ **users** : liste Compacte des utilisateurs connectés dans le système ('who' est meilleur).
- ✓ **useradd** : ajoute un utilisateur au système.
- ✓ **usermod** : modifie les paramètres d'un compte utilisateur déjà créé par useradd.
- ✓ **userdel** : supprime un utilisateur du système.

### **I. GENERALITE**

#### **1. Introduction**

Le système d'exploitation GNU/Linux offre une grande variété d'outils en ligne de commande et graphiques et pour mettre en place un serveur performant, stable et peu coûteux.

Ce cours permet d'acquérir les connaissances essentielles et les compétences pratiques pour exploiter au maximum les possibilités de Linux.

Il apprend à créer et modifier des fichiers Linux, à effectuer des recherches dans ces fichiers, à gérer les autorisations et la propriété, à traiter et formater des données texte ainsi qu'à utiliser des scripts shell pour réaliser de nombreuses tâches.

#### **2. Activités récurrentes d'un administrateur système**

La complexité des tâches d'administration de système Linux 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. 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.

Les principales tâches d'un administrateur système Linux sont :

- ✓ Gérer, contrôler et automatiser les outils du système GNU/Linux,
- ✓ Créer, modifier et supprimer des utilisateurs et groupes d'utilisateurs,
- ✓ Créer, modifier et rechercher des fichiers et dossiers Linux,
- ✓ Limiter l'accès au système de fichiers en contrôlant les permissions et droits,
- ✓ Exploiter les fonctions du shell Bash pour améliorer l'interface en ligne de commande,
- ✓ Exécuter des tâches multiples au moyen de scripts shell.

## II. GESTIONNAIRES DE PAQUETS

### 1. Définition

Un paquet est une **archive** comprenant les **fichiers informatiques**, les **informations** et **procédures** nécessaires à l'installation d'un **logiciel** sur un **système d'exploitation** au sein d'un agrégat logiciel, en s'assurant de la cohérence fonctionnelle du système ainsi modifié.

Un **gestionnaire de paquets** est un (ou plusieurs) outil(s) automatisant le processus d'installation, désinstallation, mise à jour de logiciels installés sur un système informatique. Le terme est surtout utilisé pour les systèmes d'exploitation basés sur Unix, tels **GNU/Linux**.

Les systèmes d'exploitation **GNU/Linux** utilisent dans leur majorité un gestionnaire de paquets, souvent fourni en standard.

Le gestionnaire de paquets permet d'effectuer différentes opérations sur les paquets disponibles :

- ✓ Utilisation des paquets provenant de supports variés (CD d'installation, **dépôts** sur internet, partage réseau...) ;
- ✓ Vérification des **sommes de contrôle** de chaque paquet récupéré pour en vérifier l'**intégrité** ;
- ✓ Vérification des **dépendances logicielles** afin d'obtenir une version fonctionnelle d'un paquetage.

### 2. Types de gestionnaires de paquets

Plusieurs programmes permettent d'automatiser les résolutions de dépendance et le téléchargement des paquets.

- ✓ **RPM** : Red Hat Package Manager, système de gestion de paquets de logiciels.
- ✓ **APT** : Advanced Packaging Tool, système de gestion de paquets (Debian et dérivés)
- ✓ **SMART** : système de gestion de paquets de logiciels multi-distributions ( .deb, .rpm,...)
- ✓ **URPMI** : système de gestion de paquets de logiciels développé par Mandriva.
- ✓ **YUM** : système de gestion de paquets de logiciels développé pour Yellow Dog.
- ✓ **DNF** : Dandified Yum, logiciel multi-distributions : Fedora, CentOS, et Red Hat.

### 3. Techniques de gestions des paquets

#### 3.1. Gestionnaire des paquets sous Debian

Il y a beaucoup d'outils utilisés pour la gestion des paquets Debian, depuis des outils basés sur des interfaces texte ou graphique aux outils de bas niveau utilisés pour l'installation des paquets.

Tous les outils disponibles reposent sur les outils de plus bas niveau pour fonctionner correctement et sont présentés ici selon un niveau de complexité décroissant.

Il est important de comprendre que les outils de gestion des paquets Debian de plus haut niveau comme **aptitude** ou **synaptic** reposent sur **apt** qui, lui-même, utilise **dpkg** pour la gestion des paquets sur le système.

## ✓ APT

APT (Advanced Package Tool) est une interface avancée pour le système de gestion des paquets Debian et il fournit le programme **apt-get**. Il fournit des outils en ligne de commandes pour chercher et gérer des paquets, ainsi que pour chercher des informations à leur sujet ou accéder à toutes les fonctionnalités de bas niveau de la bibliothèque d'arborescence libapt-pkg.

Certaines commandes fréquemment utilisées comme **apt-get** et **apt-cache** ont des équivalents dans le nouveau binaire **apt**.

Cela signifie que certaines commandes populaires telles que :

- apt-get update,
- apt-get install,
- apt-get remove,
- apt-cache search, ou apt-cache show.

sont disponibles maintenant à l'aide d'**apt**, ce qui donne **apt update**, **apt install**, **apt remove**, **apt search**, ou **apt show**.

Voici un aperçu des anciennes et des nouvelles commandes :

- apt-get update -> apt update
- apt-get upgrade -> apt upgrade
- apt-get dist-upgrade -> apt full-upgrade
- apt-get install paquet -> apt install paquet
- apt-get remove paquet -> apt remove paquet
- apt-get autoremove -> apt autoremove
- apt-cache search chaîne -> apt search chaîne
- apt-cache policy paquet -> apt list -a paquet
- apt-cache show paquet -> apt show paquet
- apt-cache showpkg paquet -> apt show -a paquet

L'outil **apt** fusionne des fonctions d'apt-get et d'apt-cache. Pour des scripts ou des cas d'usage avancé, apt-get reste préférable voire nécessaire.

Les options les plus courantes d'**apt-get** :

- Pour **mettre à jour** la liste des paquets connus par votre système :  
**apt update**
- Pour **installer** le paquet foo et toutes ses dépendances :  
**apt install foo**
- Pour **supprimer** le paquet foo de votre système :  
**apt remove foo**
- Pour supprimer le paquet foo et ses fichiers de configuration de votre système :  
**apt purge foo**
- Pour lister tous les paquets pour lesquels une nouvelle version est disponible:  
**apt list -upgradable**
- Pour mettre à jour tous les paquets de système, sans installer de paquets supplémentaires ou en supprimer :  
**apt upgrade**

**NB :** Etre authentifié en tant que root pour exécuter toute commande modifiant les paquets du système.

- Pour trouver les paquets dont la description contient *mot* :  
**apt search *mot***
- Pour afficher des informations détaillées sur un paquet :  
**apt show *paquet***
- Pour afficher les dépendances d'un paquet :  
**apt-cache depends *paquet***
- Pour afficher des informations détaillées des versions disponibles pour un paquet et les paquets ayant des dépendances inverses sur lui :  
**apt-cache showpkg *paquet***

### 3.2. Gestion des packages RPM

**RPM Package Manager** (Red Hat Package Manager), ou plus simplement **RPM**, est un système de gestion de paquets de logiciels utilisé sur certaines distributions GNU/Linux.

RPM s'utilise en **ligne de commande** ou avec une interface graphique et permet d'installer, de désinstaller, de vérifier, d'interroger et de mettre à jour des paquets logiciels.

Chaque paquet de logiciels se compose d'une archive de fichiers et d'informations relatives au paquet, tels que sa licence, sa version, sa description, ses dépendances, etc.

La commande rpm permet, entre autre, d'installer, de supprimer ou de mettre à jour un applicatif par l'utilisation de l'option adéquate.

**#rpm [option] nom\_de\_package**

Les noms de package ont un format bien déterminé ; par exemple le package de gestion des quotas disque aura pour nom :

**quota-1.55-4.i386.rpm**

**Tableau N°2 : Format de nom de package**

|  |  |
|--|--|
| Format :<br><b>quota-1.55-4.i386.rpm</b> |  |
| <b>quota</b>                             | <b>est le nom</b>                              |
| <b>1.55</b>                              | <b>est la version</b>                          |
| <b>4</b>                                 | <b>est la release</b>                          |
| <b>i386</b>                              | <b>est l'architecture matérielle supportée</b> |
| <b>.rpm</b>                              | <b>est l'extension typique</b>                 |

Les **distributions les plus connues** utilisant le système de gestion de paquet RPM sont :

- Red Hat Enterprise Linux,
- Fedora,
- CentOS,
- Mandriva,
- openSUSE,
- SUSE Linux Enterprise.

## Commandes d'utilisation :

### Installation et mise à jour de paquets

- **rpm -ivh** nom\_paquet\_xxx.rpm  
ou alors
- **rpm -Uvh** nom\_paquet\_xxx.rpm

### Interrogation

- **rpm -qpil** nom\_paquet\_xxx.rpm
- **rpm -qil** nom\_paquet
- **rpm -qf** /chemin\_du\_fichier

### Désinstallation

- **rpm -e** nom\_paquet
- **rpm -e --nodeps** nom\_paquet

### III.GESTIONS DES RESSOURCES

#### 1. Gestion des utilisateurs : utilisateurs, groupes, profils

La famille GNU/Linux, est un système foncièrement multi-utilisateurs.

Un compte d'utilisateur est une collection de paramètres relatifs à un profil particulier.

Ces paramètres incluent les détails d'identité d'un usager (nom, prénom, photo de profil, etc.), la liste du ou des groupes d'utilisateurs dont son compte fait partie et de nombreuses données personnelles (boîtes de courriels, trousseaux de mots de passe, préférences des logiciels utilisés, choix de thème de l'environnement de bureau, etc.)

Il existe deux types d'utilisateurs sur le système :

- ✓ **root**  
Login nom du compte administrateur du système.  
L'utilisateur **root** à tous les droits sur le système.
- ✓ **Tous les autres utilisateurs du système**  
Utilisateurs créés par l'administrateur.

L'outil **Utilisateurs et groupes** permet de gérer les comptes d'utilisateurs et les groupes d'utilisateurs.

Il dispose de fonctionnalités pour :

- ✓ **Créer ou supprimer** des comptes d'**utilisateurs** ;
- ✓ **Modifier** les **paramètres** des comptes des utilisateurs, y compris leurs **mots de passes** ;
- ✓ **Créer ou supprimer** des **groupes d'utilisateurs** et modifier les **propriétés** des groupes existants.

Les opérations courantes sur les utilisateurs et les groupes sont :

- Créer un nouvel utilisateur,
- Définir un mot de passe,
- Créer de nouveaux groupes,
- Ajouter un utilisateur à un groupe,
- Modifier les paramètres utilisateur et groupe,
- Verrouiller un compte,
- Modifier l'expiration du mot de passe,
- Supprimer un compte et un groupe.

Les fichiers importants à connaître sont :

- ✓ Le fichier **/etc/passwd** (concernant les utilisateurs).
- ✓ Le fichier **/etc/group** (concernant les groupes).

### Le fichier `/etc/passwd` :

Le fichier `/etc/passwd` contient toutes les **informations relatives** aux utilisateurs (login, mots de passe, ...). Seul le super utilisateur (root) doit pouvoir le modifier.

Ce fichier possède un format spécial permettant de repérer chaque utilisateur, chacune de ses lignes possède le format suivant :

**nom\_du\_compte : mot\_de\_passe : numero\_utilisateur : numero\_de\_groupe  
: commentaire : dossier : programme\_de\_demarrage**

Sept (7) champs sont explicités séparés par le caractère ":" :

- ✓ Le nom du compte de l'utilisateur
- ✓ Le mot de passe de l'utilisateur (codé bien sûr)
- ✓ L'entier qui identifie l'utilisateur pour le système d'exploitation (UID=User ID, identifiant utilisateur)
- ✓ L'entier qui identifie le groupe de l'utilisateur (GID=Group ID, identifiant de groupe)
- ✓ Le commentaire dans lequel on peut retrouver des informations sur l'utilisateur ou simplement son nom réel
- ✓ Le dossier de connexion qui est celui dans lequel il se trouve après s'être connecté au système
- ✓ La commande est celle exécutée après connexion au système (c'est fréquemment un interpréteur de commandes)

Voici un exemple de fichier *passwd* :

- root:x:0:0:root:/root:/bin/bash
- bin:x:1:1:bin:/bin:/bin/bash
- daemon:x:2:2:daemon:/sbin:/bin/bash
- news:x:9:13:News system:/etc/news:/bin/bash
- uucp:x:10:14:./var/lib/uucp/taylor\_config:/bin/bash
- cquoi:x:500:100:Cool.....:/home/cquoi:/bin/bash

Lorsqu'un utilisateur se connecte, le programme login compare le mot de passe tapé par l'utilisateur (après l'avoir chiffré) à celui qui est dans le fichier passwd. S'ils sont différents, la connexion ne peut être établie.

**UID** : identifiant (unique) de chaque compte utilisateur.

Les nombres de 0 à 99 sont fréquemment réservés à des comptes propres à la machine.

Les valeurs supérieures à 100 sont réservées aux comptes utilisateurs.

**GID** : identifiant de groupe.

Le groupe par défaut (nommé group) porte le numéro 50.

Cet identifiant est utilisé en relation avec les droits d'accès aux fichiers.



## Le fichier `/etc/group` :

Le fichier `/etc/group` contient la **liste des utilisateurs** appartenant aux différents groupes.

En effet, lorsque de nombreux utilisateurs peuvent avoir accès au système, ceux-ci sont fréquemment rassemblés en différents groupes ayant chacun leurs propres droits d'accès aux fichiers et aux dossiers.

Il se compose de différents champs séparés par ":" :

**nom\_de\_groupe : champ\_special : numero\_de\_groupe : membre1, membre2**

Le champ spécial est fréquemment vide.

Le numéro de groupe est le numéro qui fait le lien entre les fichiers `/etc/group` et `/etc/passwd`

Voici un exemple de fichier `/etc/group` :

- root:x:0:root
- bin:x:1:root,bin,daemon
- daemon:x:2:
- wheel:x:10:
- mail:x:12:cyrus

- ✓ Un même utilisateur peut apparaître dans plusieurs groupes. Lorsqu'il se connecte au système, il appartient au groupe spécifié dans le fichier `/etc/passwd` (le champ GID). Il peut en changer à l'aide de la commande `newgrp`. Des droits d'accès aux fichiers sont alors définis.
- ✓ Pour ajouter un groupe, l'administrateur peut modifier le fichier `/etc/group` à l'aide d'un éditeur de texte.
  - Il peut également utiliser la commande `addgroup` ou `groupadd` (pas toujours présentes).
  - Dans le premier cas, il aura uniquement la ou les lignes correspondant aux groupes, à ajouter. Par exemple, la ligne :  
**admin : : 56 : ccm**

- ✓ Pour ajouter un utilisateur à un groupe, il suffit d'éditer le fichier `/etc/group` et de rajouter ce nom au bout de la ligne en séparant le nom des membres par une virgule.
- ✓ Pour supprimer un groupe, il suffit d'éditer le fichier `/etc/group` et d'effacer la ligne correspondante. Mais attention, il ne faut pas oublier de changer dans le fichier `/etc/passwd` les numéros (GID) du groupe supprimé, si des utilisateurs y appartenaient. Il est également essentiel de chercher les fichiers et dossiers de ce groupe pour le changer (dans le cas contraire les fichiers et dossiers risquent d'être inaccessibles).

Il faut ensuite créer le dossier de connexion personnel de l'utilisateur (home directory dont la spécification est ensuite inscrite par le système, lors de la connexion, dans la variable `#HOME`), l'en rendre propriétaire (**chown**), attribuer le dossier à un groupe, en général celui auquel appartient l'utilisateur (**chgrp**) et le protéger (**chmod 700**).

## Exemples : création des groupes et des utilisateurs

- Création de 2 groupes

```
groupadd gRT
groupadd gEII
```

```
cat /etc/group
```

```
gRT:x:1001:
gEII:x:1002:
```

- Création des 4 utilisateurs avec création de leurs dossiers home :

```
useradd -m u1
useradd -m u2
useradd -m u3
useradd -m u4
```

```
cat /etc/passwd
```

```
u1:x:1001:100::/home/u1:/bin/sh
u2:x:1002:100::/home/u2:/bin/sh
u3:x:1003:100::/home/u3:/bin/sh
u4:x:1004:100::/home/u4:/bin/sh
```

```
ls -l /home
```

```
drwxr-xr-x 2 u1 users 4096 2007-02-01 12:12 u1
drwxr-xr-x 2 u2 users 4096 2007-02-01 12:12 u2
drwxr-xr-x 2 u3 users 4096 2007-02-01 12:12 u3
drwxr-xr-x 2 u4 users 4096 2007-02-01 12:12 u4
```

- Placement des utilisateurs dans leurs groupes

```
usermod -G gRT u1
usermod -G gRT, gEII u2
usermod -G gEII u3
usermod -G gRT, gEII u4
```

```
cat /etc/group
```

```
gRT:x:1001:u1,u2,u4
gEII:x:1002:u2,u3,u4
```

- Changement de propriétaire des dossiers

```
chown u1: gRT /home/u1
chown u2: gRT /home/u2
chown u3: gEII /home/u3
chown u4: gEII /home/u4
```

- **Création des dossiers communs**

```
mkdir /home/Rgroup1  
mkdir /home/Rgroup2
```

- **Mise en place des permissions pour permettre aux utilisateurs d'écrire dans le dossier de leur groupe**

```
chgrp gRT /home/Rgroup1  
chgrp gEII /home/Rgroup2
```

- **Mise en place de la permission**

**chmod** permet de changer les permissions de **lecture**, d'**écriture** et d'**exécution** d'un fichier ou d'un dossier pour le propriétaire, son groupe et les autres.

### **Le principe**

Il y a trois (3) types de permission différentes, le droit de **lecture**, d'**écriture** et d'**exécution**. Et cela peut être défini indépendamment au propriétaire du fichier ainsi qu'à son groupe et pour tous les autres.

Il existe trois (3) syntaxes différentes pour données des droits, dans les trois (3) cas le résultat est le même.

Syntaxe :

```
chmod [options] 777 [fichier_ou_dossier]  
chmod [options] +rwx ugo [fichier_ou_dossier]  
chmod [options] rwx rwx rwx [fichier_ou_dossier]
```

```
chmod 756 [fichier_ou_dossier]
```

Le chiffre de gauche (7) donne les permissions au propriétaire (U)

Le chiffre du centre (5) donne les permissions au groupe (G)

Le chiffre de droite (6) donne les permissions à tous les autres utilisateurs (O)

Mais que signifie ces chiffres?

Chaque permission dispose d'une valeur différente.

La lecture = **4**

L'écriture = **2**

L'exécution = **1**

Par exemple pour donner le droit de lecture et d'exécution on va respectivement additionner  $4 + 1 = 5$

Et si on fait un "**chmod 555** [fichier\_ou\_dossier]" cela aura pour effet de mettre le droit de lecture et d'exécution au propriétaire, au groupe et aux autres.

Autre exemple si on veut donner tous les droits au propriétaire d'un fichier, donc la possibilité de lire, écrire et d'exécuter on va respectivement additionner  $4 + 2 + 1 = 7$ .

Et donner que le droit de lecture au groupe et aux autres, donc la lecture = **4**.

Cela donne comme commande final:

**chmod 744** [fichier\_ou\_dossier]

#### • Activation d'un utilisateur

**passwd u1**

Enter new UNIX password:

Retype new UNIX password:

passwd : le mot de passe a été mis à jour avec succès

**cat /etc/shadow**

**u1:\$1\$kiUUra9s\$AxchvKz0J9OBJPXO8qNf./:13545:0:99999:7:::**

## 2. Gestion des fichiers

### . Pour lire le contenu d'un fichier

La commande la plus simple pour lire le contenu d'un fichier est la commande **cat**.

```
$ cat fichier
```

cat affiche l'ensemble du fichier sur le terminal.

Les commandes **tail** et **head** permettent d'afficher respectivement la fin et le début des fichiers.

### . Pour trouver un fichier

Les deux commandes les plus courantes pour trouver un fichier sont **locate** et **find**.

locate est très rapide il recherche à partir d'un fichier d'index. Ce qui signifie qu'il ne voit pas immédiatement les nouveaux fichiers.

```
$ locate machin
```

find effectue une recherche dans un dossier spécifié. Il s'utilise principalement ainsi :

```
$ find [dossier] [-name nom]
```

(ce qui est entre [] est optionnel)

Il va rechercher dans le dossier les fichiers de nom. Il est possible d'appliquer une grande variété de condition pour n'afficher que les dossiers ou les fichiers d'un certain âge par exemple.

### . Pour supprimer un fichier

Pour supprimer un fichier, on utilise la commande **rm** (il est possible de donner plusieurs fichiers) :

```
$ rm fich
```

Supprime fich sans demander de confirmation, même si le fichier est protégé en écriture.

```
$ rm -f fich
```

Permet de demander à l'utilisateur une confirmation avant de supprimer le ou les fichiers en question.

### . Pour gérer les dossiers

Pour savoir dans quel dossier vous êtes, utilisez la commande **pwd** :

```
$ pwd
```

```
/home/monlogin
```

### **. Pour changer de dossier**

Il faut utiliser la commande `cd` suivie du chemin vers le dossier où vous souhaitez aller.

\$ **cd** /mon/dossier

Si on omet le dossier, on arrive dans le dossier personnel de l'utilisateur.

### **. Pour lister le contenu d'un dossier**

La commande `ls` pour lister le contenu des dossiers. Il est suivi du (ou des) dossier que l'on veut regarder, si on ne met pas de dossier ce sera le dossier courant.

\$ **ls** /dossier

### **. Pour créer/supprimer un dossier**

Pour créer un dossier on utilise la commande `mkdir` suivi du (ou des) dossier que l'on veut créer :

\$ **mkdir** /dossier

L'option `-p` permet de créer tous les dossiers parents qui n'existent pas encore.

---

### 3. Gestion des disques : partitions, formatage et montage de disques, profils

Le partitionnement est le fractionnement d'un support physique (disque dur, carte mémoire, clef USB) en plusieurs parties virtuelles, des partitions, destinées à accueillir un système de fichiers.

Le système de fichiers (en anglais file system, abrégé FS) est la façon dont le système d'exploitation structure les données sur le support physique (soit typiquement, le disque dur).

Comme tout système d'exploitation, GNU/Linux dispose de ses propres systèmes de fichiers :

- ✓ **ext4** : C'est le système de fichier le plus répandu sous GNU/Linux (issu de ext2 et ext3).
- ✓ **ReiserFS** : C'est un système de fichiers plus rapide que ext4 pour le traitement de dossiers contenant des milliers de fichiers de petite taille.

#### Nom des périphériques sous Linux

Le nom des disques et des partitions sous Linux diffèrent des autres systèmes d'exploitation.

Les noms du partitionnement utilisés lors de conventions de nommage :

- ✓ Le premier **CD-ROM SCSI** est nommé « **/dev/scd0** ».
- ✓ Le **disque maître** sur le **contrôleur IDE** primaire est nommé « **/dev/hda** ».
- ✓ Le **disque esclave** sur le **contrôleur IDE** primaire est nommé « **/dev/hdb** ».
- ✓ Les disques **maître** et **esclave** sur le second contrôleur sont nommés respectivement « **/dev/hdc** » et « **/dev/hdd** ».
- ✓ Le premier **disque SCSI** (selon l'identifiant SCSI) est nommé « **/dev/sda** ».
- ✓ Le second **disque SCSI** (selon l'identifiant) est nommé « **/dev/sdb** », ainsi de suite.

Supposons qu'il a deux disques SCSI, l'un à l'adresse SCSI 2 et l'autre à l'adresse 4.

Le premier disque principal est nommé « **sda** », et le second « **sdb** ».

Si le disque « **sda** » a 5 partitions, elles s'appelleront « **sda1** », « **sda2** », ..., « **sda5** ».

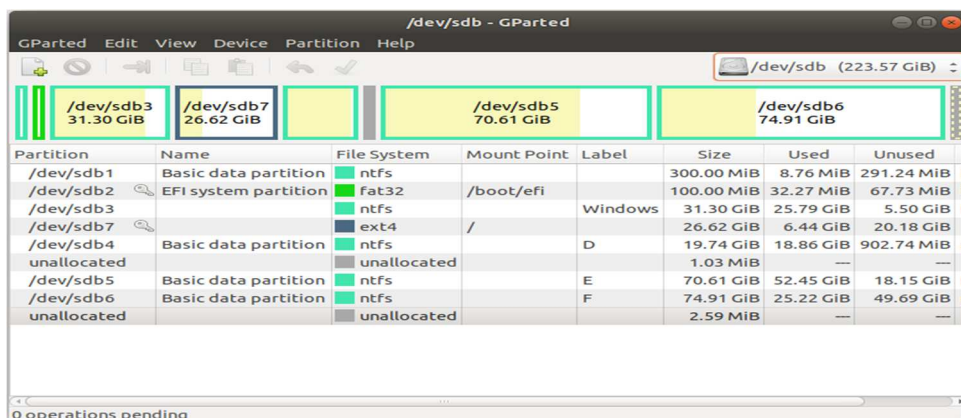
La même convention s'applique au disque « **sdb** » et ses partitions.

Linux représente les partitions primaires par le nom du disque suivi des nombres **1 à 4**.

Par exemple la première partition sur le premier **disque IDE** est **/dev/hda1**.

Les partitions logiques sont numérotées à partir de 5.

Donc, la première partition logique sur ce même disque est **/dev/hda5**.



| Partition   | Name                 | File System | Mount Point | Label   | Size       | Used      | Unused     |
|-------------|----------------------|-------------|-------------|---------|------------|-----------|------------|
| /dev/sdb1   | Basic data partition | ntfs        |             |         | 300.00 MiB | 8.76 MiB  | 291.24 MiB |
| /dev/sdb2   | EFI system partition | fat32       | /boot/efi   |         | 100.00 MiB | 32.27 MiB | 67.73 MiB  |
| /dev/sdb3   |                      | ntfs        |             | Windows | 31.30 GiB  | 25.79 GiB | 5.50 GiB   |
| /dev/sdb7   |                      | ext4        | /           |         | 26.62 GiB  | 6.44 GiB  | 20.18 GiB  |
| /dev/sdb4   | Basic data partition | ntfs        |             | D       | 19.74 GiB  | 18.86 GiB | 902.74 MiB |
| unallocated |                      | unallocated |             |         | 1.03 MiB   | —         | —          |
| /dev/sdb5   | Basic data partition | ntfs        |             | E       | 70.61 GiB  | 52.45 GiB | 18.15 GiB  |
| /dev/sdb6   | Basic data partition | ntfs        |             | F       | 74.91 GiB  | 25.22 GiB | 49.69 GiB  |
| unallocated |                      | unallocated |             |         | 2.59 MiB   | —         | —          |

Figure 7 : Partitionnement de disque

#### 4. Gestions d'amorçage : LILO et Grub

Un chargeur d'amorçage (ou bootloader) est un logiciel permettant de lancer un ou plusieurs systèmes d'exploitation (multiboot), c'est-à-dire qu'il permet d'utiliser plusieurs systèmes, à des moments différents, sur la même machine.

Par exemple, si on dispose d'une partition Linux et d'une Windows, c'est ce programme qui va permettre de choisir l'un ou l'autre système au démarrage de votre PC.

Les chargeurs d'amorçage les plus usuels sont :

- ✓ **GRUB** (GRand Unified Bootloader),
- ✓ **LILO** (Linux loader) pour le BIOS et elilo pour EFI (voir (en) elilo),
- ✓ **NTLDR** (NT LoaDeR ou Chargeur d'amorçage de Windows NT),
- ✓ **IsoLinux** de Syslinux pour booter à partir d'un DVD ISO 9660,
- ✓ **PXELinux** de Syslinux pour booter à partir d'une carte réseau.

- **LILO (Linux LOader)**

LILO est un chargeur d'amorçage utilisé dans les systèmes d'exploitation Linux. Il peut démarrer (jusqu'à 16) systèmes d'exploitation à partir d'un disque dur, clé USB, etc., car il ne dépend pas d'un système de fichiers spécifique.

LILO a été utilisé comme chargeur de démarrage par défaut sous Linux jusqu'à la fin de 2001.

Il est maintenant inclus dans la liste des packages dépréciés (dans Red Hat).

- **GRUB (GNU GRand Unified Bootloader)**

GRUB est un chargeur d'amorçage développé par le projet GNU.

Il le chargeur de démarrage par défaut utilisé dans la plupart des distributions Linux aujourd'hui.

Il peut être configuré de manière dynamique car il permet de modifier la configuration au moment du démarrage.

Les utilisateurs disposent d'une interface de ligne de commande simple pour insérer de nouvelles configurations de démarrage de manière dynamique.

LILO et GRUB sont deux types de bootloader (« chargeurs d'OS ») utilisés dans le système d'exploitation Linux. LILO n'est utilisé que pour Linux, tandis que GRUB peut être utilisé pour un système d'exploitation autre que Linux.

#### 5. Gestion des processus : planification de tâches, les signaux

Les deux commandes les plus couramment utilisées pour visualiser les processus sont :

- ✓ **top**,
- ✓ **ps**.

La différence entre les deux est que **top** est utilisé de manière interactive dans un terminal et que **ps** est plutôt utilisé dans les scripts, combiné avec d'autres commandes bash.



- **top**

Cette commande est probablement la plus basique.

Les usages utiles :

- Pour **afficher** les processus d'un utilisateur en particulier, vous pouvez utiliser: top -u utilisateur
- Pour **tuer/arrêter** un processus en cours d'exécution, à trouver le PID du processus que à tuer et à appuyer sur k.
- pour **enregistrer** les paramètres actuels de la commande top en utilisant le raccourci clavier Shift + W.

Avec la commande top, on peut également utiliser certaines options, telles que:

- **-d** (delay) : pour spécifier l'intervalle de rafraîchissement à la place du délai.
- **-n** (number) : pour rafraîchir un certain nombre de fois, puis quitter top.
- **-p** (pid) : pour afficher uniquement et surveiller les processus qui ont un id de processus ( pid ) particulier.
- **-q** : rafraîchir sans délai.

- **ps**

Une autre commande très utile pour afficher les processus sous Linux.

Les options fréquemment utilisées avec la commande ps :

- **-e** : Affiche tous les processus.
- **-f** : Listing complet.
- **-r** : Affiche uniquement les processus en cours d'exécution.
- **-u** : Possibilité d'utiliser un nom d'utilisateur (ou plusieurs) en particulier.
- **-C** : Filtrer les processus par leur nom ou leur commande.
- **-o** : Affiche les informations associées à une liste de mots-clés séparés par des virgules ou des espaces.

## **Tuer et hiérarchiser les processus**

### **La commande kill**

Par exemple :

**kill pid**

Ici, au lieu du PID, on peut entrer l'ID du processus qu'on veut tuer. Si le processus ne veut pas s'arrêter, on peut utiliser: kill -9 pid.

## IV. CONFIGURATION DES SERVICES

### 1. Réseau

#### 1.1. Notion de configuration d'une interface Ethernet

L'essentiel de la configuration d'un réseau peut être fait en passant par le fichier de configuration **interfaces** du dossier **/etc/network/interfaces**.

Là, on peut donner à une carte de réseau une adresse IP (ou utiliser dhcp), configurer les informations de routage ou le masquage d'IP, le routage par défaut et bien d'autres choses.

Il faudrait ajouter à la ligne **'auto'** les interfaces que l'on souhaite lancer au démarrage de l'ordinateur.

Pour plus d'informations.

Voir **man interfaces**.

#### ✓ Démarrer et stopper des interfaces

Les interfaces configurées avec **/etc/network/interfaces** peuvent être **activées** et **désactivées** avec les commandes **ifup** et **ifdown**.

Quelques guides surannés enseignent de redémarrer le service réseau pour appliquer les changements à **/etc/network/interfaces**.

Cependant, cela est en effet obsolète car il est possible que toutes les interfaces ne soient pas redémarrées.

A Utiliser plutôt **ifup** et **ifdown** pour appliquer les changements pour chaque interface.

Par exemple, avec une interface nommée **enp7s0** :

```
# ifdown enp7s0
# ifup enp7s0
```

#### ✓ Noms d'interface réseau

Depuis Stretch, les nouveaux systèmes n'utilisent par défaut désormais plus les anciens noms d'interface tels que :

**eth0, eth1, wlan0, wlan1.**

Le nouveau système utilise des noms basés sur les emplacements du matériel tels que :

**no0, enp0s31f6, wlp1s7.**

On peut lister les interfaces avec : **ls /sys/class/net.**

Divers exemples plus bas continuent d'utiliser « **eth0** » comme nom d'interface par défaut, même s'il est peu probable que cela existe sur un système moderne.

## 1.2. Installation de la carte réseau

Les cartes réseau sont souvent détectées au démarrage. Si ce n'est pas le cas il faudra charger les modules correspondants.

Pour obtenir la liste des interfaces réseau qui ont été détectées, on peut utiliser dans l'invite de commandes :

**ifconfig -a**

Les sections qui commencent par ethX correspondent aux cartes Ethernet, où X est le numéro de la carte.

Si la carte n'est pas détectée, il faudra charger le module avec la commande

**modprobe <nom du module>**

Parmi les modules courants on peut noter : **ne2k-pci** pour les cartes NE2000, via-rhine, rtl8139...

## 1.3. Configuration de l'interface réseau

Une fois votre carte reconnue par le noyau, vous pouvez au moins préciser son adresse IP et son masque de sous-réseau.

Dans le cas d'un réseau local connecté à l'internet, vous devez aussi ajouter l'adresse IP de la passerelle et l'adresse IP d'un ou plusieurs serveurs DNS.

### ✓ Adresse IP

Pour attribuer une adresse IP à une interface réseau, on peut utiliser la commande **ifconfig** :

**ifconfig <interface> <adresse ip>**

Par exemple :

**ifconfig eth0 192.168.0.1**

Le **masque de sous-réseau** est déterminé automatiquement en fonction de la **classe de l'adresse IP**.

S'il est différent on peut le spécifier avec l'option netmask :

**ifconfig eth0 192.168.0.1 netmask 255.255.255.0**

Pour voir si la carte réseau est bien configurée, on peut utiliser la commande :

**ifconfig eth0** ou **ip a**

### ✓ Passerelle et routage

Pour ajouter une passerelle, on peut utiliser la commande route :

**route add default gw <adresse ip>**

Pour afficher les routes vers les différents réseaux :

**route -n**

### ✓ Tester le réseau

Pour vérifier que la carte réseau fonctionne, on peut essayer de communiquer avec une autre machine avec la commande

**ping <adresse ip>**

La commande ping envoie un paquet à l'adresse IP puis attend que la machine réponde. Elle affiche ensuite le temps qu'a pris toute l'opération, en millisecondes.

### ✓ Informations sur les interfaces

Pour vérifier les ports ouverts, on peut utiliser la commande

**netstat -a**

### ✓ Nom d'hôte (hostname)

Le fichier **/etc/hostname** contient le nom de la machine. Il suffit de l'éditer pour changer le nom d'hôte de la machine. Cette modification n'est pas prise en compte immédiatement par le système, elle le sera au prochain démarrage de la machine ou après avoir lancé :

On peut également changer le nom d'hôte avec la **commande** suivante, mais il ne sera pas conservé au prochain démarrage :

**hostname <nom d'hôte>**

## 1.4. Configuration automatique au démarrage

Le fichier **/etc/network/interfaces** permet de configurer les cartes réseau de manière permanente.

Par exemple :

```
auto lo
iface lo inet loopback
```

```
auto eth0 ou (allow-hotplug eth0)
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
gateway 192.168.0.1
dns-nameservers 8.8.8.8
```

Avec cette configuration, le serveur demeure après redémarrage

Cette configuration initialisera automatiquement les interfaces **"lo"** et **"eth0"**.

L'interface **"lo"** est souvent indispensable au système, il est important de l'initialiser. Elle aura systématiquement l'adresse IP 127.0.0.1.

L'interface **"eth0"** sera configurée avec l'adresse IP 192.168.0.1, le masque de sous-réseau 255.255.255.0 et la passerelle 192.168.0.1 (ce paramètre est facultatif).

NB :

Démarrer le service : **systemctl restart networking.service**

Voir le statut : **systemctl status networking.service**

Dans le cas d'une IP fixe, il vaut mieux renseigner un serveur DNS (ci-dessus celui de Google). Sinon, si l'interface eth0 doit être configurée automatiquement par un serveur **DHCP**, il faut indiquer :

```
auto eth0
iface eth0 inet dhcp
```

Pour que les modifications de ce fichier soient prises en compte, il faut redémarrer ou utiliser les commandes ifup et ifdown.

Par exemple :

```
ifup eth0
```

#### Démarrage de services :

```
Systemctl start networking.service
Systemctl status networking.service
```

### 1.5. Configuration WIFI

Le WIFI (protocole 802.11) est une technologie de réseaux locaux sans fil.

Il existe deux protocoles courant de cryptage en WIFI: **WEP** et **WPA**.

L'**ESSID** est le nom du réseau sans fil.

Le fichier interfaces configuré avec une clef de cryptage WEP.

```
# cat /etc/network/interfaces
auto lo
iface lo inet loopback
```

L'interface **eth1** correspond ici à la carte wifi

```
auto eth1
iface eth1 inet dhcp
wireless-essid mon_essidwireless-mode managed
wireless-key AF32852BE7A39B522BG60C4353
```

L'ESSID et la clef WEP doivent correspondre et être correctement configurés sur le serveur sans fil.

### 1.6. Résolution de noms d'hôte

#### ✓ Fichier hosts

Le fichier **/etc/hosts** contient une liste de résolutions de noms (adresses IP et noms de machine).

Par exemple :

```
192.168.0.1 dcutm
192.168.0.2 MXutm
```

Ce fichier indique que **dcutm** correspond à l'adresse IP 192.168.0.1, qui sera accessible par cet alias.

## 1.7. Serveurs DNS

Le fichier de configuration **resolv.conf** du dossier **/etc/resolv.conf** contient les informations qui permettent à un ordinateur connecté au réseau de convertir les **noms en adresses**.

Le fichier **resolv.conf** contient habituellement les adresses IP des **serveurs de noms de domaine** (DNS) qui tenteront de convertir les noms en adresse pour tous les nœuds disponibles sur le réseau.

Par exemple :

```
search utm.bf  
nameserver 192.168.0.1  
nameserver 192.168.0.254
```

La commande **search** indique que si un nom de domaine n'est pas trouvé, il faudra essayer en lui ajoutant utm.bf

## 2. DNS : BIND

### 2.1. Définition

BIND est le serveur DNS le plus utilisé sur **Internet**, spécialement sur les systèmes de type UNIX et est devenu de facto un standard.

Le **service DNS** (Domain Name System) permet la correspondance entre un nom de domaine qualifié (**FQDN** : Fully Qualified Domain Name) et une adresse IP, par exemple `www.som.bf = 175.33.220.10`.

Ainsi, grâce à DNS, il n'est pas nécessaire de se souvenir des adresses IP.

Un serveur qui héberge le service DNS est appelé « **serveur de noms** ».

Cette partie du cours est destinée à apprendre comment configurer et maintenir un serveur **DNS BIND9**.

BIND9 peut être utilisé de différentes façons :

#### ✓ **Serveur cache**

Dans cette configuration, BIND9 va effectuer les requêtes DNS et se rappeler de la réponse pour la prochaine requête.

#### ✓ **Serveur primaire**

Utilisé pour contenir les enregistrements DNS d'un nom de domaine enregistré. Un ensemble d'enregistrements DNS pour un nom de domaine est appelé une "zone".

#### ✓ **Serveur secondaire**

Un serveur secondaire est utilisé en complément à un serveur primaire. Cela assure la disponibilité de la zone DNS, même si le serveur primaire est hors ligne.

#### ✓ **Serveurs hybrides**

Un serveur BIND9 peut être configuré à la fois comme serveur cache et comme serveur primaire, comme serveur cache et serveur secondaire, ou même serveur cache, serveur primaire et secondaire.

#### ✓ **Serveurs Récursifs / Non récursifs**

Les serveurs BIND9 peuvent être récursifs, c'est-à-dire interroger tour à tour les serveurs DNS nécessaires jusqu'à obtenir la réponse, et la transmettre à leur client.

Dans le cas contraire (par défaut), le serveur DNS délègue la résolution du nom de domaine à un autre serveur DNS.

## 2.2. Enregistrements DNS

Il existe de nombreux type d'enregistrements DNS, mais certains sont plus communs :

### ✓ Enregistrement de type A (Address)

C'est le type le plus courant.

Cet enregistrement fait correspondre une **adresse IP** à un **nom** de machine.

```
dcutm    IN    A    192.168.0.1
```

### ✓ Enregistrement de type CNAME (Alias)

Utilisé pour créer un alias depuis un enregistrement de type A.

Il est possible de créer un enregistrement de type CNAME qui pointe vers un autre enregistrement CNAME, mais ceci double le nombre de requêtes qui seront faites au serveur de noms. Cette méthode est donc déconseillée.

```
www      IN    CNAME  dcutm.utm.bf.
```

### ✓ Enregistrement MX (Mail Exchange)

Utilisé pour définir vers quel serveur de la zone un email à destination du domaine doit être envoyé, et avec quelle priorité.

Cet enregistrement doit pointer vers un enregistrement de type A, et non un alias CNAME. Il peut y avoir plusieurs enregistrements MX s'il existe plusieurs serveurs de messagerie sur le domaine. Le plus petit nombre a la plus grande priorité.

```
; Enregistrements "MX"
```

```
@      IN    MX  10  dcutm.utm.bf.
```

```
@      IN    MX  20  MXutm.utm.bf.
```

### ✓ Enregistrement NS (Name Server)

Utilisé pour définir quels **serveurs** répondent pour cette zone.

Cet enregistrement doit **pointer** vers un **enregistrement de type A**, non pas vers un enregistrement de type CNAME.

C'est ici que le serveur maître et les esclaves sont définis.

```
@      IN    NS    dcutm.utm.bf.
```

```
@      IN    NS    MXutm.utm.bf.
```

```
dcutm    IN    A    192.168.0.1
```

```
MXutm    IN    A    192.168.0.2
```



### 2.3. Fichiers de configuration

Les fichiers de configuration sont placés dans le dossier **/etc/bind/** :

- ✓ Fichier **/named.conf**

Ce fichier est le fichier de **configuration principal** du serveur DNS.

- ✓ Fichier **/named.conf.default-zones**

Ce fichier contient des zones par défaut :

- forward,
- reverse,
- broadcast.

- ✓ Fichier **/named.conf.options**

Ce fichier contient l'ensemble des options de configuration du serveur DNS.

- ✓ Fichier **/named.conf.local**

Ce fichier contient la configuration locale du serveur DNS, on y déclare les zones associées au domaine.

- ✓ Fichier **/named.conf.log**

Ce fichier est à créer dans **/etc/bind**

### 2.4. Installation de BIND9

**Installation des paquets :**

```
sudo apt-get install bind9
sudo apt-get install bind9utils
sudo apt-get install bind9-doc
ou
sudo apt-get install bind9 bind9utils bind9-doc
```

### 2.5. Configurations des zones

- ✓ **Définition de la zone**

A éditer le fichier **/etc/bind/named.conf** et à déclarer les deux (2) zones:

- **Définition de la zone directe**

```
zone "utm.bf" IN {
    type master;
    file "/etc/bind/db.utm.bf";
};
```

- **Définitions de la zone reverse**

```
zone "0.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/db.0.168.192.in-addr.arpa";
};
```

### ✓ Configuration de la zone directe

Création et édition de **db.utm.bf** dans /etc/bind/

On fait correspondre ici un **nom** à une adresse **IP**.

Ainsi, **dcutm.utm.bf** aura l'adresse 192.168.0.1

```
$TTL 604800
@ IN SOA dcutm.utm.bf. admin.utm.bf. (17032304 604800 86400 2417200 604800)

@ IN NS dcutm.utm.bf.
@ IN NS MXutm.utm.bf.
dcutm IN A 192.168.0.1
MXutm IN A 192.168.0.2
www IN CNAME dcutm.utm.bf.
@ IN MX 10 MXutm.utm.bf.
```

### ✓ Configuration du reverse

Création et édition de **db.0.168.192.in-addr.arpa** dans /etc/bind/

Ici on fait correspondre l'**IP** au **nom**.

Ainsi, à partir des adresses : 192.168.0.1 et 192.168.0.2 on obtient les **noms** des machines respectives sur le domaine utm.bf.

```
$TTL 1800
@ IN SOA dcsom.som.bf. admin.som.bf. (3 14400 3600 604800 10800)

@ IN NS dcutm.utm.bf.
@ IN NS MXutm.utm.bf.
dcsom IN A 192.168.0.1
MXsom IN A 192.168.0.2
1 IN PTR dcutm.utm.bf.
2 IN PTR MXutm.utm.bf.
```

### ✓ Configuration des options

Dans le fichier /etc/bind/named.conf.options

```
options {
    directory "/var/cache/bind";
    allow-recursion { any; };
    forwarders { 8.8.8.8; };
    dnssec-enable no;
    dnssec-validation no;
    auth-nxdomain no; # conform to RFC1035
    listen-on-v6 { any; };
    listen-on { any; };
    allow-query { any; };
};
```

## 2.6. Vérifications et outils de tests

### ✓ Statut du service

```
# systemctl status bind9
# service bind9 reload
```

### ✓ Journaux

Les journaux permettent d'afficher les informations renvoyées par le serveur :

```
# journalctl -f -u bind9
```

Enlever -f pour un affichage normal (less).

-f : permet d'avoir le même comportement que tail -f

### ✓ Tests

```
# named-checkconf -z
# named-checkzone utm.bf /etc/bind/db.utm.bf
zone utm.bf /IN: loaded serial 17032304
```

Si OK = OK

```
# named-checkzone som.bf /etc/bind/db.0.168.192.in-addr.arpa
zone som.bf /IN: loaded serial 3
```

Si OK = OK

La commande **dig**:

```
#dig @192.168.0.1 dcutm.utm.bf
```

La commandes **nslookup** :

```
# nslookup www
```

```
# nslookup
> set type=soa
> utm.bf
```

```
# nslookup
> set type=ns
> utm.bf
```

```
# nslookup
> set type=a
> dcutm.utm.bf
```

### 3. Serveur web (HTTP)

#### 3.1. Définition

Un serveur web est un logiciel qui permet de répondre et de renvoyer des pages HTML à partir d'un navigateur afin de consulter un site web.

Il existe plusieurs serveurs web sous Linux, mais les principaux concurrents sont :

- ✓ **Apache httpd,**
- ✓ **Nginx,**
- ✓ **Lighttpd.**

Le serveur web qui sera étudié est **Apache httpd** mais souvent appelé **Apache**.

Apache existe depuis plus de 20 ans et même reste le serveur web le plus utilisé au monde.

Apache est alors le serveur web le plus connu et le plus répandu. Mais il en existe d'autres qui peuvent le concurrencer dans certains cas d'usage.

Par exemple, lorsqu'il s'agit de servir des fichiers statiques ou d'agir en serveur mandataire (proxy), il est judicieux de s'intéresser à des alternatives comme : **Nginx** et **lighttpd**.

Avec Apache, un administrateur peut configurer un serveur web pour héberger plusieurs domaines ou sites à partir d'une seule interface ou IP en utilisant un système de correspondance.

#### 3.2. Présentation de l'environnement

Le serveur HTTP Apache2 est un programme modulaire. Mise à part quelques modules directement intégrés dans le **programme binaire httpd**, l'administrateur peut choisir les fonctionnalités qu'il souhaite en activant des modules.

De même, il existe plusieurs fichiers de configuration tous présents dans `/etc/apache2/` :

- Le fichier de configuration principal est : **apache2.conf**  
Il contient les **paramètres généraux** et **communs** à tous les serveurs et plusieurs "Include" vers les autres fichiers.
- Le fichier de configuration : **ports.conf**  
Il contient la liste des ports en écoute.
- Les fichiers concernant les **modules** dans le dossier : `/etc/apache2/mods-available/`.

On y trouve deux catégories de fichiers : **\*.load** et **\*.conf**

- Les fichiers avec l'**extension load** « charge » effectivement les **modules dynamiques** :

**userdir.load**

- Les fichiers avec l'**extension conf** sont les fichiers de **configuration** des modules:

**userdir.conf**

- On trouve tous les **fichiers de configuration** des **serveurs web** `/etc/apache2/sites-available/`

- On trouve les **fichiers de configuration** des **sites web activés** dans le dossier `/etc/apache2/sites-enabled/` : ce sont uniquement ces fichiers qui sont inclus dans le fichier de configuration principal par la directive : `Include /etc/apache2/sites-enabled/[^.#]*`

Et ces fichiers sont en fait des **liens** qui **pointent** vers les fichiers de `/etc/apache2/sites-available`

pour **activer un site**, il existe une commande :

**a2ensite fichier\_conf.**

"fichier\_conf" étant un fichier de configuration présent dans `/etc/apache2/sites-available/`

La documentation est dans `/usr/share/doc.`

Les journaux sont dans `/var/log/apache2/`.

Le script de lancement du service serveur est dans `/etc/init.d`

### 3.3.Principe de configuration d'Apache :

La configuration d'Apache est dans le dossier `/etc/apache2/`.

La configuration est découpée en un grand nombre de fichiers.

Le fichier principal se nomme : **apache2.conf** .

#### ✓ La conception **modulaire**

La conception d'Apache est modulaire et permet d'ajouter ou d'enlever à la volée les fonctionnalités nécessaires.

Ainsi, dans le dossier **mods-available**, il y a trois (3) modules du type **MPM\_\*.load** .

Ces (MPM) **MultiProcessing Modules** définissent la manière dont Apache va gérer les connexions HTTP.

Pour fonctionner, toutefois Apache a besoin qu'**un et un seul** de ces modules soit chargé :

- **prefork** : avec ce module, Apache crée un processus fils pour chaque nouvelle connexion. C'est le plus sûr mais pas toujours le plus performant car il est long et gourmand en mémoire de gérer autant de processus.
- **worker** : avec ce module, chaque processus fils d'Apache peut gérer plusieurs connexions dans des "*threads*". C'est généralement mieux en terme de performances mais ça peut poser des problèmes avec les applications web qui gèrent mal ces threads.
- **event** : c'est le module le plus récent et celui choisi par défaut sous Ubuntu. C'est une évolution de worker qui permet de mieux gérer les connexions "*keepalive*". Ces connexions permettent de faire plusieurs requêtes HTTP à travers une même connexion réseau.

NB : il est conseillé de garder le MPM "event".

## ✓ Les fichiers de configuration :

- Le fichier de configuration **ports.conf** :  
**Listen 80** (plus toujours nécessaire)

Elle indique à Apache d'écouter sur toutes les interfaces sur le port 80 (port par défaut pour HTTP).

- Les fichiers de configuration placés dans :
  - **conf-enabled/**
  - **mods-enabled/**
  - **sites-enabled/**

## Les principes de fonctionnement :

**conf-available/** : les fichiers de configurations additionnelles

**mods-available/** : les modules installés et leurs configurations

**sites-available/** : les configurations des différents sites

Pour chacun de ces fichiers, il faut activer la **fonction correspondante** en faisant un lien symbolique du type :

**XXXX-enabled/YYYY.conf -> XXXX-available/YYYY.conf**

Pour **activer/désactiver** une fonction, il suffit de **faire/supprimer** des **liens symboliques**.

- la configuration de **virtual host**

La configuration d'un virtual host est comprise dans des balises "type HTML"

**<VirtualHost \*:80>**

**</VirtualHost>** .

Le \*:80 indique que le virtual host peut être utilisé sur le port 80 sur toutes les interfaces.

Les détails des directives utilisées :

- **ServerName** : précise l'hôte pour lequel cette configuration sera utilisée.
- **ServerAlias** : précise les ServerAlias séparés par des espaces.
- **ServerAdmin** : indique une adresse mail de contact.
- **DocumentRoot** : indique la racine de l'arborescence de site web.

Une configuration minimale pourrait s'arrêter là ; les **autres directives** sont des **options supplémentaires** :

- **ErrorLog** : demande de stocker aussi les logs d'erreur dans un fichier à part mais leur format est **fixe**.
- **CustomLog** : demande de stocker les fichiers de logs au format **combined** dans un fichier à part plutôt que dans le fichier de logs général
- **<Directory />** : dans la balise Directory, on définit des règles qui s'appliquent uniquement au contenu d'un **dossier** dans apache2.conf.

une partie des **options activées** :

- **ExecCGI** : l'exécution de scripts à l'aide du module CGI est permise
- **FollowSymlinks** : le serveur va suivre les *liens symboliques*.
- **Includes** : les inclusions côté serveur à l'aide du module *mod\_include* sont autorisées
- **Indexes** : si aucun fichier par défaut type *index.html* n'est présent, le module *mod\_autoindex* va présenter une *liste des fichiers et dossiers* formatée par Apache
- **AllowOverride None** : indique qu'aucune option ne peut être surchargée par les options qui seraient contenues dans un fichier appelé *.htaccess* placé dans cette arborescence.

On pu préciser une à une les options qui peuvent être surchargées ou utiliser **All**  
All : dans ce cas, les options du fichier *.htaccess* auront la priorité sur celles du Virtual Host.

### 3.4. Installation d'un serveur Apache2

Apache2 a très certainement été installé par défaut lors de l'installation de Debian.

Pour le vérifier : **dpkg -l | grep apache2**

```
apache2
apache2-common
apache2-mpm-prefork
apache2-utils
libapache2-mod-perl2
libapache2-mod-php4 (ou libapache2-mod-php5)
```

NB : apache2 est installé suit à la vérification par **| grep**

Si Apache2 n'est pas installé, la commande :

**apt-get install apache2** (installera le serveur web avec ses dépendances).

**Ou**

**sudo apt-get update**

**sudo apt-get install apache2**

On aura certainement besoin par la suite du module php alors autant l'installer tout de suite:

**apt-get install libapache2-mod-php4 (ou apt-get install libapache2-mod-php5)**

Pour installer la documentation en local :

**apt-get install apache2-doc**

### 3.5. Configuration d'hôtes virtuels

Apache Virtual Hosts permet **d'héberger plusieurs domaines** sur une même machine.

Un hôte virtuel est une identité (supplémentaire) assumée par le serveur web.

Apache distingue deux (2) types d'hôtes virtuels :

- Se basant sur l'**adresse IP** (ou le port) ;
- Se basant sur le **nom DNS** du serveur web.

La première méthode nécessite **une adresse IP différente pour chaque site** tandis que la seconde n'emploie qu'**une adresse IP** et différencie les sites par le nom d'hôte communiqué par le client HTTP.

La configuration **par défaut** d'Apache 2 exploite les hôtes virtuels **basés sur le nom**.

#### Exemple de configuration :

Sur les systèmes Debian, les fichiers de configuration Apache Virtual Hosts se trouvent dans le dossier `/etc/apache2/sites-available` et peuvent être activés en créant des liens symboliques vers le dossier `/etc/apache2/sites-enabled`.

De plus, un **hôte virtuel par défaut** a été défini dans le fichier :

```
/etc/apache2/sites-enabled/000-default.conf
```

Cet hôte virtuel sera employé si aucun autre hôte virtuel ne correspond à la requête du client.

Le **premier hôte virtuel** défini répondra systématiquement aux requêtes concernant des hôtes virtuels inconnus.

Chaque **hôte virtuel supplémentaire** est ensuite décrit par un fichier placé dans le dossier `/etc/apache2/sites-available/`

```
/etc/apache2/sites-available# cp 000-default.conf utm.conf
```

```
<VirtualHost *:80>
```

```
ServerName      dcutm.utm.bf
ServerAlias     www.utm.bf
DocumentRoot    /var/www/html/utm
```

```
</VirtualHost>
```

Ainsi, la mise en place du domaine **utm.bf** se résume à créer le fichier ci-dessous, puis à l'activer avec : `/etc/apache2/sites-available# a2ensite utm.conf`



### 3.6. Configuration des fichiers de log :

L'analyseur de **logs** est un compagnon fréquent du serveur web puisqu'il permet aux administrateurs d'avoir **une idée plus précise** de l'usage fait de ce service.

Le serveur Apache peut être configuré pour n'utiliser qu'un seul fichier de log pour tous les hôtes virtuels.

Mais, on pourrait changer en intégrant des directives **CustomLog**.

Il est donc nécessaire de personnaliser le format de ce fichier pour y intégrer le nom de l'hôte virtuel.

Pour cela, on ajoutera un fichier :

**/etc/apache2/conf-available/customlog.conf** définissant un nouveau format (directive **LogFormat**)

et on l'activera avec **a2enconf customlog**.

Ou il faut supprimer (ou passer en commentaire) la ligne **CustomLog** du fichier **/etc/apache2/sites-available/000-default.conf**.

### 3.7. Configuration de directives courantes

Cette section passe en revue quelques-unes des directives de configuration d'Apache les plus usitées.

Le fichier de configuration principal contient habituellement plusieurs blocs **Directory** destinés à **paramétrer le comportement du serveur** en fonction de l'**emplacement** du fichier servi dans **apache2.conf**.

#### Exemple Bloc Directory :

```
<Directory /home/httpd/html>
# Options possibles : "None", "All", ou plusieurs combinaisons de:
# "Indexes", "Includes", "FollowSymLinks", "ExecCGI", ou "MultiViews".
Options Indexes Includes FollowSymLinks
# AllowOverride = All pour donner la priorité aux fichiers .htaccess
AllowOverride All
order allow,deny
# allow from = all pour permettre à tout le monde d'accéder aux documents
allow from all
</Directory>
```

```
<Directory /var/www/html/som >
    Options Indexes FollowSymlinks
    AllowOverride All
    DirectoryIndex index.php index.html index.htm
</Directory>
```

## V. OUTILS D'ADMINISTRATION

Le travail en tant qu'**Administrateur** système peut inclure les tâches suivantes : Analyser des enregistrements système et identifier des problèmes potentiels liés aux ordinateurs du réseau local ou distant. Installer les mises à jour du système d'exploitation, les correctifs et les modifications de la configuration.

L'**Administrateur** est alors souvent confronté au défi de l'administration des serveurs. Et des outils viennent apporter un plus gérer le processus de démarrage de *Linux* ou autre.

En tant qu'administrateur système, il n'est plus question de travailler tout simplement sur le système d'exploitation principal (Linux ou Windows), mais occuper également de ce qui y est hébergé. Il peut s'agir d'une base de données, d'un serveur d'applications, d'un serveur Web, d'applications de messagerie, de mise en cache, etc.

Les outils suivants devraient pouvoir aider à bien des égards.

### 1. Webmin

L'outil Webmin offre une interface Web pour administrer tous les aspects d'un serveur Linux. Du partage de fichiers au DNS, en passant par le serveur Web Apache et diverses bases de données.

Si on a besoin de quelque chose qui n'est pas inclus dans le package par défaut, on peut en choisir un dans un vaste catalogue de modules tiers qui apportent des fonctionnalités supplémentaires.

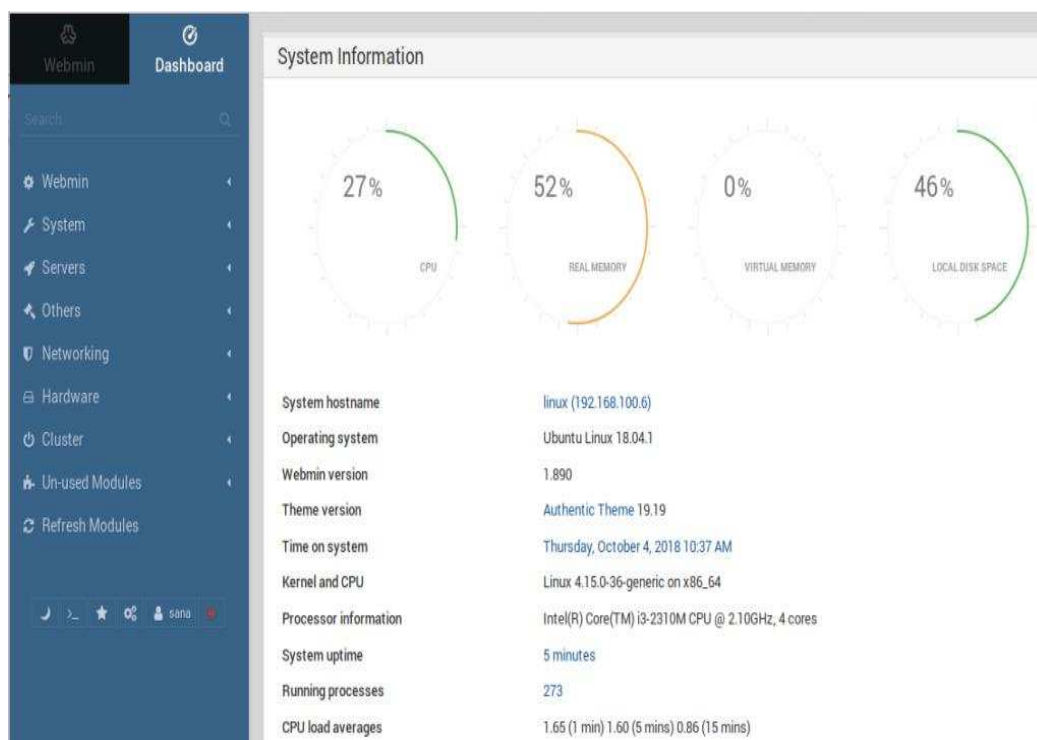


Figure 8: Webmin

Il suffit de télécharger la dernière version et de la copier dans le dossier d'accueil du serveur. Après cela, on exécute simplement la commande : `dpkg -i webmin_(version).deb`.

## 2. Poste de pilotage

Poste de pilotage est un autre outil d'administration de serveur qui offre une interface graphique Web qui facilite les **tâches d'administration** du **stockage**, le **démarrage / l'arrêt des services**, l'**inspection du journal** et la gestion de la configuration **multi-serveurs**.

L'outil a été créé pour Red Hat, mais il fonctionne sur de nombreuses distributions de serveurs Linux, en plus de Red Hat Enterprise Linux (RHEL): Fedora, Arch Linux, Ubuntu, CentOS, entre autres.

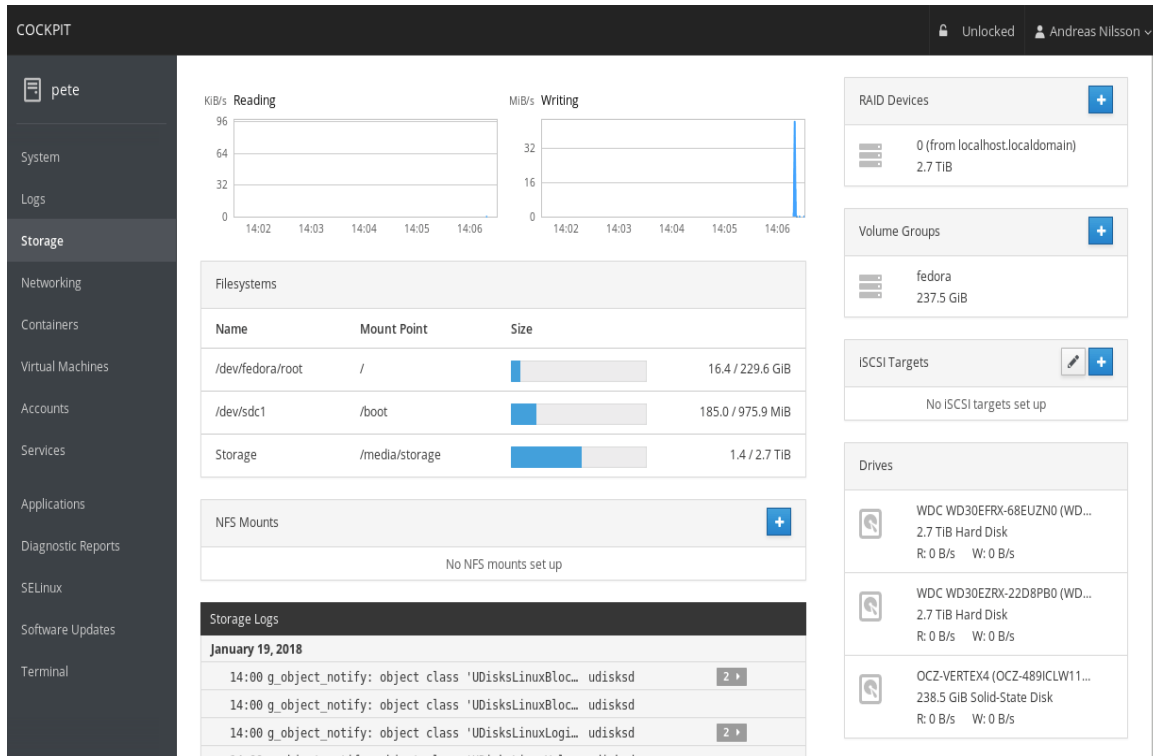


Figure 9: Poste de pilotage

La procédure d'installation varie pour chaque distribution. Dans certaines distributions, comme Fedora, CentOS et RHEL, Cockpit se trouve dans les référentiels officiels. Dans d'autres, comme Debian, Ubuntu et Linux Mint, vous pouvez installer Cockpit à partir de son PPA officiel.

### 3. Shorewall

Un outil de filtrage de paquets qui ajoute une couche d'abstraction pour obtenir une configuration de niveau supérieur de Netfilter.

Shorewall lit les fichiers de configuration et configure Netfilter dans le noyau Linux, avec le support des utilitaires **ip**, **tc**, **iptables** et **iptables-restore**.

L'avantage d'utiliser Shorewall est qu'il divise les interfaces en zones, en attribuant différents niveaux d'accès à chacune.

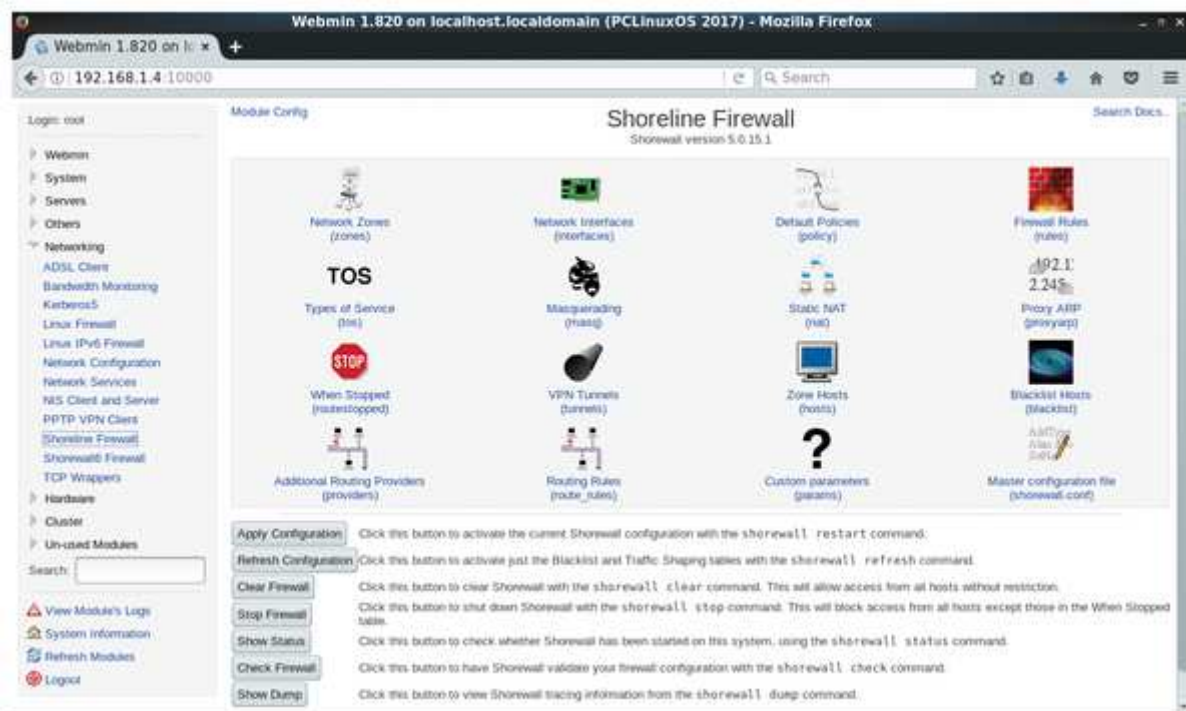


Figure 10 : Shorewall

Il permet à l'utilisateur d'opérer sur des groupes d'ordinateurs connectés à l'interface, au lieu de travailler sur des groupes d'adresses. Les utilisateurs peuvent facilement déployer différentes stratégies pour chaque zone.

## 4. Nagios

Un outil de surveillance de réseau open-source qui a été lancé en 2002 sous le nom de NetSaint.

Depuis, Nagios a parcouru un long chemin, se méritant une solide réputation pour son excellent travail de surveillance des serveurs et des périphériques réseau.

Cela fonctionne bien dès la sortie de la boîte lorsqu'il est chargé de surveiller un environnement avec de nombreux protocoles de base.

Nagios fournit également une base pour d'autres utilitaires de surveillance, tels que Naemon, Icinga et OP5.

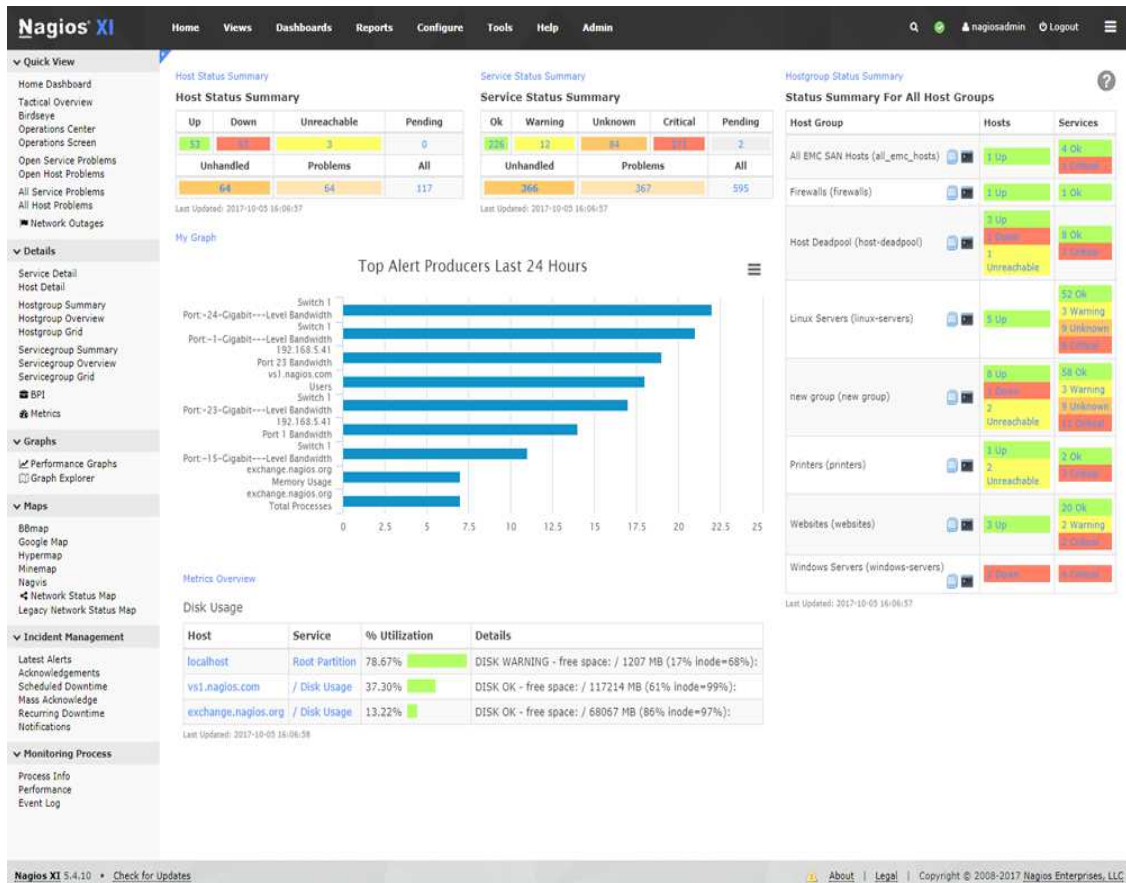


Figure 11: Nagios

## 5. phpMyAdmin

phpMyAdmin est un outil classique et très populaire qui gère la maintenance des bases de données.

phpMyAdmin s'exécute dans un navigateur Web, ce qui signifie qu'on peut l'utiliser à partir de n'importe quel appareil, même à partir d'un smartphone. Une autre raison est qu'il couvre toutes les fonctions nécessaires à la gestion de la base de données, et on n'a pratiquement pas besoin de savoir comment écrire des requêtes en SQL pour ce faire.

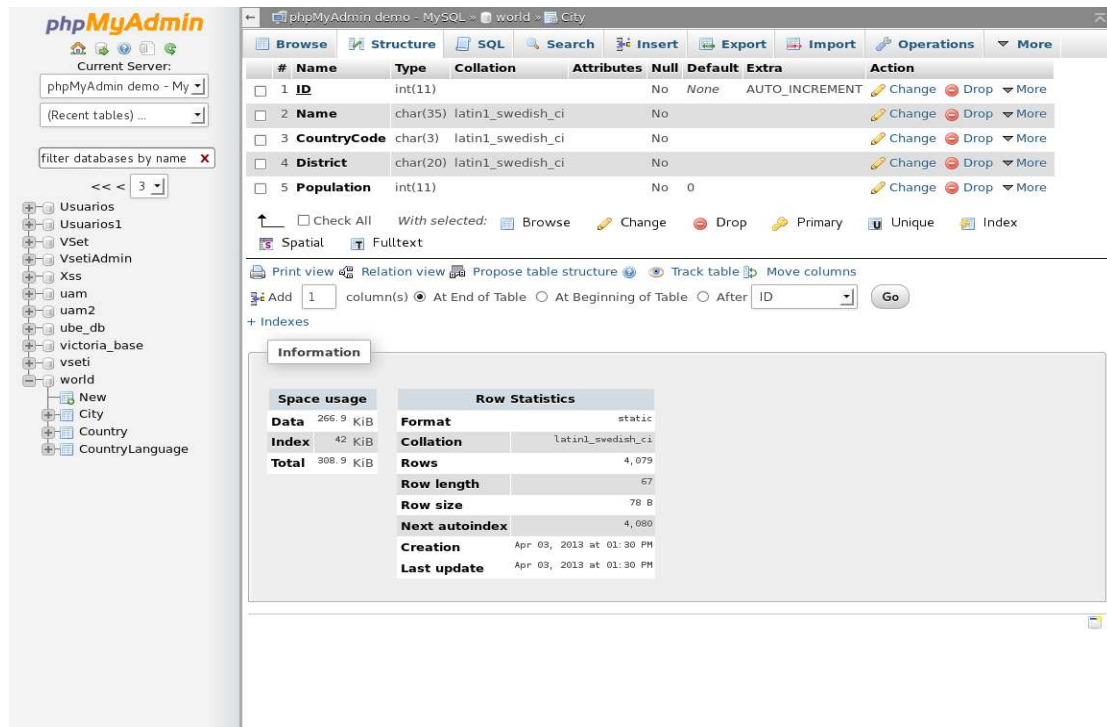


Figure 12: phpMyAdmin

## CONCLUSION GENERALE

Les libertés d'étudier et améliorer un logiciel supposent un accès au code source du logiciel.

L'accès au code source est important car les logiciels sont généralement distribués sous une forme compilée en langage machine, prêts à être exécutés par un ordinateur.

Les logiciels libres ont, dans leur grande majorité, tendance à respecter les formats standards ouverts, ce qui favorise l'interopérabilité. Par exemple, le développement du navigateur web Mozilla Firefox s'applique à respecter autant que possible les recommandations émises par le World Wide Web Consortium.

Les applications des logiciels populaires :

- ✓ **Firefox,**
- ✓ **VLC media player,**
- ✓ **Adobe Reader,**
- ✓ **WinRAR,**
- ✓ **CCleaner,**
- ✓ **Internet Download Manager,**
- ✓ **Bitdefender Total Security.**

Le contexte de logiciel libre est devenu un mouvement à part entière et une méthode de travail qui dépasse la simple création de logiciels.

Le mouvement s'appuie sur les valeurs et le modèle de production décentralisée pour trouver de nouvelles manières de résoudre les problèmes auxquels font face des communautés et des secteurs d'activité.

A jeter un œil sur mon blog :

**<https://afric-teach.blog4ever.com>**

Sans **www**