

# **FORMATUX - Support de cours GNU/Linux**

Version 1  
17-04-2017



---

# Table des matières

Préface .....	xv
1. Crédits .....	xvi
2. Licence .....	xvi
3. Gestion des versions .....	xvii
I. Administration .....	1
1. Introduction .....	11
1.1. Qu'est-ce qu'un système d'exploitation ? .....	11
1.2. Généralités UNIX - GNU/Linux .....	12
1.2.1. Historique .....	12
1.2.2. Parts de marché .....	14
1.2.3. Architecture .....	15
1.2.4. La philosophie Unix .....	16
1.3. Les distributions GNU/LINUX .....	17
1.3.1. Les environnements de bureaux .....	17
1.3.2. Libre / Open source .....	19
1.4. Les domaines d'emploi .....	20
1.5. Shell .....	20
1.5.1. Généralités .....	20
1.6. Fonctionnalités .....	21
1.6.1. Principe .....	22
2. Commandes pour utilisateurs Linux .....	23
2.1. Généralités .....	23
2.1.1. Les utilisateurs .....	24
2.1.2. Le Shell .....	25
2.2. Les commandes générales .....	26
2.2.1. Les commandes man et whatis .....	26
2.2.2. La commande shutdown .....	27
2.2.3. La commande history .....	28
2.3. Affichage et identification .....	29
2.3.1. La commande clear .....	29
2.3.2. La commande echo .....	29
2.3.3. La commande date .....	30
2.3.4. Les commandes id, who et whoami .....	32
2.4. Arborescence de fichiers .....	32

2.4.1. La commande pwd .....	34
2.4.2. La commande cd .....	34
2.4.3. La commande ls .....	35
2.4.4. La commande mkdir .....	38
2.4.5. La commande touch .....	39
2.4.6. La commande rmdir .....	39
2.4.7. La commande rm .....	40
2.4.8. La commande mv .....	41
2.4.9. La commande cp .....	42
2.5. Visualisation .....	43
2.5.1. La commande file .....	43
2.5.2. La commande more .....	44
2.5.3. La commande less .....	44
2.5.4. Les commandes cat et tac .....	45
2.5.5. La commande head .....	46
2.5.6. La commande tail .....	46
2.5.7. La commande sort .....	47
2.5.8. La commande wc .....	50
2.6. Recherche .....	50
2.6.1. La commande find .....	50
2.6.2. La commande whereis .....	51
2.6.3. La commande grep .....	52
2.6.4. Les métacaractères .....	53
2.7. Redirections et tubes .....	54
2.7.1. L'entrée et les sorties standards .....	54
2.7.2. Les tubes (pipe) .....	57
2.8. Points particuliers .....	58
2.8.1. La commande tee .....	58
2.8.2. Les commandes alias et unalias .....	59
2.8.3. Le caractère ; .....	62
2.9. TD commandes de bases .....	63
2.10. Correction du TD commandes de bases .....	68
3. La gestion des utilisateurs .....	79
3.1. Généralités .....	79
3.2. Gestion des groupes .....	79
3.2.1. Commande groupadd .....	80

3.2.2. Commande groupmod .....	80
3.2.3. Commande groupdel .....	81
3.2.4. Fichier /etc/group .....	82
3.3. Gestion des utilisateurs .....	83
3.3.1. Définition .....	83
3.3.2. Commande useradd .....	84
3.3.3. Commande usermod .....	85
3.3.4. Commande userdel .....	87
3.3.5. Fichier /etc/passwd .....	88
3.3.6. Fichier /etc/shadow .....	89
3.4. Propriétaires des fichiers .....	89
3.4.1. Commandes de modifications : .....	89
3.4.2. Commande chgrp .....	90
3.5. Gestion des invités .....	91
3.5.1. Commande gpasswd .....	91
3.5.2. Commande id .....	92
3.5.3. Commande newgrp .....	92
3.6. Sécurisation .....	93
3.6.1. Commande passwd .....	93
3.7. Commande chage .....	95
3.8. Gestion avancée .....	96
3.8.1. Fichier /etc/default/useradd .....	96
3.8.2. Fichier /etc/login.defs .....	97
3.8.3. Fichier /etc/skel .....	98
3.9. Changement d'identité .....	98
3.9.1. Commande su .....	98
4. Système de fichiers .....	101
4.1. Partitionnement .....	101
4.1.1. Commande cfdisk .....	104
4.2. LVM .....	105
4.2.1. Les groupes de volume .....	105
4.2.2. Les volumes logiques .....	106
4.2.3. Commandes LVM pour la gestion des volumes .....	108
4.2.4. Commandes LVM pour visualiser les informations concernant les volumes .....	110
4.2.5. Préparation du support physique .....	111

4.3. Structure d'un système de fichiers .....	111
4.3.1. Commande mkfs .....	111
4.3.2. Bloc de boot .....	112
4.3.3. Super bloc .....	112
4.3.4. Table des inodes .....	113
4.3.5. Zone de données .....	114
4.3.6. Réparation du système de fichiers .....	114
4.4. Organisation d'un système de fichiers .....	115
4.4.1. Le fichier /etc/fstab .....	117
4.4.2. Commandes de gestion des montages .....	119
4.5. Types de fichiers .....	120
4.5.1. Détails du nom d'un fichier .....	121
4.5.2. Différents types de fichiers .....	122
4.6. Attributs des fichiers .....	126
4.6.1. Droits associés aux fichiers ordinaires .....	127
4.6.2. Droits associés aux répertoires .....	127
4.6.3. Gestion des attributs .....	127
4.6.4. Les droits particuliers .....	132
4.7. Droits par défaut et masque .....	134
4.7.1. Commande umask .....	135
5. Gestion des processus .....	137
5.1. Généralités .....	137
5.2. Visualisation des processus .....	138
5.3. Types de processus .....	140
5.4. Permissions et droits .....	140
5.5. Gestion des processus .....	141
5.5.1. La priorité d'un processus .....	141
5.5.2. Modes de fonctionnements .....	141
5.6. Les commandes de gestion des processus .....	142
5.6.1. La commande kill .....	142
5.6.2. La commande nohup .....	143
5.6.3. Instruction & .....	143
5.6.4. Les commandes fg et bg .....	144
5.6.5. La commande jobs .....	144
5.6.6. Les commandes nice/renice .....	145
5.6.7. La commande top .....	146

6. Sauvegardes et restaurations .....	147
6.1. Généralités .....	147
6.1.1. La démarche .....	148
6.1.2. Méthodes de sauvegardes .....	148
6.1.3. Périodicité .....	148
6.1.4. Méthodes de restauration .....	149
6.1.5. Les outils .....	149
6.1.6. Convention de nommage .....	150
6.1.7. Contenu d'une sauvegarde .....	150
6.1.8. Modes de stockage .....	151
6.2. Tape ArchiveR - tar .....	152
6.2.1. Consignes de restauration .....	152
6.2.2. La sauvegarde avec tar .....	152
6.3. CoPy Input Output - cpio .....	160
6.3.1. Créer une sauvegarde .....	160
6.3.2. Type de sauvegarde .....	161
6.3.3. Ajouter à une sauvegarde .....	162
6.3.4. Compresser une sauvegarde .....	162
6.3.5. Lire le contenu d'une sauvegarde .....	163
6.3.6. Restaurer une sauvegarde .....	163
6.4. Utilitaires de compression - décompression .....	166
6.4.1. Compresser avec gzip .....	166
6.4.2. Compresser avec bunzip2 .....	166
6.4.3. Décompresser avec gunzip .....	167
6.4.4. Décompresser avec bunzip2 .....	167
7. Démarrage du système .....	169
7.1. Démarrage de l'ordinateur .....	169
7.1.1. Séquence de démarrage .....	169
7.1.2. Amorçage matériel .....	169
7.1.3. Le chargeur de démarrage .....	170
7.1.4. Types de chargeurs .....	170
7.2. Le chargeur GRUB .....	170
7.2.1. Choix du système .....	171
7.2.2. Fichier /boot/grub/grub.conf .....	172
7.3. Sécuriser GRUB .....	174
7.3.1. Définir un mot de passe .....	174

7.3.2. Chiffrer le mot de passe .....	174
7.3.3. Menu interactif verrouillé .....	176
7.3.4. Lancement verrouillé .....	176
7.4. Démarrage du noyau .....	176
7.4.1. Niveaux de démarrage .....	177
7.4.2. Étapes du démarrage .....	177
7.5. Le processus init (généralités) .....	177
7.5.1. Les différents niveaux d'exécution .....	177
7.5.2. La commande init .....	178
7.5.3. La commande runlevel .....	178
7.5.4. Le fichier /etc/inittab .....	179
7.5.5. Changement de niveau .....	179
7.5.6. Activer ou désactiver les terminaux .....	179
7.5.7. Autoriser à root l'accès aux terminaux .....	180
7.6. Le processus init (démon) .....	181
7.6.1. Démarrage des démons .....	181
7.6.2. Script de démarrage des services .....	181
7.6.3. Répertoires d'ordonnancement .....	181
7.6.4. Nom des liens .....	181
7.6.5. Le programme /etc/rc.d/rc .....	182
7.6.6. Architecture de démarrage .....	183
7.7. La gestion des services .....	183
7.7.1. La commande ln .....	183
7.7.2. La commande chkconfig .....	184
7.8. Arrêt du système .....	187
7.8.1. Commande shutdown .....	188
7.8.2. Commande halt .....	189
7.8.3. Commande reboot .....	189
8. Démarrage du système sous CentOS 7 .....	191
8.1. Le processus de démarrage .....	191
8.1.1. Le démarrage du BIOS .....	191
8.1.2. Le Master boot record (MBR) .....	191
8.1.3. Le chargeur de démarrage GRUB2 (Bootloader) .....	191
8.1.4. Le noyau .....	192
8.1.5. systemd .....	192
8.2. Protéger le chargeur de démarrage GRUB2 .....	193

8.3. Systemd .....	195
8.3.1. Gérer les services systèmes .....	196
8.3.2. Exemple de fichier .service pour le service postfix ....	198
8.3.3. Utiliser les targets systèmes .....	198
8.3.4. Le processus journald .....	202
8.3.5. La commande journalctl .....	202
9. Gestion des tâches .....	205
9.1. Généralités .....	205
9.2. Fonctionnement du service .....	205
9.3. La sécurité .....	206
9.3.1. Autorisations .....	207
9.3.2. Autoriser un utilisateur .....	207
9.3.3. Interdire un utilisateur .....	207
9.4. La planification des tâches .....	207
9.4.1. Commande crontab .....	208
9.4.2. Intérêts de la planification .....	209
9.5. Le fichier de planification .....	209
9.5.1. Processus d'exécution d'une tâche .....	211
10. Mise en oeuvre du réseau .....	213
10.1. Généralités .....	213
10.1.1. Adresse MAC / Adresse IP .....	215
10.1.2. Domaine DNS .....	216
10.1.3. Rappel du modèle OSI .....	216
10.2. Le nommage des interfaces .....	217
10.3. Utiliser la commande IP .....	217
10.4. Le nom de machine .....	218
10.5. Le fichier /etc/hosts .....	219
10.6. Le fichier /etc/nsswitch.conf .....	220
10.7. Le fichier /etc/resolv.conf .....	220
10.8. La commande IP .....	221
10.9. Configuration DHCP .....	222
10.10. Configuration statique .....	223
10.11. Routage .....	224
10.12. Résolution de noms .....	225
10.13. Dépannage .....	226
10.13.1. La commande dig .....	227

10.13.2. La commande getent .....	228
10.13.3. La commande ipcalc .....	229
10.13.4. La commande ss .....	230
10.13.5. La commande netstat .....	230
10.13.6. Les conflits d'adresses IP ou d'adresses MAC .....	231
10.14. Configuration à chaud .....	232
10.15. En résumé .....	232
11. Gestion des logiciels .....	235
11.1. Généralités .....	235
11.2. RPM : RedHat Package Manager .....	235
11.2.1. Commande rpm .....	236
11.3. YUM : Yellow dog Updater Modified .....	237
11.3.1. Commande yum .....	238
11.3.2. Fonctionnement de YUM .....	239
11.4. Gérer son dépôt .....	239
11.5. Le dépôt EPEL .....	240
11.5.1. Installation .....	240
II. Sécurité .....	241
12. Elévation des priviléges (su et sudo) .....	247
12.1. Limiter le compte root .....	248
12.2. La commande su .....	248
12.3. La commande sudo .....	249
12.3.1. Avantages .....	250
12.4. Commande visudo .....	250
12.4.1. Le fichier /etc/sudoers .....	251
12.4.2. Le groupe wheel .....	251
12.4.3. Ne plus utiliser root .....	251
12.4.4. Restreindre le sudo .....	252
13. Les modules d'authentification PAM .....	257
13.1. Généralités .....	257
13.1.1. Syntaxe d'une directive .....	258
13.2. Les mécanismes .....	259
13.2.1. Le mécanisme auth - Authentification .....	259
13.2.2. Le mécanisme account - Gestion de compte .....	259
13.2.3. Le mécanisme session - Gestion de session .....	259

13.2.4. Le mécanisme password - Gestion des mots de passe .....	260
13.3. Les indicateurs de contrôle .....	260
13.3.1. L'indicateur de contrôle required .....	260
13.3.2. L'indicateur de contrôle requisite .....	260
13.3.3. L'indicateur de contrôle sufficient .....	261
13.3.4. L'indicateur de contrôle optional .....	261
13.3.5. En conclusion .....	262
13.4. Les modules de PAM .....	262
13.4.1. Le module pam_unix .....	262
13.4.2. Le module pam_cracklib .....	263
13.4.3. Le module pam_tally .....	264
13.4.4. Le module pam_time .....	265
13.4.5. Le module pam_nologin .....	266
13.4.6. Le module pam_wheel .....	266
13.4.7. Le module pam_mount .....	267
14. Securisation SELinux .....	269
14.1. Généralités .....	269
14.1.1. Le contexte SELinux .....	270
14.2. Gestion .....	273
14.2.1. Administrer les objets de type booléens .....	274
14.3. Mode de fonctionnement .....	275
14.3.1. Le fichier /etc/sysconfig/selinux .....	276
14.4. Les jeux de règles (Policy Type) : .....	277
14.5. Contexte .....	277
14.5.1. Aller plus loin avec SELinux .....	279
15. IPtables, le parefeu Linux .....	281
15.1. Gestion du pare-feu .....	282
15.1.1. Démarrer/arrêter ou redémarrer le parefeu .....	282
15.1.2. Afficher les règles .....	282
15.2. Quelques exemples .....	283
15.2.1. Bloquer des adresses IP spécifiques .....	283
15.2.2. Bloquer/accepter des ports spécifiques .....	283
15.2.3. Autoriser un réseau .....	284
15.2.4. Limiter le nombre de connexion d'une adresse .....	284
15.2.5. Bloquer le protocole ICMP (ping) .....	284

15.2.6. Accès à la loopback .....	285
15.2.7. Logger les paquets refusés .....	285
15.2.8. Gérer les connexions établies .....	285
15.2.9. Supprimer les paquets invalides .....	285
15.2.10. Bloquer le trafic SMTP sortant .....	285
15.3. Conclusion .....	286
16. Fail2ban .....	287
16.1. Installation .....	287
16.2. Configuration .....	287
16.3. Lancement du service .....	289
16.4. Vérification du service .....	289
16.5. Interface graphique .....	290
17. Sécuriser le serveur SSH .....	291
17.1. Configuration .....	291
17.2. Changer le port d'écoute et la version du protocole .....	291
17.3. Utilisation de clefs privées/publiques .....	291
17.4. Limiter les accès .....	292
17.5. Interdire l'accès à root !!! .....	292
17.6. Sécurité par le parefeu .....	292
18. Autorité de certification TLS avec easy-rsa .....	293
18.1. Installer easy-rsa : .....	293
18.2. Configuration .....	293
18.3. Créer une autorité de certification .....	294
18.4. Créer une biclé serveur .....	295
18.5. Installer le certificat de l'autorité de certification .....	295
Glossaire .....	297
III. Shell .....	299
19. Les scripts SHELL - Niveau 1 .....	303
19.1. Premier script .....	304
19.2. Variables .....	306
19.2.1. Supprimer et verrouiller les variables .....	308
19.2.2. Variables d'environnements .....	308
19.2.3. Exporter une variable .....	309
19.2.4. La substitution de commande .....	309
19.2.5. Améliorations du script de sauvegarde .....	310
19.3. Saisie et manipulations .....	312

19.3.1. La commande read .....	312
19.3.2. La commande cut .....	313
19.3.3. La commande tr .....	314
19.3.4. Extraire le nom et le chemin d'un fichier .....	315
19.3.5. Arguments d'un script .....	315
20. Scripts shell - Instructions de contrôle .....	321
20.1. Tests .....	321
20.1.1. Tester le type d'un fichier .....	322
20.1.2. Comparer deux fichiers .....	323
20.1.3. Tester une variable .....	323
20.1.4. Tester une chaîne de caractères .....	324
20.1.5. Comparaison de numériques entiers .....	324
20.1.6. Combinaison de tests .....	325
20.1.7. Les opérations numériques .....	326
20.1.8. La commande typeset .....	327
20.1.9. La commande let .....	327
20.2. Structures conditionnelles .....	328
20.2.1. Structure alternative conditionnelle case .....	330
20.3. Boucles .....	332
20.3.1. La structure boucle conditionnelle while .....	332
20.3.2. La commande exit .....	333
20.3.3. La commande break / continue .....	334
20.3.4. Les commandes true / false .....	334
20.3.5. La structure boucle conditionnelle until .....	335
20.3.6. La structure choix alternatif select .....	335
20.3.7. La structure boucle sur liste de valeurs for .....	336
21. TP Scripting SHELL .....	339
21.1. Etude du besoin .....	339
21.2. Consignes .....	340
21.3. Pistes de travail .....	340
21.4. Proposition de correction .....	341
21.4.1. Création de l'utilisateur .....	341
21.4.2. Menu .....	341
21.4.3. Quelques fonctions utilitaires .....	343
21.4.4. La gestion du réseau .....	345
21.4.5. La gestion des services .....	346

21.4.6. L'affichage des utilisateurs .....	347
Index .....	351

---

# Préface

## Table des matières

1. Crédits .....	xvi
2. Licence .....	xvi
3. Gestion des versions .....	xvii

GNU/Linux est un **système d'exploitation** libre fonctionnant sur la base d'un **noyau Linux**, également appelé **kernel Linux**.

Linux est une implémentation libre du système **UNIX** et respecte les spécifications **POSIX**.

GNU/Linux est généralement distribué dans un ensemble cohérent de logiciels, assemblés autour du noyau Linux et prêt à être installé. Cet ensemble porte le nom de "**Distribution**".

- La plus ancienne des distributions est la distribution **Slackware**.
- Les plus connues et utilisées sont les distributions **Debian**, **Redhat** et **Arch**, et servent de base pour d'autres distributions comme **Ubuntu**, **CentOS**, **Fedora**, **Mageia** ou **Manjaro**.

Chaque distribution présente des particularités et peut être développée pour répondre à des besoins très précis :

- services d'infrastructure ;
- pare-feu ;
- serveur multimédia ;
- serveur de stockage ;
- etc.

La distribution présentée dans ces pages est la **CentOS**, qui est le pendant gratuit de la distribution **RedHat**. La distribution **CentOS** est particulièrement adaptée pour un usage sur des serveurs d'entreprises.

## 1. Crédits

Ce support de cours a été rédigé par les formateurs :

- Patrick Finet ;
- Antoine Le Morvan ;
- Xavier Sauvignon ;

## 2. Licence

Formatux propose des supports de cours Linux libres de droits à destination des formateurs ou des personnes désireuses d'apprendre à administrer un système Linux en autodidacte.

Les supports de Formatux sont publiés sous licence Creative Commons-BY-SA et sous licence Art Libre. Vous êtes ainsi libre de copier, de diffuser et de transformer librement les œuvres dans le respect des droits de l'auteur.

**BY** : Paternité. Vous devez citer le nom de l'auteur original.

**SA** : Partage des Conditions Initiales à l'Identique.

- Licence Creative Commons-BY-SA : <https://creativecommons.org/licenses/by-sa/3.0/fr/>
- Licence Art Libre : <http://artlibre.org/>

Les documents de formatux et leurs sources sont librements téléchargeables sur framagit :

- <https://framagit.org/alemorvan/formatux.fr-support/>

Vous y trouverez la dernière version de ce document.

A partir des sources, vous pouvez générer votre support de formation personnalisé. Nous vous recommandons le logiciel AsciidocFX téléchargeable ici : <http://asciidocfx.com/>

### 3. Gestion des versions

Tableau 1. Historique des versions du document

Version	Date	Observations
1.0	Avril 2017	Version initiale.



---

# **Partie I. Administration**



---

# Table des matières

1. Introduction .....	11
1.1. Qu'est-ce qu'un système d'exploitation ? .....	11
1.2. Généralités UNIX - GNU/Linux .....	12
1.2.1. Historique .....	12
1.2.2. Parts de marché .....	14
1.2.3. Architecture .....	15
1.2.4. La philosophie Unix .....	16
1.3. Les distributions GNU/LINUX .....	17
1.3.1. Les environnements de bureaux .....	17
1.3.2. Libre / Open source .....	19
1.4. Les domaines d'emploi .....	20
1.5. Shell .....	20
1.5.1. Généralités .....	20
1.6. Fonctionnalités .....	21
1.6.1. Principe .....	22
2. Commandes pour utilisateurs Linux .....	23
2.1. Généralités .....	23
2.1.1. Les utilisateurs .....	24
2.1.2. Le Shell .....	25
2.2. Les commandes générales .....	26
2.2.1. Les commandes man et whatis .....	26
2.2.2. La commande shutdown .....	27
2.2.3. La commande history .....	28
2.3. Affichage et identification .....	29
2.3.1. La commande clear .....	29
2.3.2. La commande echo .....	29
2.3.3. La commande date .....	30
2.3.4. Les commandes id, who et whoami .....	32
2.4. Arborescence de fichiers .....	32
2.4.1. La commande pwd .....	34
2.4.2. La commande cd .....	34
2.4.3. La commande ls .....	35
2.4.4. La commande mkdir .....	38
2.4.5. La commande touch .....	39

---

2.4.6. La commande rmdir .....	39
2.4.7. La commande rm .....	40
2.4.8. La commande mv .....	41
2.4.9. La commande cp .....	42
2.5. Visualisation .....	43
2.5.1. La commande file .....	43
2.5.2. La commande more .....	44
2.5.3. La commande less .....	44
2.5.4. Les commandes cat et tac .....	45
2.5.5. La commande head .....	46
2.5.6. La commande tail .....	46
2.5.7. La commande sort .....	47
2.5.8. La commande wc .....	50
2.6. Recherche .....	50
2.6.1. La commande find .....	50
2.6.2. La commande whereis .....	51
2.6.3. La commande grep .....	52
2.6.4. Les méta-caractères .....	53
2.7. Redirections et tubes .....	54
2.7.1. L'entrée et les sorties standards .....	54
2.7.2. Les tubes (pipe) .....	57
2.8. Points particuliers .....	58
2.8.1. La commande tee .....	58
2.8.2. Les commandes alias et unalias .....	59
2.8.3. Le caractère ; .....	62
2.9. TD commandes de bases .....	63
2.10. Correction du TD commandes de bases .....	68
3. La gestion des utilisateurs .....	79
3.1. Généralités .....	79
3.2. Gestion des groupes .....	79
3.2.1. Commande groupadd .....	80
3.2.2. Commande groupmod .....	80
3.2.3. Commande groupdel .....	81
3.2.4. Fichier /etc/group .....	82
3.3. Gestion des utilisateurs .....	83
3.3.1. Définition .....	83

---

3.3.2. Commande useradd .....	84
3.3.3. Commande usermod .....	85
3.3.4. Commande userdel .....	87
3.3.5. Fichier /etc/passwd .....	88
3.3.6. Fichier /etc/shadow .....	89
3.4. Propriétaires des fichiers .....	89
3.4.1. Commandes de modifications : .....	89
3.4.2. Commande chgrp .....	90
3.5. Gestion des invités .....	91
3.5.1. Commande gpasswd .....	91
3.5.2. Commande id .....	92
3.5.3. Commande newgrp .....	92
3.6. Sécurisation .....	93
3.6.1. Commande passwd .....	93
3.7. Commande chage .....	95
3.8. Gestion avancée .....	96
3.8.1. Fichier /etc/default/useradd .....	96
3.8.2. Fichier /etc/login.defs .....	97
3.8.3. Fichier /etc/skel .....	98
3.9. Changement d'identité .....	98
3.9.1. Commande su .....	98
4. Système de fichiers .....	101
4.1. Partitionnement .....	101
4.1.1. Commande cfdisk .....	104
4.2. LVM .....	105
4.2.1. Les groupes de volume .....	105
4.2.2. Les volumes logiques .....	106
4.2.3. Commandes LVM pour la gestion des volumes .....	108
4.2.4. Commandes LVM pour visualiser les informations concernant les volumes .....	110
4.2.5. Préparation du support physique .....	111
4.3. Structure d'un système de fichiers .....	111
4.3.1. Commande mkfs .....	111
4.3.2. Bloc de boot .....	112
4.3.3. Super bloc .....	112
4.3.4. Table des inodes .....	113

---

---

4.3.5. Zone de données .....	114
4.3.6. Réparation du système de fichiers .....	114
4.4. Organisation d'un système de fichiers .....	115
4.4.1. Le fichier /etc/fstab .....	117
4.4.2. Commandes de gestion des montages .....	119
4.5. Types de fichiers .....	120
4.5.1. Détails du nom d'un fichier .....	121
4.5.2. Différents types de fichiers .....	122
4.6. Attributs des fichiers .....	126
4.6.1. Droits associés aux fichiers ordinaires .....	127
4.6.2. Droits associés aux répertoires .....	127
4.6.3. Gestion des attributs .....	127
4.6.4. Les droits particuliers .....	132
4.7. Droits par défaut et masque .....	134
4.7.1. Commande umask .....	135
5. Gestion des processus .....	137
5.1. Généralités .....	137
5.2. Visualisation des processus .....	138
5.3. Types de processus .....	140
5.4. Permissions et droits .....	140
5.5. Gestion des processus .....	141
5.5.1. La priorité d'un processus .....	141
5.5.2. Modes de fonctionnements .....	141
5.6. Les commandes de gestion des processus .....	142
5.6.1. La commande kill .....	142
5.6.2. La commande nohup .....	143
5.6.3. Instruction & .....	143
5.6.4. Les commandes fg et bg .....	144
5.6.5. La commande jobs .....	144
5.6.6. Les commandes nice/renice .....	145
5.6.7. La commande top .....	146
6. Sauvegardes et restaurations .....	147
6.1. Généralités .....	147
6.1.1. La démarche .....	148
6.1.2. Méthodes de sauvegardes .....	148
6.1.3. Périodicité .....	148

---

---

6.1.4. Méthodes de restauration .....	149
6.1.5. Les outils .....	149
6.1.6. Convention de nommage .....	150
6.1.7. Contenu d'une sauvegarde .....	150
6.1.8. Modes de stockage .....	151
6.2. Tape ArchiveR - tar .....	152
6.2.1. Consignes de restauration .....	152
6.2.2. La sauvegarde avec tar .....	152
6.3. CoPy Input Output - cpio .....	160
6.3.1. Créer une sauvegarde .....	160
6.3.2. Type de sauvegarde .....	161
6.3.3. Ajouter à une sauvegarde .....	162
6.3.4. Compresser une sauvegarde .....	162
6.3.5. Lire le contenu d'une sauvegarde .....	163
6.3.6. Restaurer une sauvegarde .....	163
6.4. Utilitaires de compression - décompression .....	166
6.4.1. Compresser avec gzip .....	166
6.4.2. Compresser avec bunzip2 .....	166
6.4.3. Décompresser avec gunzip .....	167
6.4.4. Décompresser avec bunzip2 .....	167
7. Démarrage du système .....	169
7.1. Démarrage de l'ordinateur .....	169
7.1.1. Séquence de démarrage .....	169
7.1.2. Amorçage matériel .....	169
7.1.3. Le chargeur de démarrage .....	170
7.1.4. Types de chargeurs .....	170
7.2. Le chargeur GRUB .....	170
7.2.1. Choix du système .....	171
7.2.2. Fichier /boot/grub/grub.conf .....	172
7.3. Sécuriser GRUB .....	174
7.3.1. Définir un mot de passe .....	174
7.3.2. Chiffrer le mot de passe .....	174
7.3.3. Menu interactif verrouillé .....	176
7.3.4. Lancement verrouillé .....	176
7.4. Démarrage du noyau .....	176
7.4.1. Niveaux de démarrage .....	177

---

---

7.4.2. Étapes du démarrage .....	177
7.5. Le processus init (généralités) .....	177
7.5.1. Les différents niveaux d'exécution .....	177
7.5.2. La commande init .....	178
7.5.3. La commande runlevel .....	178
7.5.4. Le fichier /etc/inittab .....	179
7.5.5. Changement de niveau .....	179
7.5.6. Activer ou désactiver les terminaux .....	179
7.5.7. Autoriser à root l'accès aux terminaux .....	180
7.6. Le processus init (démon) .....	181
7.6.1. Démarrage des démons .....	181
7.6.2. Script de démarrage des services .....	181
7.6.3. Répertoires d'ordonnancement .....	181
7.6.4. Nom des liens .....	181
7.6.5. Le programme /etc/rc.d/rc .....	182
7.6.6. Architecture de démarrage .....	183
7.7. La gestion des services .....	183
7.7.1. La commande ln .....	183
7.7.2. La commande chkconfig .....	184
7.8. Arrêt du système .....	187
7.8.1. Commande shutdown .....	188
7.8.2. Commande halt .....	189
7.8.3. Commande reboot .....	189
8. Démarrage du système sous CentOS 7 .....	191
8.1. Le processus de démarrage .....	191
8.1.1. Le démarrage du BIOS .....	191
8.1.2. Le Master boot record (MBR) .....	191
8.1.3. Le chargeur de démarrage GRUB2 (Bootloader) .....	191
8.1.4. Le noyau .....	192
8.1.5. systemd .....	192
8.2. Protéger le chargeur de démarrage GRUB2 .....	193
8.3. Systemd .....	195
8.3.1. Gérer les services systèmes .....	196
8.3.2. Exemple de fichier .service pour le service postfix .....	198
8.3.3. Utiliser les targets systèmes .....	198
8.3.4. Le processus journald .....	202

---

8.3.5. La commande journalctl .....	202
9. Gestion des tâches .....	205
9.1. Généralités .....	205
9.2. Fonctionnement du service .....	205
9.3. La sécurité .....	206
9.3.1. Autorisations .....	207
9.3.2. Autoriser un utilisateur .....	207
9.3.3. Interdire un utilisateur .....	207
9.4. La planification des tâches .....	207
9.4.1. Commande crontab .....	208
9.4.2. Intérêts de la planification .....	209
9.5. Le fichier de planification .....	209
9.5.1. Processus d'exécution d'une tâche .....	211
10. Mise en oeuvre du réseau .....	213
10.1. Généralités .....	213
10.1.1. Adresse MAC / Adresse IP .....	215
10.1.2. Domaine DNS .....	216
10.1.3. Rappel du modèle OSI .....	216
10.2. Le nommage des interfaces .....	217
10.3. Utiliser la commande IP .....	217
10.4. Le nom de machine .....	218
10.5. Le fichier /etc/hosts .....	219
10.6. Le fichier /etc/nsswitch.conf .....	220
10.7. Le fichier /etc/resolv.conf .....	220
10.8. La commande IP .....	221
10.9. Configuration DHCP .....	222
10.10. Configuration statique .....	223
10.11. Routage .....	224
10.12. Résolution de noms .....	225
10.13. Dépannage .....	226
10.13.1. La commande dig .....	227
10.13.2. La commande getent .....	228
10.13.3. La commande ipcalc .....	229
10.13.4. La commande ss .....	230
10.13.5. La commande netstat .....	230
10.13.6. Les conflits d'adresses IP ou d'adresses MAC .....	231

---

---

10.14. Configuration à chaud .....	232
10.15. En résumé .....	232
11. Gestion des logiciels .....	235
11.1. Généralités .....	235
11.2. RPM : RedHat Package Manager .....	235
11.2.1. Commande rpm .....	236
11.3. YUM : Yellow dog Updater Modified .....	237
11.3.1. Commande yum .....	238
11.3.2. Fonctionnement de YUM .....	239
11.4. Gérer son dépôt .....	239
11.5. Le dépôt EPEL .....	240
11.5.1. Installation .....	240

---

# 1

## Introduction

---

### 1.1. Qu'est-ce qu'un système d'exploitation ?

Linux est un **système d'exploitation**.

Un système d'exploitation est un **ensemble de programmes permettant la gestion des ressources disponibles d'un ordinateur**.

Parmi cette gestion des ressources, le système d'exploitation est amené à :

- Gérer la mémoire physique ou virtuelle.
  - La **mémoire physique** est composée des barrettes de mémoires vives et de la mémoire cache du processeur, qui sert pour l'exécution des programmes.
  - La **mémoire virtuelle** est un emplacement sur le disque dur (la partition **swap**) qui permet de décharger la mémoire physique et de sauvegarder l'état en cours du système durant l'arrêt électrique de l'ordinateur (hibernation du système).
- Intercepter les **accès aux périphériques**. Les logiciels ne sont que très rarement autorisés à accéder directement au matériel (à l'exception des cartes graphiques pour des besoins très spécifiques).
- Offrir aux applications une **gestion correcte des tâches**. Le système d'exploitation est responsable de l'ordonnancement des processus pour l'occupation du processeur.
- **Protéger les fichiers** contre tout accès non autorisé.

- **Collecter les informations** sur les programmes utilisés ou en cours d'utilisation.

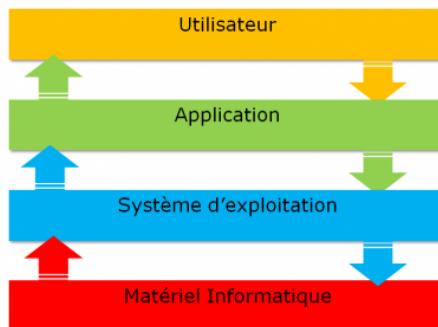


Figure 1.1. Fonctionnement d'un système d'exploitation

## 1.2. Généralités UNIX - GNU/Linux

### 1.2.1. Historique

#### UNIX

- De **1964 à 1968** : MULTICS (MULTIplexed Information and Computing Service) est développé pour le compte du MIT, des laboratoires Bell Labs (AT&T) et de General Electric.
- **1969** : Après le retrait de Bell (1969) puis de General Electric du projet, deux développeurs (Ken Thompson et Dennis Ritchie), rejoints plus tard par Brian Kernighan, jugeant MULTICS trop complexe, lancent le développement d'UNIX (UNiplexed Information and Computing Service). À l'origine développé en assembleur, les concepteurs d'UNIX ont développé le langage B puis le langage C (1971) et totalement réécrit UNIX. Ayant été développé en 1970, la date de référence des systèmes Unix/Linux est toujours fixée au 01 janvier 1970.

Le langage C fait toujours partie des langages de programmation les plus populaires aujourd'hui ! Langage de bas niveau, proche du matériel, il permet l'adaptation du système d'exploitation à toute architecture machine disposant d'un compilateur C.

Unix est un système d'exploitation ouvert et évolutif ayant joué un rôle primordial dans l'histoire de l'informatique. Il a servi de base pour de nombreux autres systèmes : Linux, BSD, Mac OSX, etc.

Unix est toujours d'actualité (HP-UX, AIX, Solaris, etc.)

## ***Minix***

- **1987** : Minix. A.S. Tanenbaum développe MINIX, un UNIX simplifié, pour enseigner les systèmes d'exploitation de façon simple. M. Tanenbaum rend disponible les sources de son système d'exploitation.

## ***Linux***



**Figure 1.2. Linus Torvalds, créateur du noyau Linux**

- **1991** : Linux. Un étudiant finlandais, Linus Torvalds, crée un système d'exploitation dédié à son ordinateur personnel et le nomme Linux. Il publie sa première version 0.02, sur le forum de discussion Usenet et d'autres développeurs viennent ainsi l'aider à améliorer son système. Le terme Linux est un jeu de mot entre le prénom du fondateur, Linus, et Unix.
- **1993** : La distribution Debian est créée. Debian est une distribution non commerciale à gestion associative. À l'origine développée pour une utilisation sur des serveurs, elle est particulièrement bien adaptée à ce rôle, mais elle se veut être un système universel et donc utilisable également sur un ordinateur personnel. Debian est utilisée comme base pour de nombreuses autres distributions, comme Mint ou Ubuntu.
- **1994** : La distribution commerciale RedHat est créée par la société RedHat, qui est aujourd'hui le premier distributeur du système d'exploitation GNU/Linux.

RedHat soutient la version communautaire Fedora et depuis peu la distribution libre CentOS.

- **1997** : L'environnement de bureau KDE est créé. Il est basé sur la bibliothèque de composants Qt et sur le langage de développement C++.
- **1999** : L'environnement de bureau Gnome est créé. Il est quant à lui basé sur la bibliothèque de composants GTK+.
- **2002** : La distribution Arch est créée. Sa particularité est d'être diffusée en Rolling Release (mise à jour en continue).
- **2004** : Ubuntu est créée par la société Canonical (Mark Shuttleworth). Elle se base sur Debian, mais regroupe des logiciels libres et privés.

### 1.2.2. Parts de marché

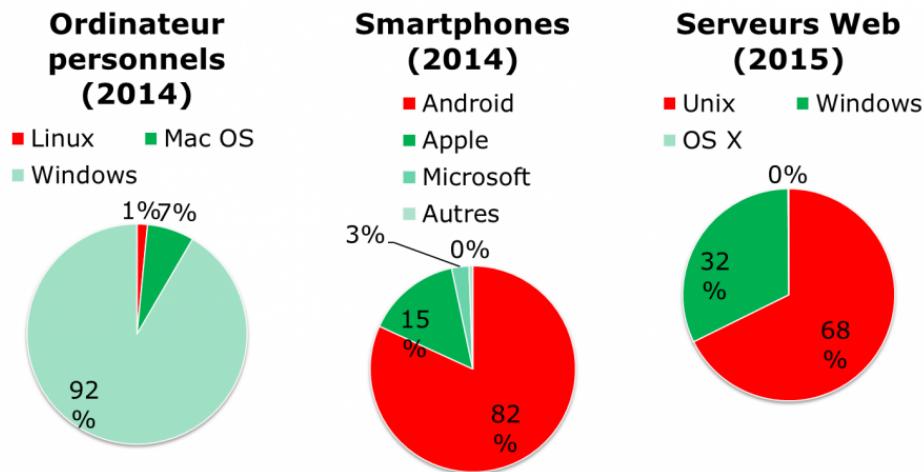


Figure 1.3. Les parts de marché de Linux

Linux est finalement encore peu connu du grand public, alors que ce dernier l'utilise régulièrement. En effet, Linux se cache dans les **smartphones**, les **téléviseurs**, les **box internet**, etc. Presque **70% des pages web** servies dans le monde le sont par un serveur Linux ou Unix !

Linux équipe un peu plus d'**1,5% des ordinateurs personnels** mais plus de **82% des smartphones**. **Android** étant un système d'exploitation dont le kernel est un Linux.

### **1.2.3. Architecture**

- Le noyau (ou kernel) est le premier composant logiciel.
  - Il est le cœur du système UNIX.
  - C'est lui qui gère les ressources matérielles du système.
  - Les autres composants logiciels passent obligatoirement par lui pour accéder au matériel.
- Le Shell est un utilitaire qui interprète les commandes de l'utilisateur et assure leur exécution.
  - Principaux shell : Bourne shell, C shell, Korn shell et Bourne Again shell (bash).
- Les applications regroupent les programmes utilisateurs comme :
  - le navigateur internet ;
  - le traitement de texte ;
  - ...

### **Multitâche**

Linux fait partie de la famille des systèmes d'exploitation à temps partagé. Il partage le temps d'utilisation processus entre plusieurs programmes, passant de l'un à l'autre de façon transparente pour l'utilisateur. Cela implique :

- exécution simultanée de plusieurs programmes ;
- distribution du temps CPU par l'ordonnanceur ;
- réduction des problèmes dus à une application défaillante ;
- diminution des performances lorsqu'il y a trop de programmes lancés.

### **Multiutilisateur**

La finalité de Multics était de permettre à plusieurs utilisateurs de travailler à partir de plusieurs terminaux (écran et clavier) sur un seul ordinateur (très coûteux à l'époque). Linux étant un descendant de ce système d'exploitation, il a gardé cette capacité à pouvoir fonctionner avec plusieurs utilisateurs simultanément

et en toute indépendance, chacun ayant son compte utilisateur, son espace de mémoire et ses droits d'accès aux fichiers et aux logiciels.

## **Multiprocesseur**

Linux est capable de travailler avec des ordinateurs multiprocesseurs ou avec des processeurs multicœurs.

## **Multiplateforme**

Linux est écrit en langage de haut niveau pouvant s'adapter à différents types de plateformes lors de la compilation. Il fonctionne donc sur :

- les ordinateurs des particuliers (le PC ou l'ordinateur portable) ;
- les serveurs (données, applications,...) ;
- les ordinateurs portables (les smartphones ou les tablettes) ;
- les systèmes embarqués (ordinateur de voiture) ;
- les éléments actifs des réseaux (routeurs, commutateurs) ;
- les appareils ménagers (téléviseurs, réfrigérateurs,...).

## **Ouvert**

Linux se base sur des standards reconnus ([posix<sup>1</sup>](#), TCP/IP, NFS, Samba ...) permettant de partager des données et des services avec d'autres systèmes d'applications.

### **1.2.4. La philosophie Unix**

- Tout est fichier.
- Portabilité.
- Ne faire qu'une seule chose et la faire bien.
- KISS : Keep It Simple and Stupid.
- "Unix est simple, il faut juste être un génie pour comprendre sa simplicité" (*Dennis Ritchie*)

---

<sup>1</sup> <http://fr.wikipedia.org/wiki/POSIX>

- “Unix est convivial. Cependant Unix ne précise pas vraiment avec qui.” (*Steven King*)

## 1.3. Les distributions GNU/LINUX

Une distribution Linux est un **ensemble cohérent de logiciels** assemblés autour du noyau Linux et prêt à être installé. Il existe des distributions **associatives ou communautaires** (Debian, CentOS) ou **commerciales** (RedHat, Ubuntu).

Chaque distribution propose un ou plusieurs **environnements de bureau**, fournit un ensemble de logiciels pré-installés et une logithèque de logiciels supplémentaires. Des options de configuration (options du noyau ou des services par exemple) sont propres à chacune.

Ce principe permet d'avoir des distributions orientées **débutants** (Ubuntu, Linux Mint ...) ou d'une approche plus complexe (Gentoo, Arch), destinées à faire du **serveur** (Debian, Red Hat, ...) ou dédiées à des **postes de travail**.

### 1.3.1. Les environnements de bureaux

Les environnements graphiques sont nombreux : **Gnome**, **Kde**, Lxde, Xfce, etc. Il y en a pour tous les goûts, et leurs **ergonomies** n'ont pas à rougir de ce que l'on peut retrouver sur les systèmes Microsoft ou Apple !

Alors pourquoi si peu d'engouement pour Linux, alors qu'il **n'existe pas (ou presque pas) de virus pour ce système** ? Parce que tous les éditeurs (Adobe) ou constructeur (NVidia) ne jouent pas le jeu du libre et ne fournissent pas de version de leurs logiciels ou de leurs drivers pour GNU/Linux. Trop peu de jeux également sont (mais plus pour longtemps) distribués sous Linux.

La donne changera-t-elle avec l'arrivée de la steam-box qui fonctionne elle aussi sous Linux ?



Figure 1.4. L'environnement de bureau Gnome

L'environnement de bureau **Gnome 3** n'utilise plus le concept de Bureau mais celui de Gnome Shell (à ne pas confondre avec le shell de la ligne de commande). Il sert à la fois de bureau, de tableau de bord, de zone de notification et de sélecteur de fenêtre. L'environnement de bureau Gnome se base sur la bibliothèque de compostants GTK+.



Figure 1.5. L'environnement de bureau Kde

L'environnement de bureau **KDE** se base sur la bibliothèque de composants **QT**.

Il est traditionnellement plus conseillé aux utilisateurs venant d'un monde Windows.

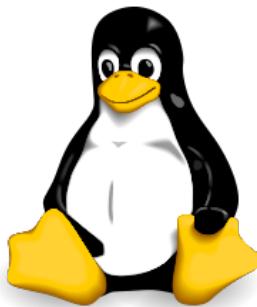


Figure 1.6. Tux, la mascotte Linux

### **1.3.2. Libre / Open source**

Un utilisateur de système d'exploitation Microsoft ou Mac doit s'affranchir d'une licence d'utilisation du système d'exploitation. Cette licence a un coût, même s'il est généralement transparent (le prix de la licence étant inclus dans le prix de l'ordinateur).

Dans le monde GNU/Linux, le mouvement du Libre permet de fournir des distributions gratuites.

Libre ne veut pas dire gratuit !

Open source : les codes sources sont disponibles, il est donc possible de les consulter, de les modifier et de les diffuser.

#### ***Licence GPL (General Public License)***

Cette licence garantit à l'auteur d'un logiciel sa propriété intellectuelle, mais autorise la modification, la rediffusion ou la revente de logiciels par des tiers, sous condition que les codes sources soient fournis avec le logiciel. La licence GPL est la licence issue du projet GNU (Gnu is Not Unix), projet déterminant dans la création de Linux.

Elle implique :

- la liberté d'exécuter le programme, pour tous les usages ;
- la liberté d'étudier le fonctionnement du programme et de l'adapter aux besoins ;
- la liberté de redistribuer des copies ;
- la liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté.

Par contre, même des produits sous licences GPL peuvent être payants. Ce n'est pas le produit en lui-même mais la garantie qu'une équipe de développeurs continue à travailler dessus pour le faire évoluer et dépanner les erreurs, voire fournir un soutien aux utilisateurs.

## 1.4. Les domaines d'emploi

Une distribution Linux excelle pour :

- **Un serveur** : HTTP, messagerie, groupware, partage de fichiers, etc.
- **La sécurité** : Passerelle, parefeu, routeur, proxy, etc.
- **Ordinateur central** : Banques, assurances, industrie, etc.
- **Système embarqué** : Routeurs, Box Internet, SmartTV, etc.

Linux est un choix adapté pour l'hébergement de base de données ou de sites web, ou comme serveur de messagerie, DNS, parefeu, firewall. Bref Linux peut à peu près tout faire, ce qui explique la quantité de distributions spécifiques.

## 1.5. Shell

### 1.5.1. Généralités

Le shell, interface de commandes en français, permet aux utilisateurs d'envoyer des ordres au système d'exploitation. Il est moins visible aujourd'hui, depuis la mise en place des interfaces graphiques, mais reste un moyen privilégié sur les systèmes Linux qui ne possèdent pas tous des interfaces graphiques et dont les services ne possèdent pas toujours une interface de réglage.

Il offre un véritable langage de programmation comprenant les structures classiques : boucles, alternatives et les constituants courants : variables, passage

de paramètres, sous-programmes. Il permet donc la création de scripts pour automatiser certaines actions (sauvegardes, création d'utilisateurs, surveillance du système,...).

Il existe plusieurs types de Shell disponibles et configurables sur une plateforme ou selon le choix préférentiel de l'utilisateur :

- sh, le shell aux normes POSIX ;
- csh, shell orienté commandes en C ;
- bash, Bourne Again Shell, shell de Linux.
- etc, ...

## 1.6. Fonctionnalités

- Exécution de commandes (vérifie la commande passée et l'exécute) ;
- Redirections Entrées/Sorties (renvoi des données dans un fichier au lieu de l'inscrire sur l'écran) ;
- Processus de connexion (gère la connexion de l'utilisateur) ;
- Langage de programmation interprété (permettant la création de scripts) ;
- Variables d'environnement (accès aux informations propres au système en cours de fonctionnement).

### 1.6.1. Principe

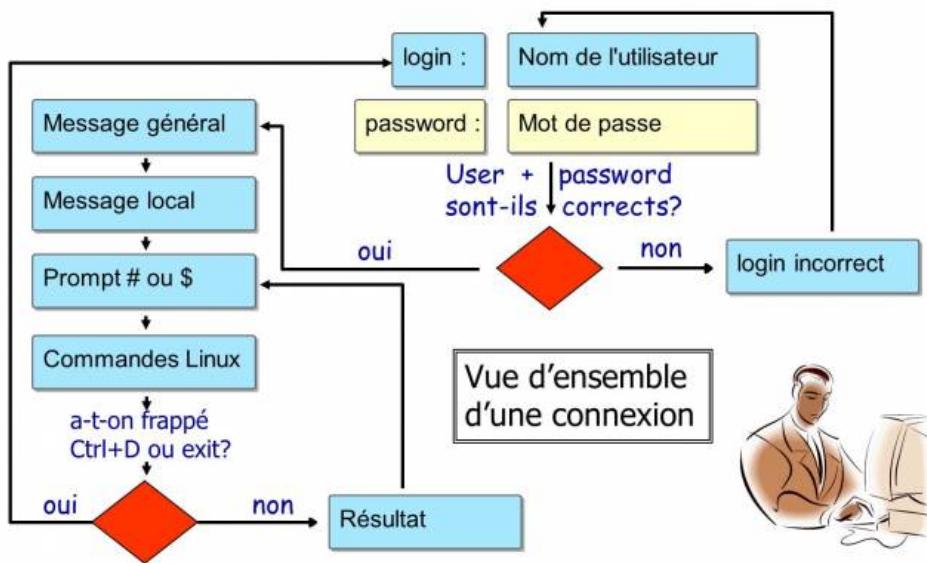


Figure 1.7. Principe de fonctionnement du SHELL

---

# 2

## Commandes pour utilisateurs Linux

---

L'objectif de ce chapitre est d'apprendre aux futurs administrateurs à :

- se **déplacer** dans l'arborescence du système ;
- **créer** un fichier texte, **afficher** son contenu et le **modifier** ;
- utiliser les commandes les plus utiles de Linux.

### 2.1. Généralités

Les systèmes Linux actuels possèdent des utilitaires graphiques dédiés au travail d'un administrateur. Toutefois, il est important d'être capable d'utiliser l'interface en mode ligne de commandes et cela pour plusieurs raisons :

- La majorité des commandes du système sont communes à toutes les distributions Linux, ce qui n'est pas le cas des outils graphiques.
- Il peut arriver que le système ne démarre plus correctement mais qu'un shell de secours reste accessible.
- L'administration à distance se fait en ligne de commandes avec un terminal SSH.
- Afin de préserver les ressources du serveur, l'interface graphique n'est soit pas installée, soit lancée à la demande.
- L'administration se fait par des scripts.

L'apprentissage de ces commandes permet à l'administrateur de se connecter à un terminal Linux, de gérer ses ressources, ses fichiers, d'identifier la station, le terminal et les utilisateurs connectés, etc.

### 2.1.1. Les utilisateurs

L'utilisateur du système Linux est défini (dans le fichier /etc/passwd) par :

- un **nom de connexion**, plus communément appelé « login », ne contenant pas d'espace ;
- un identifiant numérique : **UID** (User Identifier) ;
- un identifiant de groupe : **GID** (Group Identifier) ;
- un **mot de passe**, qui sera chiffré avant d'être stocké ;
- un **interpréteur de commandes**, un Shell, qui peut être différent d'un utilisateur à l'autre ;
- un **répertoire de connexion**, le « home directory » ;
- un **prompt de connexion**, qui sera symbolisé par un # pour les administrateurs et un \$ pour les autres utilisateurs.

En fonction de la politique de sécurité mise en œuvre sur le système, le mot de passe devra comporter un certain nombre de caractères et respecter des exigences de complexité.

Parmi les Shells existants, le **Bourne Again Shell** (/bin/bash) est le Shell le plus fréquemment utilisé. Il est affecté par défaut aux nouveaux utilisateurs. Pour diverses raisons, des utilisateurs avancés de Linux choisiront des Shells alternatifs parmi le Ksh, le Csh, etc.

Le répertoire de connexion de l'utilisateur est par convention stocké dans le répertoire /home du poste de travail. Il contiendra les données personnelles de l'utilisateur. Par défaut, à la connexion, le répertoire de connexion est sélectionné comme répertoire courant.

Une installation type poste de travail (avec interface graphique) démarre cette interface sur le terminal 1. Linux étant multi-utilisateurs, il est possible de connecter plusieurs utilisateurs plusieurs fois, sur des **terminaux physiques** (TTY) ou **virtuels** (PTS) différents. Les terminaux virtuels sont disponibles au sein d'un environnement graphique. Un utilisateur bascule d'un terminal physique à l'autre à l'aide des touches **Alt+Fx** depuis la ligne de commande ou à l'aide des touches **Ctrl+Alt+Fx**.

### 2.1.2. Le Shell

Une fois que l'utilisateur est connecté sur une console, le Shell affiche le prompt. Il se comporte ensuite comme une boucle infinie, à chaque saisie d'instruction :

- affichage du prompt ;
- lecture de la commande ;
- analyse de la syntaxe ;
- substitution des caractères spéciaux ;
- exécution de la commande ;
- affichage du prompt ;
- etc.

La séquence de touche **Ctrl+C** permet d'interrompre une commande en cours d'exécution.

L'utilisation d'une commande respecte généralement cette séquence :

#### Séquence d'une commande.

```
commande [option(s)] [arguments(s)]
```

Le nom de la commande est **toujours en minuscules**.

Un espace sépare chaque élément.

Les **options courtes** commencent par un tiret (-l), alors que les **options longues** commencent par deux tirets (--list). Un double tiret (--) indique la fin de la liste d'options. Il est possible de regrouper certaines options courtes :

#### Options courtes.

```
[root]# ls -l -i -a
```

est équivalent à :

#### Regroupement d'options.

```
[root]# ls -lia
```

Il peut bien entendu y avoir plusieurs arguments après une option :

### **Arguments d'une commande.**

```
[root]# ls -lia /etc /home /var
```

Dans la littérature, le terme « option » est équivalent au terme « paramètre », plus utilisé dans le domaine de la programmation. Le côté optionnel d'une option ou d'un argument est symbolisé en le mettant entre crochets [ et ]. Lorsque plusieurs options sont possibles, une barre verticale appelée « pipe » les sépare [a|e|i].

## **2.2. Les commandes générales**

### **2.2.1. Les commandes *man* et *whatis***

Il est impossible pour un administrateur, quel que soit son niveau, de connaître toutes les commandes et options dans les moindres détails. Une commande a été spécialement conçue pour accéder en ligne de commande à un ensemble d'aides, sous forme d'un manuel : la commande *man* (« le man est ton ami »).

Ce manuel est divisé en 8 sections, regroupant les informations par thème, la section par défaut étant la section 1 :

1. Commande utilisateur ;
2. Appels système ;
3. Fonctions de bibliothèque C ;
4. Périphériques et fichiers spéciaux ;
5. Formats de fichiers ;
6. Jeux ;
7. Divers ;
8. Outils d'administration système et démons.

Des informations sur chaque section sont accessibles en saisissant *man x intro*, x indiquant le numéro de section.

La commande :

### **Syntaxe de la commande *man*.**

```
[root]# man passwd
```

informera l'administrateur sur la commande passwd, ses options, etc. Alors qu'un :

#### Syntaxe de la commande man avec section.

```
[root]# man 5 passwd
```

l'informera sur les fichiers en relations avec la commande.

Toutes les pages du manuel ne sont pas traduites de l'anglais. Elles sont toutefois généralement très précises et fournissent toutes les informations utiles. La syntaxe utilisée et le découpage peut dérouter l'administrateur débutant, mais avec de la pratique, l'administrateur y retrouvera rapidement l'information qu'il recherche.

La navigation dans le manuel se fait avec les flèches « Haut » et « Bas ». Le manuel se quitte en appuyant sur la touche « q ».

La commande **whatis** permet de faire une recherche par mot clef au sein des pages de manuel :

#### Syntaxe de la commande whatis.

```
[root]# whatis clear
```

### 2.2.2. La commande shutdown

La commande **shutdown** permet de **stopper électriquement**, immédiatement ou après un certain laps de temps, un serveur Linux.

#### Syntaxe de la commande shutdown.

```
[root]# shutdown [-h] [-r] heure [message]
```

L'heure d'arrêt est à indiquer au format **hh:mm** pour une heure précise, ou **+mm** pour un délai en minutes.

Pour forcer un arrêt immédiat, le mot « **now** » remplacera l'heure. Dans ce cas, le message optionnel n'est pas envoyé aux autres utilisateurs du système.

## Exemples

### Exemples de la commande shutdown.

```
[root]# shutdown -h 0:30 "Arrêt du serveur à 0h30"  
[root]# shutdown -r +5
```

## Options

Tableau 2.1. Options de la commande shutdown

Options	Observations
-h	Arrête le système électriquement
-r	Redémarre le système

### 2.2.3. La commande history

La commande **history** permet d'afficher l'historique des commandes qui ont été saisies par l'utilisateur.

Les commandes sont mémorisées dans le fichier **.bash\_history** du répertoire de connexion de l'utilisateur.

### Exemple de commande history.

```
[root]# history  
147 man ls  
148 man history
```

Tableau 2.2. Options de la commande history

Options	Commentaires
-w	L'option -w permet d'y copier l'historique de la session en cours.
-c	L'option -c effacera l'historique de la session en cours (mais pas le contenu du fichier <b>.bash_history</b> ).

## Manipuler l'historique

Pour manipuler l'historique, des commandes permettent depuis le prompt de :

Touches	Fonction
!!	Rappeler la dernière commande passée.
!n	Rappeler la commande par son numéro dans la liste.
!string	Rappeler la commande la plus récente commençant par la chaîne de caractères.
[↑]	Remonter l'historique des commandes.
[↓]	Redescendre l'historique des commandes.

## L'auto-complétion

L'auto-complétion est également d'une aide précieuse.

- Elle permet de compléter les commandes, les chemins saisis ou les noms de fichiers.
- Un appui sur la touche **[TAB]** complète la saisie dans le cas d'une seule solution.
- Sinon, il faudra faire un deuxième appui pour obtenir la liste des possibilités.

Si un double appui sur la touche **[TAB]** ne provoque aucune réaction de la part du système, c'est qu'il n'existe aucune solution à la complétion en cours.

## 2.3. Affichage et identification

### 2.3.1. La commande clear

La commande clear permet d'effacer le contenu de l'écran du terminal. En réalité, pour être plus précis, elle permet de décaler l'affichage de sorte que le prompt se retrouve en haut de l'écran sur la première ligne.

Dans un terminal, l'affichage sera définitivement masqué tandis que dans une interface graphique, un ascenseur permettra de remonter dans l'historique du terminal virtuel.

### 2.3.2. La commande echo

La commande echo permet d'afficher une chaîne de caractères.

Cette commande est plus particulièrement utilisée dans les scripts d'administration pour informer l'utilisateur pendant l'exécution.

L'option `-n` permet de ne pas revenir à la ligne après avoir affiché le texte (ce qui est le comportement par défaut de la commande).

Pour diverses raisons, le développeur du script peut être amené à utiliser des séquences spéciales (commençant par un caractère `\`). Dans ce cas, l'option `-e` sera stipulée, permettant l'interprétation des séquences.

Parmi les séquences fréquemment utilisées, nous citerons :

Tableau 2.3. Séquences spéciales de la commande echo

Séquence	Résultat
<code>\a</code>	Émet un bip sonore
<code>\b</code>	Retour en arrière
<code>\n</code>	Ajoute un saut de ligne
<code>\t</code>	Ajoute une tabulation horizontale
<code>\v</code>	Ajoute une tabulation verticale

### 2.3.3. La commande date

La commande date permet d'afficher la date et l'heure. La commande respecte la syntaxe suivante :

#### Syntaxe de la commande date.

```
[root]# date [-d AAAAMMJJ] [format]
```

Exemples :

```
[root]# date  
mer. Avril 17 16:46:53 CEST 2013  
[root]# date -d 20150729 +%j  
210
```

Dans ce dernier exemple, l'option `-d` affiche une date donnée. L'option `+%j` formate cette date pour n'afficher que le quantième.

Attention : Le format d'une date peut changer suivant la valeur de la langue définie dans la variable d'environnement \$LANG.

L'affichage de la date peut suivre les formats suivants :

**Tableau 2.4. Formats de la commande date**

Option	Format
+%A	Nom complet du jour
+%B	Nom complet du mois
+%c	Affichage complet de la date
+%d	Numéro du jour
+%F	Date au format AAAA-MM-JJ
+%G	Année
+%H	Heure
+%j	Quantième du jour
+%m	Numéro du mois
+%M	Minute
+%R	Heure au format hh:mm
+%s	Secondes depuis le 1er janvier 1970
+%T	Heure au format hh:mm:ss
+%u	Jour de la semaine (1 pour lundi)
+%V	Numéro de la semaine
+%x	Date au format JJ/MM/AAAA

La commande date permet également de modifier la date et l'heure système. Dans ce cas, l'option –s sera utilisée.

```
[root]# date -s "2013-04-17 10:19"
jeu. Avril 17 10:19:00 CEST 2013
```

Le format à respecter pour l'argument suivant l'option –s est celui-ci :

```
date -s "[AA]AA-MM-JJ hh:mm:[ss]"
```

### 2.3.4. Les commandes id, who et whoami

La commande id affiche le nom de l'utilisateur courant et ses groupes ou ceux d'un utilisateur, si le login de celui-ci est fourni comme argument.

```
[root]# id util1  
uid=501(util1) gid=501(group1) groups=501(group1),502(group2)
```

Les options -g, -G, -n et -u affiche respectivement le GID du groupe principal, les GID des groupes secondaires, les noms au lieu des identifiants numériques et l'UID de l'utilisateur.

La commande whoami affiche le login de l'utilisateur courant.

La commande who seule affiche le nom des utilisateurs connectés :

```
[root]# who  
root tty1 2014-09-15 10:30  
root pts/0 2014-09-15 10:31
```

Linux étant multi-utilisateurs, il est probable que plusieurs sessions soient ouvertes sur la même station, que ce soit physiquement ou à travers le réseau. Il est intéressant de savoir quels utilisateurs sont connectés, ne serait-ce que pour communiquer avec eux par l'envoi de messages.

- tty : représente un terminal.
- pts/ : représente une console virtuelle sous environnement graphique.

L'option « -r » affiche en plus le niveau d'exécution (voir chapitre « démarrage »).

## 2.4. Arborescence de fichiers

Sous Linux, l'arborescence des fichiers se présente sous la forme d'un arbre inversé, appelé **arborescence hiérarchique unique**, dont la racine est le répertoire « / ».

Le **répertoire courant** est le répertoire où se trouve l'utilisateur.

Le **répertoire de connexion** est le répertoire de travail associé à l'utilisateur. Les répertoires de connexion sont, en standard, stockés dans le répertoire **/home**.

À la connexion de l'utilisateur, le répertoire courant est le répertoire de connexion.

Un **chemin absolu** référence un fichier depuis la racine en parcourant l'arborescence complète jusqu'au niveau du fichier :

- /home/groupeA/alice/monfichier

Le **chemin relatif** référence ce même fichier en parcourant l'arborescence complète depuis le répertoire courant :

- ../alice/monfichier

Dans l'exemple précédent, les “..” font référence au répertoire parent du répertoire actuel.

Un répertoire, même s'il est vide, contiendra obligatoirement au minimum **deux références** :

- « . » : référence sur lui-même.
- « .. » : référence le répertoire parent du répertoire actuel.

Un chemin relatif peut ainsi commencer par « ./ » ou par « ../ ». Lorsque le chemin relatif fait référence à un sous dossier ou à un fichier du répertoire courant, alors le « ./ » est souvent omis. Mentionner le premier « ./ » de l'arborescence ne sera réellement requis que pour lancer un fichier exécutable.

Les erreurs dans les chemins peuvent être la cause de nombreux problèmes : création de dossier ou de fichiers aux mauvais endroits, suppressions involontaires, etc. Il est donc fortement recommandé d'utiliser l'auto-complétion (cf. 2.2) lors des saisies de chemin.

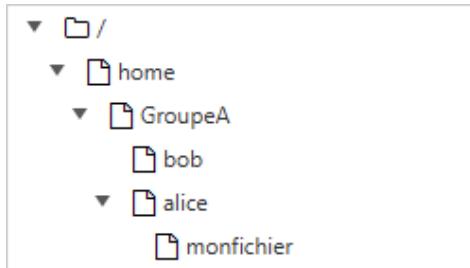


Figure 2.1. Notre arborescence exemple

Dans l'exemple ci-dessus, nous cherchons à donner l'emplacement du fichier monfichier depuis le répertoire de bob.

- Par un **chemin absolu**, le répertoire courant importe peu. Nous commençons par la racine, pour descendre successivement dans les répertoires "home", "groupeA", "alice" et enfin le fichier "monfichier" : `/home/groupeA/alice/monfichier`.
- Par un **chemin relatif**, notre point de départ étant le répertoire courant "bob", nous remontons d'un niveau par ".." (soit dans le répertoire groupeA), puis nous descendons dans le répertoire "alice", et enfin le fichier "monfichier" : `../alice/monfichier`.

#### **2.4.1. La commande `pwd`**

La commande `pwd` (Print Working Directory) affiche le chemin absolu du répertoire courant.

```
[root]# pwd  
/root/instructeur
```

Pour se déplacer à l'aide d'un chemin relatif, il faut impérativement connaître son positionnement dans l'arborescence.

Selon le shell, le prompt peut également afficher le nom du répertoire courant.

#### **2.4.2. La commande `cd`**

La commande `cd` (Change Directory) permet de changer le répertoire courant, autrement dit, de se déplacer dans l'arborescence.

```
[root]# cd /tmp  
[root]# pwd  
/tmp  
[root]# cd ../  
[root]# pwd  
/  
[root]# cd  
[root]# pwd  
/root
```

Comme vous pouvez le constater dans le dernier exemple ci-dessus, la commande cd sans argument permet de repositionner le répertoire courant sur le répertoire de connexion (home directory).

### **2.4.3. La commande ls**

La commande ls affiche le contenu d'un répertoire.

#### **Syntaxe de la commande ls.**

```
ls [-a] [-i] [-l] [répertoire1] [répertoire2] [...]
```

Exemple :

```
[root]# ls /home  
... STAGE
```

#### **Options**

Les options principales de la commande ls sont :

**Tableau 2.5. Options principales de la commande ls**

Option	Information
-a	Affiche tous les fichiers, même ceux cachés. Les fichiers cachés sous Linux sont ceux qui commencent par un “.”.
-i	Affiche les numéros d'inode.
-l	Affiche sous forme de liste verticale la liste des fichiers avec des informations supplémentaires formatées par colonnes.

La commande ls offre toutefois de très nombreuses options (voir le man) :

**Tableau 2.6. Options complémentaires de la commande ls**

Option	Information
-d	Affiche les informations d'un répertoire au lieu de lister son contenu.
-g	Affiche les UID et GID plutôt que les noms des propriétaires.

Option	Information
-h	Affiche les tailles de fichiers dans le format le plus adapté (octet, kilo-octet, méga-octet, giga-octet, ...). h pour Human Readable.
-s	Affiche la taille en octets (sauf si option k).
-A	Affiche tous les fichiers du répertoire sauf “.” et “..”.
-R	Affiche récursivement le contenu des sous répertoires.
-F	Affiche le type des fichiers. Imprime un / pour un répertoire, * pour les exécutables, @ pour un lien symbolique, et rien pour un fichier texte.

## Description des colonnes

```
[root]# ls -lia /root
78489 drwxr-xr-x 4 root root 4096 25 oct. 08:10 STAGE
```

Tableau 2.7. Description des colonnes du résultat généré par la commande ls

Valeur	Information.
78489	Numéro d'inode.
drwxr-xr-x	Type de fichier (d) et droits (rwxr-xr-x).
4	Nombre de sous-répertoires (“.” et “..” inclus). Pour un fichier de type lien physique : nombre de liens physiques.
root	Utilisateur propriétaire.
root	Groupe propriétaire.
4096	Taille en octets.
25 oct. 08:10	Date de dernière modification.
STAGE	Nom du fichier (ou du répertoire).

## Alias

Des alias sont fréquemment positionnés au sein des distributions courantes.

C'est le cas de l'alias ll :

**Alias de la commande ls -l.**

```
alias ll='ls -l --color=auto'
```

***Utilisations avancées***

- Lister les fichiers de **/etc** par ordre de dernière modification :

```
[root]# ls -ltr /etc
total 1332
-rw-r--r--. 1 root root    662 29 aout   2007 logrotate.conf
-rw-r--r--. 1 root root    272 17 nov.   2009 mailcap
-rw-----. 1 root root    122 12 janv.  2010 securetty
...
-rw-r--r--. 2 root root     85 18 nov.  17:04 resolv.conf
-rw-r--r--. 1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--. 1 root root    283 18 nov.  17:05 mtab
```

- Lister les fichiers de **/var** plus gros qu'un mega-octet mais moins qu'un giga-octets :

```
[root]# ls -Rlh /var | grep [0-9]M
...
-rw-r--r--. 1 apache apache 1,2M 10 nov. 13:02 XB RiyazBdIt.ttf
-rw-r--r--. 1 apache apache 1,2M 10 nov. 13:02 XB RiyazBd.ttf
-rw-r--r--. 1 apache apache 1,1M 10 nov. 13:02 XB RiyazIt.ttf
...
```

- Afficher les droits sur un dossier :

Pour connaître les droits sur un dossier, dans notre exemple **/etc**, la commande suivante ne conviendrait pas :

```
[root]# ls -l /etc
total 1332
-rw-r--r--. 1 root root    44 18 nov. 17:04 adjtime
-rw-r--r--. 1 root root   1512 12 janv. 2010 aliases
-rw-r--r--. 1 root root  12288 17 nov. 17:41 aliases.db
drwxr-xr-x. 2 root root   4096 17 nov. 17:48 alternatives
...
```

puisque cette dernière liste par défaut le contenu du dossier et non le contenant.

Pour ce faire, il faut utiliser l'option -d :

```
[root]# ls -ld /etc  
drwxr-xr-x. 69 root root 4096 18 nov. 17:05 /etc
```

- Lister les fichiers par taille :

```
[root]# ls -lhS
```

- Afficher la date de modification au format “timestamp” :

```
[root]# ls -l --time-style="+%Y-%m-%d $newline%m-%d %H:%M"  
total 12378  
dr-xr-xr-x. 2 root root 4096 2014-11-23 11-23 03:13 bin  
dr-xr-xr-x. 5 root root 1024 2014-11-23 11-23 05:29 boot
```

- Ajouter le “trailing slash” à la fin des dossiers :

Par défaut, la commande ls n'affiche pas le dernier slash d'un dossier.

Dans certains cas, comme pour des scripts par exemple, il est utile de les afficher :

```
[root]# ls -dF /etc  
/etc/
```

#### **2.4.4. La commande mkdir**

La commande mkdir crée un répertoire ou une arborescence de répertoire.

##### **Syntaxe de la commande mkdir.**

```
mkdir [-p] repertoire [repertoire] [...]
```

Exemple :

```
[root]# mkdir /home/STAGE/travail
```

Le répertoire « STAGE » devra exister pour créer le répertoire « travail ».

Sinon, l'option « –p » devra être utilisée. L'option « –p » crée les répertoires parents s'ils n'existent pas.



Il est vivement déconseillé de donner des noms de commandes UNIX comme noms de répertoires ou fichiers.

#### **2.4.5. La commande touch**

La commande touch modifie l'horodatage d'un fichier ou crée un fichier vide si le fichier n'existe pas.

##### **Syntaxe de la commande touch.**

```
touch [-t date] fichier
```

Exemple :

```
[root]# touch /home/STAGE/fichier
```

Option	Information
-t date	Modifie la date de dernière modification du fichier avec la date précisée. Date au format : [AAAA]MMJJhhmm[ss]



La commande touch est utilisée en priorité pour créer un fichier vide, mais elle peut avoir un intérêt dans le cadre de sauvegarde incrémentale ou différentielle. En effet, le fait d'exécuter un touch sur un fichier aura pour seul effet de forcer sa sauvegarde lors de la sauvegarde suivante.

#### **2.4.6. La commande rmdir**

La commande rmdir supprime un répertoire vide.

Exemple :

```
[root]# rmdir /home/STAGE/travail
```

Option	Information
-p	Supprime le ou les répertoire(s) parent(s) à la condition qu'ils soient vides.



Pour supprimer à la fois un répertoire non-vide et son contenu, il faudra utiliser la commande rm.

#### 2.4.7. La commande rm

La commande rm supprime un fichier ou un répertoire.

##### Syntaxe de la commande rm.

```
rm [-f] [-r] fichier [fichier] [...]
```



ATTENTION !!! Toute suppression de fichier ou de répertoire est définitive.

Tableau 2.8. Options de la commande rm

Options	Information
-f	Ne demande pas de confirmation de la suppression.
-i	Demande de confirmation de la suppression.
-r	Supprime récursivement les sous-répertoires.



La commande rm en elle-même ne demande pas de confirmation lors de la suppression de fichiers. Ce comportement est propre à la distribution RedHat/CentOS.

La commande rm est ici un alias de la commande rm - i. Ne soyez pas surpris sur une autre distribution, type Debian par exemple, de ne pas obtenir de demande de confirmation.

La suppression d'un dossier à l'aide de la commande rm, que ce dossier soit vide ou non, nécessitera l'ajout de l'option -r.

La fin des options est signalée au shell par un double tiret “--”.

Dans l'exemple :

```
[root]# >-dur-dur # Creer un fichier vide appelé -dur-dur  
[root]# rm -f -- -dur-dur
```

Le nom du fichier -dur-dur commence par un “-”. Sans l'usage du “--” le shell aurait interprété le “-d” de “-dur-dur” comme une option.

### 2.4.8. La commande mv

La commande mv déplace et renomme un fichier.

#### Syntaxe de la commande mv.

```
mv fichier [fichier ...] destination
```

Exemples :

```
[root]# mv /home/fic1 /home/fic2  
[root]# mv /home/fic1 /home/fic2 /tmp
```

Tableau 2.9. Options de la commande mv

Options	Information
-f	Ne demande pas de confirmation si écrasement du fichier de destination.
-i	Demande de confirmation si écrasement du fichier de destination (par défaut).

#### Cas concrets

```
[root]# mv /home/fic1 /home/fic2
```

Permet de renommer “fic1” en “fic2”, si “fic2” existe déjà, il sera remplacé par “fic1”.

```
[root]# mv /home/fic1 /home/fic2 /tmp
```

Permet de déplacer “fic1” et “fic2” dans le répertoire “/tmp”.

```
[root]# mv fic1 /repexiste/fic2
```

« fic1 » est déplacé dans « /repexiste » et renommé « fic2 ».

```
[root]# mv fic1 fic2
```

« fic1 » est renommé « fic2 ».

```
[root]# mv fic1 /repexiste
```

Si le répertoire de destination existe, « fic1 » est déplacé dans « /repexiste ».

```
[root]# mv fic1 /repexistepas
```

Si le répertoire de destination n'existe pas, « fic1 » est renommé « repexistepas » à la racine.

#### 2.4.9. La commande cp

La commande cp copie un fichier.

##### Syntaxe de la commande cp.

```
cp fichier [fichier ...] destination
```

Exemple :

```
[root]# cp -r /home/STAGE /tmp
```

Tableau 2.10. Options de la commande cp

Options	Information
-i	Demande de confirmation si écrasement (par défaut).
-f	Ne demande pas de confirmation si écrasement du fichier de destination.
-p	Conserve le propriétaire, les permissions et l'horodatage du fichier copié.

Options	Information
-r	Copie un répertoire avec ses fichiers et sous-réertoires.

## Cas concrets

```
[root]# cp fic1 /repexiste/fic2
```

« fic1 » est copié dans « /repexiste » sous le nom « fic2 ».

```
[root]# cp fic1 fic2
```

« fic1 » est copié sous le nom « fic2 » dans ce répertoire.

```
[root]# cp fic1 /repexiste
```

Si le répertoire de destination existe, « fic1 » est copié dans « /repexiste ».

```
[root]# cp fic1 /repexistepas
```

Si le répertoire de destination n'existe pas, « fic1 » est copié sous le nom « repexistepas ».

## 2.5. Visualisation

### 2.5.1. La commande file

La commande file affiche le type d'un fichier.

**Syntaxe de la commande file.**

```
file fichier [fichiers]
```

Exemple :

```
[root]# file /etc/passwd /etc
/etc/passwd: ASCII text
/etc: directory
```

### 2.5.2. La commande more

La commande more affiche le contenu d'un ou de plusieurs fichiers écran par écran.

#### Syntaxe de la commande more.

```
more fichier [fichiers]
```

Exemple :

```
[root]# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
```

En utilisant la touche [ENTREE], le déplacement se fait ligne par ligne. En utilisant la touche [ESPACE], le déplacement se fait page par page.

### 2.5.3. La commande less

La commande less affiche le contenu d'un ou de plusieurs fichiers. La commande less est interactive et possède des commandes d'utilisation qui lui sont propres.

#### Syntaxe de la commande less.

```
less fichiers [fichiers]
```

Les commandes propres à less sont :

Tableau 2.11. Commandes internes à less

Commande	Action
h	Aide.
Flèches	Monter, descendre d'une ligne ou pour aller à droite ou à gauche.
Entrée	Descendre d'une ligne.
Espace	Descendre d'une page.
PgAR ou PgAV	Monter ou descendre d'une page.

Commande	Action
Pos1 ou Fin	Se placer en début de fichier ou en fin de fichier.
/texte	Rechercher le texte.
q	Quitter la commande less.

## 2.5.4. Les commandes cat et tac

### La commande cat

La commande cat concatène (mettre bout à bout) le contenu de plusieurs fichiers et affiche le résultat sur la sortie standard.

#### Syntaxe de la commande cat.

```
cat fichier [fichiers]
```

Exemple 1 - Afficher le contenu d'un fichier vers la sortie standard :

```
[root]# cat /etc/passwd
```

Exemple 2 - Afficher le contenu de plusieurs fichiers vers la sortie standard :

```
[root]# cat /etc/passwd /etc/group
```

Exemple 3 - Afficher le contenu de plusieurs fichiers et rediriger la sortie standard :

```
[root]# cat /etc/passwd /etc/group > utilisateursEtGroupes.txt
```

Exemple 4 - Afficher la numérotation des lignes :

```
[root]# cat -n /etc/passwd
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
...
```

Exemple 5 - Affiche la numérotation des lignes non vides :

```
[root]# cat -b /etc/openldap/ldap.conf
```

```
1 #
2 # LDAP Defaults
3 #

4 # See ldap.conf(5) for details
5 # This file should be world readable but not world writable
```

## La commande tac

La commande tac fait quasiment l'inverse de la commande cat. Elle affiche le contenu d'un fichier en commençant par la fin (ce qui est particulièrement intéressant pour la lecture des logs !).

Exemple : Afficher un fichier de logs en affichant en premier la dernière ligne :

```
[root]# tac /var/log/messages | less
```

### 2.5.5. La commande head

La commande head affiche le début d'un fichier.

#### Syntaxe de la commande head.

```
head [-n x] fichier
```

Tableau 2.12. Options de la commande head

Option	Observation
-n x	Affiche les x premières lignes du fichier

Par défaut (sans l'option –n), la commande head affichera les 10 premières lignes du fichier.

### 2.5.6. La commande tail

La commande tail affiche la fin d'un fichier.

#### Syntaxe de la commande tail.

```
tail [-f] [-n x] fichier
```

**Tableau 2.13. Options de la commande tail**

Option	Observation
-n x	Affiche les x dernières lignes du fichier
-f	Affiche les modifications du fichier en temps réel

Exemple :

```
[root]# tail -n 3 /etc/passwd
sshd:x:74:74:Privilege-separated sshd:/var/empty /sshd:/sbin/nologin
tcpdump:x:72:72:::/sbin/nologin
user1:x:500:500:grp1:/home/user1:/bin/bash
```

Avec l'option -f, la commande tail ne rend pas la main et s'exécute tant que l'utilisateur ne l'interrompt pas par la séquence [CTRL] + [C]. Cette option est très fréquemment utilisée pour suivre les fichiers journaux (les logs) en temps réel.

Sans l'option -n, la commande tail affiche les 10 dernières lignes du fichier.

### 2.5.7. La commande sort

La commande sort trie les lignes d'un fichier.

Elle permet d'ordonner, ranger dans un ordre donné, le résultat d'une commande ou le contenu d'un fichier, selon un ordre numérique, alphabétique, par ordre de grandeur (Ko, Mo, Go) ou dans l'ordre inverse.

#### Syntaxe de la commande sort.

```
sort [-kx] [-n] [-o fichier] [-ty] fichier
```

Exemple :

```
[root]# sort -k3 -t: -n /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

**Tableau 2.14. Options de la commande sort**

Option	Observation
-kx	Précise la colonne x sur laquelle se fera le tri

Option	Observation
-n	Demande un tri numérique
-o fichier	Enregistre le tri dans le fichier précisé
-ty	Précise le caractère séparateur de champs y
-r	Inverse l'ordre du résultat

La commande sort ne trie le fichier qu'à l'affichage écran. Le fichier n'est pas modifié par le tri. Pour enregistrer le tri, il faut utiliser l'option –o ou une redirection de sortie >.

Par défaut, le tri des nombres se fait selon leur caractère. Ainsi, “110” sera avant “20”, qui sera lui-même avant “3”. Il faut préciser l'option –n pour que les blocs caractères numériques soient bien triés par leur valeur.

### ***Inverser l'ordre des résultats***

La commande sort permet d'inverser l'ordre des résultats, avec l'option -r :

```
[root]# sort -k3 -t: -n -r /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

rangera cette fois-ci le contenu du fichier /etc/passwd du plus grand uid au plus petit.

### ***Mélanger les valeurs***

La commande sort permet également de mélanger les valeurs avec l'option -R :

```
[root]# sort -R /etc/passwd
```

### ***Trier des adresses IP***

Un administrateur système est rapidement confronté au traitement des adresses IP issues des logs de ses services comme SMTP, VSFTP ou Apache. Ces adresses sont typiquement extraites avec la commande cut.

Voici un exemple avec le fichier client-dns.txt :

```
192.168.1.10
192.168.1.200
5.1.150.146
208.128.150.98
208.128.150.99
```

```
[root]# sort -nr client-dns.txt
208.128.150.99
208.128.150.98
192.168.1.200
192.168.1.10
5.1.150.146
```

### Trier des tailles de fichiers

Sort sait reconnaître les tailles de fichiers, issues de commande comme ls avec l'option -h.

Voici un exemple avec le fichier taille.txt :

```
1, 7G
18M
69K
2, 4M
1, 2M
4, 2G
6M
124M
12, 4M
4G
```

```
[root]# sort -hr taille.txt
4, 2G
4G
1, 7G
124M
18M
12, 4M
6M
2, 4M
1, 2M
```

69K

### 2.5.8. La commande wc

La commande wc compte le nombre de lignes, mots ou octets d'un fichier.

#### Syntaxe de la commande wc.

```
wc [-l] [-m] [-w] fichier [fichiers]
```

Tableau 2.15. Options de la commande wc

Option	Observation
-c	Compte le nombre d'octets.
-m	Compte le nombre de caractères.
-l	Compte le nombre de lignes.
-w	Compte le nombre de mots.

## 2.6. Recherche

### 2.6.1. La commande find

La commande find recherche l'emplacement d'un fichier.

#### Syntaxe de la commande find.

```
find répertoire [-name nom] [-type type] [-user login] [-date date]
```

Les options de la commande find étant très nombreuses, il est préférable de se référer au man.

Si le répertoire de recherche n'est pas précisé, la commande find cherchera à partir du répertoire courant.

Tableau 2.16. Options de la commande find

Option	Observation
-perm permissions	Recherche des fichiers selon leurs permissions.

Option	Observation
-size taille	Recherche des fichiers selon leur taille.

### L'option -exec

Il est possible d'utiliser l'option –exec pour exécuter une commande à chaque ligne de résultat :

```
[root]# find /tmp -name *.log -exec rm -f {} \;
```

La commande précédente recherche tous les fichiers du répertoire /tmp nommés \*.log et les supprime.

### Comprendre l'option -exec

Dans l'exemple ci-dessus, la commande find va construire une chaîne de caractères représentant la commande à exécuter.

Si la commande find trouve trois fichiers nommés log1.log, log2.log et log3.log, alors la commande find va construire la chaîne en remplaçant dans la chaîne “rm -f {} \;” les accolades par un des résultats de la recherche, et cela autant de fois qu'il y a de résultats.

Ce qui nous donnera :

```
rm -f /tmp/log1 ; rm -f /tmp/log2 ; rm -f /tmp/log3 ;
```

Le caractère “;” est un caractère spécial du Shell qui doit être protégé par un “\” pour éviter son interprétation trop tôt par la commande find (et non plus dans le exec).

### 2.6.2. La commande whereis

La commande whereis recherche des fichiers liés à une commande.

#### Syntaxe de la commande whereis.

```
whereis [-b] [-m] [-s] commande
```

Exemple :

```
[root]# whereis -b ls  
ls: /bin/ls
```

Tableau 2.17. Options de la commande whereis

Option	Observation
-b	Ne recherche que le fichier binaire.
-m	Ne recherche que les pages de manuel.
-s	Ne recherche que les fichiers sources.

### 2.6.3. La commande grep

La commande grep recherche une chaîne de caractères dans un fichier.

#### Syntaxe de la commande grep.

```
grep [-w] [-i] [-v] "chaîne" fichier
```

Exemple :

```
[root]# grep -w "root:" /etc/passwd  
root:x:0:0:root:/root:/bin/bash
```

Tableau 2.18. Options de la commande grep

Option	Observation
-i	Ignore la casse de la chaîne de caractères recherchée.
-v	Inverse le résultat de la recherche.
-w	Recherche exactement la chaîne de caractères précisée.

La commande grep retourne la ligne complète comprenant la chaîne de caractères recherchée.

- Le caractère spécial ^ permet de rechercher une chaîne de caractères placée en début de ligne.
- Le caractère spécial \$ permet de rechercher une chaîne de caractères placée en fin de ligne.

```
[root]# grep -w "^root" /etc/passwd
```



Cette commande est très puissante et il est fortement conseillé de consulter son manuel. Elle a de nombreux dérivés,

## ***Recherche récursive***

Il est possible de rechercher une chaîne de caractères dans une arborescence de fichiers avec l'option -R.

```
[root]# grep -R "Virtual" /etc/httpd
```

### ***2.6.4. Les méta-caractères***

Les méta-caractères se substituent à un ou plusieurs caractères (voire à une absence de caractère) lors d'une recherche.

Ils sont combinables.

Le caractère \* remplace une chaîne composée de plusieurs caractères quelconques. Le caractère \* peut également représenter une absence de caractère.

```
[root]# find /home -name test*
/home/test
/home/test1
/home/test11
/home/tests
/home/test362
```

Les méta-caractères permettent des recherches plus complexes en remplaçant tout ou partie d'un mot. Il suffit de remplacer les inconnues par ces caractères spéciaux.

Le caractère "?" remplace un unique caractère, quel qu'il soit.

```
[root]# find /home -name test?
/home/test1
```

```
/home/tests
```

Les crochets “[ ]” permettent de spécifier les valeurs que peut prendre un unique caractère.

```
[root]# find /home -name test[123]*
/home/test1
/home/test11
/home/test362
```



Il ne faut pas confondre les méta-caractères du shell et ceux des expressions régulières. La commande grep utilise les méta-caractères des expressions régulières.

## 2.7. Redirections et tubes

### 2.7.1. L'entrée et les sorties standards

Sur les systèmes UNIX et Linux, les flux standards sont aux nombres de trois. Ils permettent aux programmes, via la bibliothèque stdio.h de faire entrer ou sortir des informations.

Ces flux sont appelés canal X ou descripteur X de fichier.

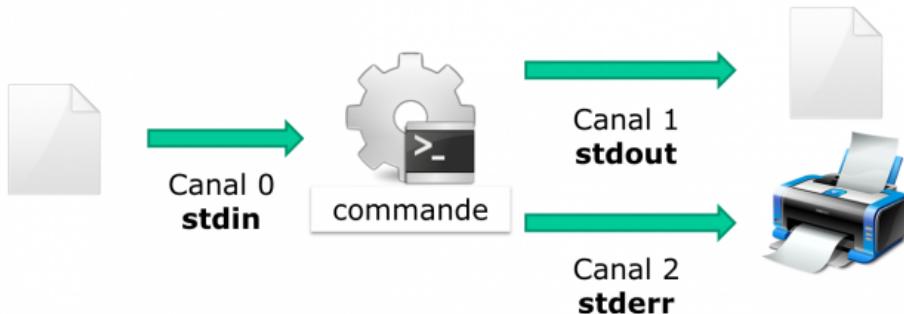
Par défaut :

- le clavier est le périphérique d'entrée pour le canal 0, appelé stdin ;
- l'écran est le périphérique de sortie pour les canaux 1 et 2, appelés stdout et stderr.



stderr reçoit les flux d'erreurs renvoyés par une commande. Les autres flux sont dirigés vers stdout.

Ces flux pointent vers des fichiers périphériques, mais comme tout est fichier sous UNIX, les flux d'entrées/sorties peuvent facilement être détournés vers d'autres fichiers. Ce principe fait toute la force du shell.



### ***La redirection d'entrée***

Il est possible de rediriger le flux d'entrée depuis un autre fichier avec le caractère inférieur "<" ou "<<". La commande lira le fichier au lieu du clavier :

```
[root]# ftp -in serverftp << cdes-ftp.txt
```



Seules les commandes demandant une saisie au clavier pourront gérer la redirection d'entrée.

La redirection d'entrée peut également être utilisée pour simuler une interactivité avec l'utilisateur. La commande lira le flux d'entrée jusqu'à rencontrer le mot clef défini après la redirection d'entrée.

Cette fonctionnalité est utilisée pour scripter des commandes interactives :

```
[root]# ftp -in serverftp << FIN
user alice password
put fichier
bye
FIN
```

Le mot clef FIN peut être remplacé par n'importe quel mot.

```
[root]# ftp -in serverftp << STOP
user alice password
put fichier
bye
STOP
```

Le shell quitte la commande `ftp` lorsqu'il reçoit une ligne ne contenant que le mot clef.

La redirection de l'entrée standard est peu utilisée car la plupart des commandes acceptent un nom de fichier en argument.

La commande `wc` pourrait s'utiliser ainsi :

```
[root]# wc -l .bash_profile
27 .bash_profile # le nombre de lignes est suivi du nom du fichier
[root]# wc -l < .bash_profile
27 # le nombre de lignes est seul
```

## **Les redirections de sortie**

Les sorties standards peuvent être redirigées vers d'autres fichiers grâce aux caractères ">" ou "`>>`".

La redirection simple "`>`" écrase le contenu du fichier de sortie :

```
[root]# date +%F > fic_date
```

alors que la redirection double "`>>`" ajoute (concatène) au contenu du fichier de sortie.

```
[root]# date +%F >> fic_date
```

Dans les deux cas, le fichier est automatiquement créé lorsqu'il n'existe pas.

La sortie d'erreur standard peut être également redirigée vers un autre fichier. Cette fois-ci, il faudra préciser le numéro du canal (qui peut être omis pour les canaux 0 et 1) :

```
[root]# ls -R / 2> fic_erreurs  
[root]# ls -R / 2>> fic_erreurs
```

## Exemples

Redirection de 2 sorties vers 2 fichiers :

```
[root]# ls -R / >> fic_ok 2>> fic_nok
```

Redirection des 2 sorties vers un fichier unique :

```
[root]# ls -R / >> fic_log 2>&1
```

Redirection de **stderr** vers un "puit sans fond" (/dev/null) :

```
[root]# ls -R / 2>> /dev/null
```

Redirection vers la ou les consoles actives :

```
[root]# ls -R / 2>> /dev/console
```



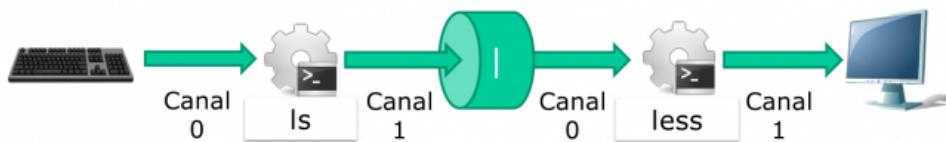
L'exemple précédent utilisant une redirection vers **/dev/console** est utilisé lors du TP sur les tâches planifiées (crontab).

Lorsque les 2 flux de sortie sont redirigés, aucune information n'est affichée à l'écran. Pour utiliser à la fois la redirection de sortie et conserver l'affichage, il faudra utiliser la commande tee.

### 2.7.2. Les tubes (pipe)

Un tube (pipe en anglais) est un mécanisme permettant de relier la sortie standard d'une première commande vers l'entrée standard d'une seconde.

Cette communication est monodirectionnelle et se fait grâce au symbole |. Le symbole pipe "|" est obtenu en appuyant simultanément sur les touches **AltGR+6**.



Toutes les données envoyées par la commande à gauche du tube à travers le canal de sortie standard sont envoyées au canal d'entrée standard de la commande placée à droite.

Les commandes particulièrement utilisées après un pipe sont des filtres.

## **Exemples**

```
# N'afficher que le début :
[root]# ls -lia / | head

# N'afficher que la fin :
[root]# ls -lia / | tail

# Trier le résultat
[root]# ls -lia / | sort

# Compter le nombre de mots / caractères
[root]# ls -lia / | wc

# Chercher une chaîne de caractères dans le résultat :
[root]# ls -lia / | grep fichier
```

## **2.8. Points particuliers**

### **2.8.1. La commande tee**

La commande tee permet de rediriger la sortie standard d'une commande vers un fichier tout en maintenant l'affichage à l'écran.

Elle est combinée avec le pipe “|” pour recevoir en entrée la sortie de la commande à rediriger.

```
[root]# ls -lia / | tee fic
```

L'option -a permet d'ajouter au fichier au lieu de l'écraser.

### 2.8.2. Les commandes alias et unalias

Utiliser les alias est un moyen pour demander au shell de se souvenir d'une commande particulière avec ses options et lui donner un nom.

Par exemple :

```
[root]# ll
```

remplacera la commande :

```
root]# ls -l
```

La commande alias liste les alias de la session en cours. Des alias sont positionnés par défaut sur les distributions linux. Ici, les alias d'un serveur centos :

```
[root]# alias
alias
alias cp='cp -i'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'
```

Les alias ne sont définis que de façon temporaire, le temps de la session utilisateur.

Pour une utilisation permanente, il faut les créer dans le fichier :

- .bashrc du répertoire de connexion de l'utilisateur ;
- /etc/profile.d/alias.sh pour tous les utilisateurs.



Une attention particulière doit être portée lors de l'usage d'alias qui peuvent potentiellement s'avérer dangereux !

Par exemple, un alias mis en place à l'insu de l'administrateur :

```
alias cd='rm -Rf'
```

La commande unalias permet de supprimer les alias.

```
[root]# unalias ll  
  
# Pour supprimer tous les alias :  
root]# unalias -a
```

## ***Alias et fonctions utiles***

### ***grep***

Colorise le résultat de la commande grep :

```
alias grep='grep --color=auto'
```

### ***mcd***

Il est fréquent de créer un dossier puis de se déplacer dedans :

```
mcd() { mkdir -p "$1"; cd "$1"; }
```

### ***cls***

Se déplacer dans un dossier et lister son contenu :

```
cls() { cd "$1"; ls; }
```

### ***backup***

Créer une copie de sauvegarde d'un fichier :

```
backup() { cp "$1"{,.bak}; }
```

## **extract**

Extrait tout type d'archive :

```
extract () {
    if [ -f $1 ] ; then
        case $1 in
            *.tar.bz2) tar xjf $1 ;;
            *.tar.gz) tar xzf $1 ;;
            *.bz2) bunzip2 $1 ;;
            *.rar) unrar e $1 ;;
            *.gz) gunzip $1 ;;
            *.tar) tar xf $1 ;;
            *.tbz2) tar xjf $1 ;;
            *.tgz) tar xzf $1 ;;
            *.zip) unzip $1 ;;
            *.Z) uncompress $1 ;;
            *.7z) 7z x $1 ;;
            *)
                echo "'$1' cannot be extracted via extract()";
        esac
    else
        echo "'$1' is not a valid file"
    fi
}
```

## **cmount**

```
alias cmount="mount | column -t"

[root]# cmount
/dev/simfs on /                                     type simfs
              (rw,relatime,usrquota,grpquota)
proc       on /proc                                 type proc
              (rw,relatime)
sysfs      on /sys                                 type sysfs
              (rw,relatime)
none       on /dev                                 type
              devtmpfs   (rw,relatime,mode=755)
none       on /dev/pts                            type devpts
              (rw,relatime,mode=600,ptmxmode=000)
none       on /dev/shm                            type tmpfs
              (rw,relatime)
```

Le caractère ;

---

none	on	/proc/sys/fs/binfmt_misc	type
binfmt_misc	(rw,relatime)		

### **2.8.3. Le caractère ;**

Le caractère ';' chaîne les commandes.

Les commandes s'exécuteront toutes séquentiellement dans l'ordre de saisie une fois que l'utilisateur aura appuyé sur [ENTREE].

```
[root]# ls /; cd /home; ls -lia; cd /
```

## 2.9. TD commandes de bases

---

### ATELIER AFFICHAGE ET IDENTIFICATION

- Objectifs
  - Prendre en main un poste de travail,
  - Se renseigner sur les utilisateurs connectés.
  
- Se connecter sur la console 1.
- Afficher les informations concernant le login courant.
- Se connecter sur la console 2 avec le même utilisateur.
- Afficher les informations concernant le login courant.
- Afficher les informations concernant l'utilisateur **patrick**.
- D'autres utilisateurs sont-ils connectés sur le serveur ?
- Depuis quelle console êtes-vous connecté ?

### A L'AIDE !

- Rechercher de l'aide sur la commande passwd
  - Rechercher des informations sur le formatage du fichier passwd
  - Quel est l'emplacement de l'UID dans le fichier « passwd » à l'aide du man.
  - Vérifier la date et la mettre à jour si nécessaire.
  - Effacer la console.
- 

### ATELIER ARBORESCENCE ET FICHIERS

- Objectifs
  - créer, supprimer, déplacer des fichiers ou des répertoires ;
  - se déplacer dans l'arborescence.

## CREATION DE REPERTOIRES

- Afficher le répertoire courant.
- Se déplacer de deux façons différentes sous le répertoire « /home ».
  - chemin absolu :
  - chemin relatif :
- Vérifier que /home soit bien le nouveau répertoire courant.
- Retourner dans le répertoire de connexion, et vérifier.
- Créer les répertoires suivants : /home/stagiaire/tickets/ /home/stagiaire/tickets/pierre/ /home/stagiaire/tickets/jacques/

## GESTION DES FICHIERS

- Créer le fichier « /home/stagiaire/tickets/listing\_en\_cours ».
- Copier ce fichier dans les répertoires « /home/stagiaire/tickets/pierre » et « /home/stagiaire/tickets/jacques ». Vérifier la taille de ces fichiers.
  - Vérifier la copie en comparant les tailles :
- Renommer le fichier « /home/stagiaire/tickets/jacques/listing\_en\_cours » en « listing\_fini ».
- Déplacer et renommer le fichier « /home/stagiaire/listing\_en\_cours » en « /STAGE/commandes/archive\_listing ».

## GESTION DES REPERTOIRES

- Copier le répertoire « /home/stagiaire/tickets/pierre/ » et son contenu en le renommant « /home/stagiaire/tickets/sauvegarde ».
- Renommer le répertoire « /home/stagiaire/tickets/sauvegarde/ » en « /home/stagiaire/tickets/archives ».
- Copier le répertoire « /home/stagiaire/tickets/ » dans le répertoire « /STAGE/commandes/ ».

## SUPPRESSION DE FICHIERS ET REPERTOIRES

- Afficher le contenu des répertoires « /home/stagiaire/tickets/jacques/ » et « /home/stagiaire/tickets/pierre/ ».
  - Supprimer le répertoire « /home/stagiaire/tickets/jacques/ » avec la commande « rmdir ».
  - Supprimer le répertoire « /home/stagiaire/pierre/ » en une seule commande.
- 

## ATELIER RECHERCHES ET FILTRES

- Objectifs
    - rechercher un fichier ;
    - rechercher du texte dans un fichier ;
    - afficher un fichier, trier son contenu.
  - Copier dans le répertoire de connexion « /home/stagiaire » le fichier « /etc/passwd ». Dorénavant, travailler sur cette copie.
  - Afficher les 7 premières lignes puis les 3 dernières.
  - Retrouvez la ligne contenant alain.
  - Trier ce fichier par ordre d'UID croissant.
  - Combien y a-t-il d'utilisateurs créés sur le serveur ?
  - Déplacer ce fichier dans le répertoire « /STAGE/commandes ».
  - Afficher les fichiers « passwd » présents dans le dossier /STAGE en précisant leur type.
- 

## ATELIER TUBES ET REDIRECTIONS

- Objectifs
  - utiliser un tube ;
  - utiliser une redirection.

- Créer un fichier « /home/stagiaire/suivi\_admin ».
- Se connecter sur le terminal 2 et suivre les modifications du fichier en direct.  
Se connecter sur le terminal 2 avec **CTRL+SHIFT+ALT+F2** et afficher le fichier en temps réel :



La suite de ce TP se fait sans éditeur de texte !

- Retourner sous le terminal 1 et ajouter au fichier « suivi\_admin » le texte "Voici les répertoires de /STAGE/commandes/gestion/ :" .
- Toujours dans suivi\_admin, ajouter la liste des répertoires de « /STAGE/commandes/gestion/ » en faisant apparaître les tailles avec l'indication Ko, Mo, Go ou To.
- Vérifier le contenu du fichier en basculant sur le terminal 2.

Se connecter sur le terminal 2 avec **CTRL+SHIFT+ALT+F2**

- Retourner sous terminal 1 et ajouter au fichier « suivi\_admin » le texte "Voici les personnes ayant un fichier listing\_en\_cours sous /STAGE/commandes/gestion/ :" .

Retourner sur l'interface graphique avec **ALT+F1**.

- Tapez la commande :

```
[stagiaire]$ find /STAGE/commandes/tickets -listing_en_cours >> /home/stagiaire/suivi_admin 2>/home/stagiaire/erreur
```

- Basculer sur le terminal 2 et vérifier que la commande se soit bien exécutée.

Sur le terminal 2, rien n'a été modifié. En fait, la commande saisie comporte une erreur. Son affichage a donc été redirigé sur le canal d'erreur, le fichier erreur, et non suivi\_admin.

- Corriger la commande pour remplir le fichier « suivi\_admin ».

- Afficher parmi les 3 dernières lignes du fichier suivi\_admin celles qui contiennent "pierre".
- Retourner sous le terminal 2 et se déconnecter.

## 2.10. Correction du TD commandes de bases

---

### ATELIER AFFICHAGE ET IDENTIFICATION

- Objectifs
    - Prendre en main un poste de travail,
    - Se renseigner sur les utilisateurs connectés.
- 

- Se connecter sur la console 1.
- Afficher les informations concernant le login courant.

```
[stagiaire]$ id  
uid=1000(stagiaire) gid=100(users) groupes=100(users) ...
```

- Se connecter sur la console 2 avec le même utilisateur.
- Afficher les informations concernant le login courant.

```
[stagiaire]$ id  
uid=1000(stagiaire) gid=100(users) groupes=100(users) ...
```

- Afficher les informations concernant l'utilisateur **patrick**.

```
[stagiaire]$ id patrick  
uid=503(patrick) gid=501(GroupeP) groupes=501(GroupeP)
```

- D'autres utilisateurs sont-ils connectés sur le serveur ?

```
[stagiaire]$ who  
stagiaire tty1 2016-01-04 13:05  
stagiaire tty2 2016-01-04 13:10
```



Il n'y a que l'utilisateur stagiaire qui est connecté sur le serveur.  
Il est connecté sur le terminal 1 (tty1). Les terminaux physiques sont nommés ttyX, les terminaux virtuels (ceux de l'interface graphique) sont nommés (pts/X).

- Depuis quelle console êtes-vous connecté ?

```
[stagiaire]$ who am i  
stagiaire tty2 2016-01-04 13:10  
[stagiaire]$ whoami  
stagiaire
```

### A L'AIDE !

- Rechercher de l'aide sur la commande passwd

```
[stagiaire]$ whatis passwd  
passwd (1) - Mettre à jour les marques d'authentification d'un  
utilisateur  
passwd (5) - Fichier des mots de passe
```

```
[stagiaire]$ man passwd
```

- Rechercher des informations sur le formatage du fichier passwd :

```
[stagiaire]$ man 5 passwd
```

- Quel est l'emplacement de l'UID dans le fichier « passwd » à l'aide du man.

```
[stagiaire]$ man 5 passwd  
/UID
```

- Vérifier la date et la mettre à jour si nécessaire.

```
[stagiaire]$ date  
jeu. mars 14 15:15:25 CET 2015
```

- Effacer la console.

```
[stagiaire]$ clear
```

---

## ATELIER ARBORESCENCE ET FICHIERS

- Objectifs
    - créer, supprimer, déplacer des fichiers ou des répertoires ;
    - se déplacer dans l'arborescence.
- 

### CREATION DE REPERTOIRES

- Afficher le répertoire courant.

```
[stagiaire]$ pwd  
/home/stagiaire
```

- Se déplacer de deux façons différentes sous le répertoire « /home ».

- chemin absolu :

```
[stagiaire]$ cd /home/
```

- chemin relatif :

```
[stagiaire]$ cd ..
```

- Vérifier que /home soit bien le nouveau répertoire courant.

```
[stagiaire]$ pwd  
/home
```

- Retourner dans le répertoire de connexion, et vérifier.

```
[stagiaire]$ cd  
[stagiaire]$ pwd  
/home/stagiaire
```

- Créer les répertoires suivants : /home/stagiaire/tickets/ /home/stagiaire/tickets/pierre/ /home/stagiaire/tickets/jacques/

```
[stagiaire]$ mkdir -p tickets/pierre tickets/jacques  
[stagiaire]$ ls tickets/  
jacques pierre
```

## GESTION DES FICHIERS

- Créer le fichier « /home/stagiaire/tickets/listing\_en\_cours ».

```
[stagiaire]$ touch tickets/listing_en_cours
```

- Copier ce fichier dans les répertoires « /home/stagiaire/tickets/pierre » et « /home/stagiaire/tickets/jacques ». Vérifier la taille de ces fichiers.

```
[stagiaire]$ cp tickets/listing_en_cours tickets/pierre/  
[stagiaire]$ cp tickets/listing_en_cours tickets/jacques/
```

- Vérifier la copie en comparant les tailles :

```
[stagiaire]$ ls -lh tickets/listing_en_cours tickets/pierre/  
listing_en_cours tickets/jacques/listing_en_cours  
-rw-r--r-- 1 stagiaire users 0 [...] tickets/listing_en_cours  
-rw-r--r-- 1 stagiaire users 0 [...] tickets/pierre/listing_en_cours  
-rw-r--r-- 1 stagiaire users 0 [...] tickets/jacques/listing_en_cours
```



La taille des fichiers est identique, 0 octet (ils sont vides).

- Renommer le fichier « /home/stagiaire/tickets/jacques/listing\_en\_cours » en « listing\_fini ».

```
[stagiaire]$ mv tickets/jacques/listing_en_cours tickets/jacques/
listing_fini
```

- Déplacer et renommer le fichier « /home/stagiaire/listing\_en\_cours » en « /STAGE/commandes/archive\_listing ».



Pour déplacer le fichier listing\_en\_cours du répertoire /home/stagiaire/tickets vers /STAGE/commandes, il faut d'abord créer ce dernier dossier :

```
[stagiaire]$ mkdir -p /STAGE/commandes
```

puis le déplacer :

```
[stagiaire]$ mv tickets/listing_en_cours /STAGE/commandes/
archive_listing
```

## GESTION DES REPERTOIRES

- Copier le répertoire « /home/stagiaire/tickets/pierre/ » et son contenu en le renommant « /home/stagiaire/tickets/sauvegarde ».

```
[stagiaire]$ cp -r tickets/pierre/ tickets/sauvegarde
```

- Renommer le répertoire « /home/stagiaire/tickets/sauvegarde/ » en « /home/stagiaire/tickets/archives ».

```
[stagiaire]$ mv tickets/sauvegarde/ tickets/archives
```

- Copier le répertoire « /home/stagiaire/tickets/ » dans le répertoire « /STAGE/commandes/ ».

```
[stagiaire]$ cp -r tickets/ /STAGE/commandes/
```

## SUPPRESSION DE FICHIERS ET REPERTOIRES

- Afficher le contenu des répertoires « /home/stagiaire/tickets/jacques/ » et « /home/stagiaire/tickets/pierre/ ».

```
[stagiaire]$ ls tickets/jacques/ tickets/pierre/
tickets/jacques/:
listing_fini

tickets/pierre/:
listing_en_cours
```

- Supprimer le répertoire « /home/stagiaire/tickets/jacques/ » avec la commande « rmdir ».

```
[stagiaire]$ rmdir tickets/jacques/
rmdir : échec de suppression de « tickets/jacques/ » : Le dossier n'est
pas vide
[root]# rm -f tickets/jacques/listing_fini
[root]# rmdir tickets/jacques/
```

- Supprimer le répertoire « /home/stagiaire/pierre/ » en une seule commande.

```
[stagiaire]$ rm -rf tickets/pierre/
```



Vérifier les suppressions :

```
[stagiaire]$ ls -R tickets/
tickets/:
archives

tickets/archives:
listing_en_cours
```

---

## ATELIER RECHERCHES ET FILTRES

- Objectifs
  - rechercher un fichier ;

- rechercher du texte dans un fichier ;
  - afficher un fichier, trier son contenu.
- 
- Copier dans le répertoire de connexion « /home/stagiaire » le fichier « /etc/passwd ». Dorénavant, travailler sur cette copie.

```
[stagiaire]$ cp /etc/passwd ./
```

- Afficher les 7 premières lignes puis les 3 dernières.

```
[stagiaire]$ head -n 7 /home/stagiaire/passwd  
[stagiaire]$ tail -n 3 /home/stagiaire/passwd
```

- Retrouvez la ligne contenant alain.

```
[stagiaire]$ grep "alain" /home/stagiaire/passwd  
alain:x:500:500::/home/GroupeA/alain:/bin/bash
```

Ou

```
[stagiaire]$ less /home/stagiaire/passwd
```

Puis

```
/alain
```

- Trier ce fichier par ordre d'UID croissant.

```
[stagiaire]$ sort -k3 -t: -n /home/stagiaire/passwd
```

- Combien y a-t-il d'utilisateurs créés sur le serveur ?

```
[stagiaire]$ wc -l /home/stagiaire/passwd
```

```
39 /home/stagiaire/passwd
```



Le fichier passwd contient 39 lignes, il y a donc 39 utilisateurs créés sur le serveur.

- Déplacer ce fichier dans le répertoire « /STAGE/commandes ».

```
[stagiaire]$ mv /home/stagiaire/passwd /STAGE/commandes
```

- Afficher les fichiers « passwd » présents dans le dossier /STAGE en précisant leur type.

```
[stagiaire]$ find /STAGE -name "passwd" -exec file {} \;  
/STAGE/commandes/passwd: ASCII text
```

---

## ATELIER TUBES ET REDIRECTIONS

- Objectifs
  - utiliser un tube ;
  - utiliser une redirection.
- Créer un fichier « /home/stagiaire/suivi\_admin ».

```
[stagiaire]$ touch /home/stagiaire/suivi_admin
```

- Se connecter sur le terminal 2 et suivre les modifications du fichier en direct.  
Se connecter sur le terminal 2 avec **CTRL+SHIFT+ALT+F2** et afficher le fichier en temps réel :

```
[stagiaire]$ tail -f /home/stagiaire/suivi_admin
```

La suite de ce TP se fait sans éditeur de texte !



Retourner sous le terminal 1 et ajouter au fichier « suivi\_admin » le texte "Voici les répertoires de /STAGE/commandes/gestion/ :" .

- Retourner sur l'interface graphique avec **ALT+F1** et modifier le fichier :

```
[stagiaire]$ echo "Voici les répertoires de /STAGE/commandes/gestion/ :"
> /home/stagiaire/suivi_admin
```

- Toujours dans suivi\_admin, ajouter la liste des répertoires de « /STAGE/ commandes/gestion/ » en faisant apparaître les tailles avec l'indication Ko, Mo, Go ou To.

```
[stagiaire]$ find /STAGE/commandes/gestion/ -type d -exec ls -sdh '{}' \
\; >> /home/stagiaire/suivi_admin
```

- Vérifier le contenu du fichier en basculant sur le terminal 2.

Se connecter sur le terminal 2 avec **CTRL+SHIFT+ALT+F2**

- Retourner sous terminal 1 et ajouter au fichier « suivi\_admin » le texte "Voici les personnes ayant un fichier listing\_en\_cours sous /STAGE/commandes/gestion/ :" .

Retourner sur l'interface graphique avec **ALT+F1**.

```
[stagiaire]$ echo "Voici les personnes ayant un fichier listing_en_cours
sous /STAGE/commandes/gestion/ :" >> /home/stagiaire/suivi_admin
```

- Tapez la commande :

```
[stagiaire]$ find /STAGE/commandes/tickets -listing_en_cours >> /home/
stagiaire/suivi_admin 2>/home/stagiaire/erreur
```

- Basculer sur le terminal 2 et vérifier que la commande se soit bien exécutée.

Sur le terminal 2, rien n'a été modifié. En fait, la commande saisie comporte une erreur. Son affichage a donc été redirigé sur le canal d'erreur, le fichier erreur, et non suivi\_admin.

- Corriger la commande pour remplir le fichier « suivi\_admin ».

Il faut donc corriger la commande :

```
[stagiaire]$ find /STAGE/commandes/tickets -name listing_en_cours >> /home/stagiaire/suivi_admin 2> /home/stagiaire/erreur
```

- Afficher parmi les 3 dernières lignes du fichier suivi\_admin celles qui contiennent "pierre".

```
[stagiaire]$ tail -n3 /home/stagiaire/suivi_admin | grep "pierre" /STAGE/commandes/tickets/pierre/listing_en_cours
```

- Retourner sous le terminal 2 et se déconnecter.

taper **Ctrl+d** puis:

```
[stagiaire]$ exit
```



# 3

## La gestion des utilisateurs

### 3.1. Généralités

Chaque utilisateur est membre d'au moins un groupe : **c'est son groupe principal.**

Plusieurs utilisateurs peuvent faire partie d'un même groupe.

Les utilisateurs peuvent appartenir à d'autres groupes. Ces utilisateurs sont invités dans ces **groupes secondaires**.



Chaque utilisateur possède un groupe principal et peut être invité dans un ou plusieurs groupes secondaires.

Les groupes et utilisateurs se gèrent par leur identifiant numérique unique GID et UID.

Les fichiers de configuration se trouvent dans "/etc".

- **UID** : User IDentifier. Identifiant unique d'utilisateur.
- **GID** : Group IDentifier. Identifiant unique de groupe.



Il est recommandé d'utiliser les commandes d'administration au lieu de modifier manuellement les fichiers.

### 3.2. Gestion des groupes

Fichiers modifiés, ajout de lignes :

- /etc/group
- /etc/gshadow

### 3.2.1. Commande groupadd

La commande groupadd permet d'ajouter un groupe au système.

#### Syntaxe de la commande groupadd.

```
groupadd [-f] [-g GID] groupe
```

Exemple :

```
[root]# groupadd -g 512 GroupeB
```

Tableau 3.1. Options de la commande groupadd

Option	Description
-g GID	GID du groupe à créer.
-f	Le système choisit un GID si celui précisé par l'option -g existe déjà.
-r	Crée un groupe système avec un GID compris entre <b>SYS_GID_MIN</b> et <b>SYS_GID_MAX</b> définies dans <b>/etc/login.defs</b> .

Règles de nommage des groupes :

- Pas d'accents, ni caractères spéciaux ;
- Différents du nom d'un utilisateur ou fichier système existant.

### 3.2.2. Commande groupmod

La commande groupmod permet de modifier un groupe existant sur le système.

#### Syntaxe de la commande groupmod.

```
groupmod [-g GID] [-n nom] groupe
```

Exemple :

```
[root]# groupmod -g 516 GroupeP  
[root]# groupmod -n GroupeC GroupeB
```

Tableau 3.2. Options de la commande groupmod

Option	Description
-g GID	Nouveau GID du groupe à modifier.
-n nom	Nouveau nom.

Il est possible de modifier le nom d'un groupe, son GID ou les deux simultanément.

Après modification, les fichiers appartenant au groupe ont un GID inconnu. Il faut leur réattribuer le nouveau GID.

```
[root]# find / -gid 502 -exec chgrp 516 {} \;
```

### 3.2.3. Commande groupdel

La commande groupdel permet de supprimer un groupe existant sur le système.

#### Syntaxe de la commande groupdel.

```
groupdel groupe
```

Exemple :

```
[root]# groupdel GroupeC
```



Pour être supprimé, un groupe ne doit plus contenir d'utilisateurs.

La suppression du dernier utilisateur d'un groupe éponyme entraînera la suppression de ce groupe par le système.



Chaque groupe possède un GID unique. Un groupe peut être dupliqué. Par convention, les GID des groupes systèmes vont de 0 (root) à 499.



Un utilisateur faisant obligatoirement partie d'un groupe, il est nécessaire de créer les groupes avant d'ajouter les utilisateurs. Par conséquent, un groupe peut ne pas avoir de membres.

### **3.2.4. Fichier /etc/group**

Ce fichier contient les informations de groupes (séparées par ' : ').

```
[root]# tail -1 /etc/group
GroupeP:x:516:stagiaire
    1   2   3           4
```

- 1 : Nom du groupe.
- 2 : Mot de passe (x si défini dans /etc/gshadow).
- 3 : GID.
- 4 : Membres invités (séparés par des virgules, ne contient pas les membres principaux).



Chaque ligne du fichier /etc/group correspond à un groupe. Les utilisateurs dont ce groupe est leur groupe principal ne sont pas listés à ce niveau.

Cette information d'appartenance est en fait déjà fournie par le fichier /etc/passwd...

### **Fichier /etc/gshadow**

Ce fichier contient les informations de sécurité sur les groupes (séparées par ' : ').

```
[root]# grep GroupeA /etc/gshadow
GroupeA:$6$2,9,v...SBn160:alain:stagiaire
    1           2           3           4
```

- 1 : Nom du groupe.
- 2 : Mot de passe chiffré.
- 3 : Administrateur du groupe.

- 4 : Membres invités (séparés par des virgules, ne contient pas les membres principaux).



Pour chaque ligne du fichier /etc/group doit correspondre une ligne du fichier /etc/gshadow.

Un ! au niveau du mot de passe indique que celui-ci est bloqué. Ainsi aucun utilisateur ne peut utiliser le mot de passe pour accéder au groupe (sachant que les membres du groupe n'en ont pas besoin).

### 3.3. Gestion des utilisateurs

#### 3.3.1. Définition

Un utilisateur se définit comme suit dans le fichier **/etc/passwd** :

1. Login ;
2. Mot de passe ;
3. UID ;
4. GID du groupe principal ;
5. Commentaire ;
6. Répertoire de connexion ;
7. Interpréteur de commandes (/bin/bash, /bin/nologin,...).

Il existe trois types d'utilisateurs :

- **root** : Administrateur du système ;
- **utilisateur système** : Utilisé par le système pour la gestion des droits d'accès des applications ;
- **utilisateur ordinaire** : Autre compte permettant de se connecter au système.

Fichiers modifiés, ajout de lignes :

- /etc/passwd
- /etc/shadow

### 3.3.2. Commande useradd

La commande useradd permet d'ajouter un utilisateur.

#### Syntaxe de la commande useradd.

```
useradd [-u UID] [-g GID] [-d répertoire] [-s shell] login
```

Exemple :

```
[root]# useradd -u 1000 -g 513 -d /home/GroupeC/carine carine
```

Tableau 3.3. Options de la commande useradd

Option	Description
-u UID	UID de l'utilisateur à créer.
-g GID	GID du groupe principal.
-d répertoire	Répertoire de connexion.
-s shell	Interpréteur de commandes.
-c	Ajoute un commentaire.
-U	Ajoute l'utilisateur à un groupe portant le même nom créé simultanément.
-M	Ne crée pas le répertoire de connexion.

À la création, le compte ne possède pas de mot de passe et est verrouillé. Il faut assigner un mot de passe pour déverrouiller le compte.

Règles de nommage des comptes :

- Pas d'accents, de majuscules ni caractères spéciaux ;
- Différents du nom d'un groupe ou fichier système existant ;
- Définir les options -u, -g, -d et -s à la création.



L'arborescence du répertoire de connexion doit être créée à l'exception du dernier répertoire. Le dernier répertoire est

créé par la commande useradd qui en profite pour y copier les fichiers du "skel".

**Un utilisateur peut faire partie de plusieurs groupes en plus de son groupe principal.**

Pour les groupes secondaires, il faut utiliser l'option -G.

Exemple :

```
[root]# useradd -u 500 -g GroupeA -G GroupeP,GroupeC albert
```

### **Valeur par défaut de création d'utilisateur.**

Modification du fichier **/etc/default/useradd**.

```
useradd -D [-b répertoire] [-g groupe] [-s shell]
```

Exemple :

```
[root]# useradd -D -g 500 -b /home -s /bin/bash
```

Tableau 3.4. Options de la commande useradd pour modifier les valeurs par défaut

Option	Description
-D	Définit les valeurs par défaut de création d'utilisateur.
-b répertoire	Définit le répertoire de connexion par défaut.
-g groupe	Définit le groupe par défaut.
-s shell	Définit le shell par défaut.
-f	Nombre de jours suivant l'expiration du mot de passe avant que le compte ne soit désactivé.
-e	Date à laquelle le compte sera désactivé.

### **3.3.3. Commande usermod**

La commande usermod permet de modifier un utilisateur.

#### **Syntaxe de la commande usermod.**

```
usermod [-u UID] [-g GID] [-d répertoire] [-m] login
```

Exemple :

```
[root]# usermod -u 544 carine
```

Options identiques à la commande useradd.

**Tableau 3.5. Options de la commande usermod**

Option	Description
-m	Associé à l'option -d, déplace le contenu de l'ancien répertoire de connexion vers le nouveau.
-l login	Nouveau nom.
-e AAAA-MM-JJ	Date d'expiration du compte.
-L	Verrouille le compte.
-U	Déverrouille le compte.
-a	Empêche la suppression de l'utilisateur d'un groupe secondaire lors de l'ajout dans un autre groupe secondaire.
-G	Précise plusieurs groupes secondaires lors de l'ajout.

Avec la commande **usermod**, le verrouillage d'un compte se traduit par l'ajout de ! devant le mot de passe dans le fichier **/etc/shadow**.



Pour être modifié un utilisateur doit être déconnecté et ne pas avoir de processus en cours.

Après modification de l'identifiant, les fichiers appartenant à l'utilisateur ont un UID inconnu. Il faut leur réattribuer le nouvel UID.

```
[root]# find / -uid 1000 -exec chown 544: {} \;
```

Il est possible d'inviter un utilisateur dans un ou plusieurs groupes secondaires avec les options -a et -G.

Exemple :

```
[root]# usermod -aG GroupeP,GroupeC albert
```

La commande **usermod** agit en modification et non en ajout.

Pour un utilisateur invité dans un groupe par l'intermédiaire de cette commande et déjà positionné comme invité dans d'autres groupes secondaires, il faudra indiquer dans la commande de gestion de groupe tous les groupes dont il fait partie sinon il disparaîtra de ceux-ci.

L'option **-a** empêche ce problème.

Exemples :

- Invite albert dans le groupe GroupeP

```
[root]# usermod -G GroupeP albert
```

- Invite albert dans le groupe GroupeG, mais le supprime de la liste des invités de GroupeP.

```
[root]# usermod -G GroupeG albert
```

- Donc soit :

```
[root]# usermod -G GroupeP,GroupeG albert
```

- Soit :

```
[root]# usermod -aG GroupeG albert
```

### **3.3.4. Commande userdel**

La commande **userdel** permet de supprimer le compte d'un utilisateur.

**Syntaxe de la commande userdel.**

```
[root]# userdel -r carine
```

Tableau 3.6. Options de la commande userdel

Option	Description
-r	Supprime le répertoire de connexion et les fichiers contenus.



Pour être supprimé, un utilisateur doit être déconnecté et ne pas avoir de processus en cours.

userdel supprime la ligne de l'utilisateur dans les fichiers /etc/passwd et /etc/gshadow



Chaque utilisateur possède un UID unique. Par convention, les UID des utilisateurs 'système' vont de 0 (root) à 499.



Un utilisateur est obligatoirement membre d'un groupe. Il est donc nécessaire de créer les groupes avant d'ajouter les utilisateurs.

### 3.3.5. Fichier /etc/passwd

Ce fichier contient les informations des utilisateurs (séparées par ':' ).

```
[root]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
1 2 3 4 5       6      7
```

- 1 : Login.
- 2 : Mot de passe (x si défini dans /etc/shadow).
- 3 : UID.
- 4 : GID du groupe principal.
- 5 : Commentaire.
- 6 : Répertoire de connexion.
- 7 : Interpréteur de commandes.

### 3.3.6. Fichier /etc/shadow

Ce fichier contient les informations de sécurité des utilisateurs (séparées par ':' ).

```
[root]# tail -1 /etc/shadow
root:$6$...:15399:0:99999:7:::
    1   2     3   4   5   6,7,8,9
```

- 1 : Login.
- 2 : Mot de passe chiffré.
- 3 : Date du dernier changement.
- 4 : Durée de vie minimale du mot de passe.
- 5 : Durée de vie maximale du mot de passe.
- 6 : Nombre de jours avant avertissement.
- 7 : Délai avant désactivation du compte après expiration.
- 8 : Délai d'expiration du compte.
- 9 : Réservé pour une utilisation future.



Pour chaque ligne du fichier /etc/passwd doit correspondre une ligne du fichier /etc/shadow.

## 3.4. Propriétaires des fichiers



Tous les fichiers appartiennent forcément à un utilisateur et à un groupe.

Le groupe principal de l'utilisateur qui crée le fichier est, par défaut, le groupe propriétaire du fichier.

### 3.4.1. Commandes de modifications :

#### **Commande chown**

La commande chown permet de modifier les propriétaires d'un fichier.

**Syntaxe de la commande chown.**

```
chown [-R] [-v] login[:groupe] fichier
```

Exemples :

```
[root]# chown root fichier  
[root]# chown albert:GroupeA fichier
```

**Tableau 3.7. Options de la commande chown**

Option	Description
-R	Modifie les propriétaires du répertoire et de son contenu.
-v	Affiche les modifications exécutées.

Pour ne modifier que l'utilisateur propriétaire :

```
[root]# chown albert fichier
```

Pour ne modifier que le groupe propriétaire :

```
[root]# chown :GroupeA fichier
```

Modification de l'utilisateur et du groupe propriétaire :

```
[root]# chown albert:GroupeA fichier
```

Dans l'exemple suivant le groupe attribué sera le groupe principal de l'utilisateur précisé.

```
[root]# chown albert: fichier
```

### **3.4.2. Commande chgrp**

La commande chgrp permet de modifier le groupe propriétaire d'un fichier.

#### **Syntaxe de la commande chgrp.**

```
chgrp [-R] [-v] groupe fichier
```

Exemple :

```
[root]# chgrp groupe1 fichier
```

**Tableau 3.8. Options de la commande chgrp**

Option	Description
-R	Modifie les groupes propriétaires du répertoire et de son contenu (récursivité).
-v	Affiche les modifications exécutées.



Il est possible d'appliquer à un fichier un propriétaire et un groupe propriétaire en prenant comme référence ceux d'un autre fichier :

```
chown [options] --reference=RRFILE FILE
```

Par exemple :

```
chown --reference=/etc/groups /etc/passwd
```

## 3.5. Gestion des invités

### 3.5.1. Commande gpasswd

La commande gpasswd permet de gérer un groupe.

#### Syntaxe de la commande gpasswd.

```
gpasswd [-a login] [-A login] [-d login] [-M login] groupe
```

Exemples :

```
[root]# gpasswd -A alain GroupeA
[alain]$ gpasswd -a patrick GroupeA
```

**Tableau 3.9. Options de la commande gpasswd**

<b>Option</b>	<b>Description</b>
-a login	Ajoute l'utilisateur au groupe.
-A login	Définit l'administrateur du groupe.
-d login	Retire l'utilisateur du groupe.
-M login	Définit la liste exhaustive des invités.

La commande gpasswd -M agit en modification et non en ajout.

```
# gpasswd GroupeA
New Password :
Re-enter new password :
```

### **3.5.2. Commande id**

La commande id affiche les noms des groupes d'un utilisateur.

#### **Syntaxe de la commande id.**

```
id login
```

Exemple :

```
[root]# id alain
uid=500(alain) gid=500(GroupeA) groupes=500(GroupeA),516(GroupeP)
```

### **3.5.3. Commande newgrp**

La commande newgrp permet d'utiliser temporairement un groupe secondaire pour la création de fichiers.

#### **Syntaxe de la commande newgrp.**

```
newgrp [groupesecondaire]
```

Exemple :

```
[alain]$ newgrp GroupeB
```



Après utilisation de cette commande, les fichiers seront créés avec le GID de son groupe secondaire.

La commande newgrp sans paramètre réaffecte le groupe principal.

## 3.6. Sécurisation

### 3.6.1. Commande passwd

La commande passwd permet de gérer un mot de passe.

**Syntaxe de la commande passwd.**

```
passwd [-d] [-l] [-S] [-u] [login]
```

Exemples :

```
[root]# passwd -l albert
[root]# passwd -n 60 -x 90 -w 80 -i 10 patrick
```

Tableau 3.10. Options de la commande passwd

Option	Description
-d	Supprime le mot de passe.
-l	Verrouille le compte.
-S	Affiche le statut du compte.
-u	Déverrouille le compte.
-e	Fait expirer le mot de passe.
-n jours	Durée de vie minimale du mot de passe.
-x jours	Durée de vie maximale du mot de passe.
-w jours	Délai d'avertissement avant expiration.
-i jours	Délai avant désactivation lorsque le mot de passe expire.

Avec la commande passwd, le verrouillage d'un compte se traduit par l'ajout de !! devant le mot de passe dans le fichier /etc/shadow.

L'utilisation de la commande usermod -U ne supprime qu'un seul des !. Le compte reste donc verrouillé.



Cette commande est accessible aux utilisateurs pour modifier leur mot de passe (l'ancien mot de passe est demandé).

L'administrateur peut modifier les mots de passe de tous les utilisateurs sans restriction.

Exemple :

- Alain change son mot de passe :

```
[alain]$ passwd
```

- root change le mot de passe d'alain :

```
[root]# passwd alain
```



La commande passwd est accessible aux utilisateurs pour modifier leur mot de passe (l'ancien mot de passe est demandé). L'administrateur peut modifier les mots de passe de tous les utilisateurs sans restriction.

Ils devront se soumettre aux restrictions de sécurité.

Lors d'une gestion des comptes utilisateurs par script shell, il peut être utile de définir un mot de passe par défaut après avoir créé l'utilisateur.

Ceci peut se faire en passant le mot de passe à la commande passwd.

Exemple :

```
[root]# echo "azerty,1" | passwd --stdin philippe
```



Le mot de passe est saisi en clair, passwd se charge de le chiffrer.

### 3.7. Commande chage

La commande chage permet de gérer la stratégie de compte.

#### Syntaxe de la commande chage.

```
chage [-d date] [-E date] [-I jours] [-l] [-m jours] [-M jours] [-W
jours] [login]
```

Exemple :

```
[root]# chage -m 60 -M 90 -W 80 -I 10 alain
```

**Tableau 3.11. Options de la commande chage**

Option	Description
-l jours	Délai avant désactivation, mot de passe expiré (l majuscule).
-I	Affiche le détail de la stratégie (l minuscule).
-m jours	Durée de vie minimale du mot de passe.
-M jours	Durée de vie maximale du mot de passe.
-d AAA-MM-JJ	Dernière modification du mot de passe.
-E AAA-MM-JJ	Date d'expiration du compte.
-W jours	Délai d'avertissement avant expiration.

La commande chage propose également un mode interactif.

L'option -d force la modification du mot de passe à la connexion.

Exemples :

```
[root]# chage philippe
[root]# chage -d 0 philippe
```



En l'absence d'utilisateur précisé, la commande concerne l'utilisateur qui la saisit.

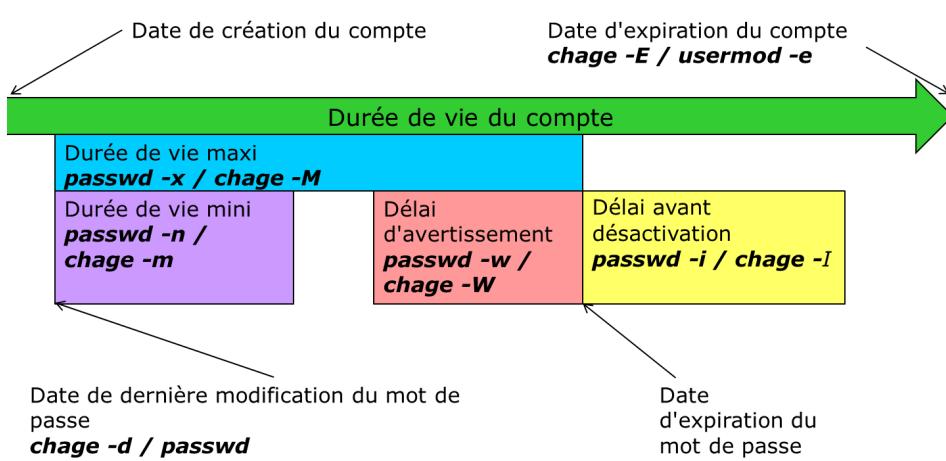


Figure 3.1. Gestion des comptes utilisateurs avec chage

## 3.8. Gestion avancée

Fichiers de configuration :

- /etc/default/useradd
- /etc/login.defs
- /etc/skel



L'édition du fichier /etc/default/useradd se fait grâce à la commande useradd.

Les autres fichiers sont à modifier avec un éditeur de texte.

### 3.8.1. Fichier /etc/default/useradd

Ce fichier contient le paramétrage des données par défaut.



Lors de la création d'un utilisateur, si les options ne sont pas précisées, le système utilise les valeurs par défaut définies dans /etc/default/useradd.

Ce fichier est modifié par la commande **useradd -D** (**useradd -D** saisie sans autre option affiche le contenu du fichier **/etc/default/useradd**).

**Tableau 3.12. Contenu du fichier /etc/default/useradd**

Valeur	Commentaire
GROUP	Groupe par défaut.
HOME	Chemin dans lequel le répertoire de connexion du nom de l'utilisateur sera créé.
INACTIVE	Nombre de jours suivant l'expiration du mot de passe avant que le compte ne soit désactivé.
EXPIRE	Date d'expiration du compte.
SHELL	Interpréteur de commandes.
SKEL	Répertoire squelette du répertoire de connexion.
CREATE_MAIL_SPOOLDIR	Création de la boîte aux lettres dans /var/spool/mail.



Sans l'option **-g**, la commande **useradd** crée un groupe du nom de l'utilisateur et l'y place.

Pour que la commande **useradd** récupère la valeur du champ **GROUP** du fichier **/etc/default/useradd**, il faut préciser l'option **-N**.

Exemple :

```
[root]# useradd -u 501 -N GroupeA
```

### 3.8.2. Fichier /etc/login.defs

Ce fichier contient de nombreux paramètres par défaut utiles aux commandes de création ou de modification d'utilisateurs. Ces informations sont regroupées par paragraphe en fonction de leur utilisation :

- Boites aux lettres ;
- Mots de passe ;
- UID et GID ;
- Umask ;
- Connexions ;

- Terminaux.

### **3.8.3. Fichier /etc/skel**

Lors de la création d'un utilisateur, son répertoire personnel et ses fichiers d'environnement sont créés.

Ces fichiers sont copiés automatiquement à partir du répertoire /etc/skel.

- .bash\_logout
- .bash\_profile
- .bashrc

Tous les fichiers et répertoires placés dans ce répertoire seront copiés dans l'arborescence des utilisateurs lors de leur création.

## **3.9. Changement d'identité**

### **3.9.1. Commande su**

La commande su permet de modifier l'identité de l'utilisateur connecté.

**Syntaxe de la commande su.**

```
su [-] [-c commande] [login]
```

Exemples :

```
[root]# su - alain  
[albert]$ su -c "passwd alain"
```

Tableau 3.13. Options de la commande su

Option	Description
-	Charge l'environnement complet de l'utilisateur.
-c commande	Exécute la commande sous l'identité de l'utilisateur.

Si le login n'est pas spécifié, ce sera root.

Les utilisateurs standards devront taper le mot de passe de la nouvelle identité.



Il y a création de couches successives. Pour passer d'un utilisateur à un autre, il faut d'abord taper la commande exit pour reprendre son identité puis la commande su pour prendre une autre identité.

## ***Chargement du profil***

root endosse alain avec su :

```
...  
/home/GroupeA/alain/bash_rc  
/etc/bashrc  
...
```

root endosse alain avec su - :

```
...  
/home/GroupeA/alain/bash_profile  
/home/GroupeA/alain/bash_rc  
/etc/bashrc  
...
```

Un utilisateur peut endosser temporairement (pour une autre commande ou une session entière) l'identité d'un autre compte.

Si aucun utilisateur n'est précisé, la commande concerne root su -.

Il est nécessaire de connaître le mot de passe de l'utilisateur dont l'identité est endossé sauf si c'est root qui exécute la commande.

Un administrateur peut ainsi travailler sur un compte utilisateur standard et n'utiliser les droits du compte root que ponctuellement.



---

# 4

## Système de fichiers

---

### 4.1. Partitionnement

Le partitionnement va permettre l'installation de plusieurs systèmes d'exploitation car il est impossible d'en faire cohabiter plusieurs sur un même lecteur logique. Le partitionnement permet également de cloisonner des données (sécurité, optimisation d'accès, ...).

Le découpage du disque physique en volumes partitionnés est inscrit dans la table des partitions stockée dans le premier secteur du disque (MBR : Master Boot Record).

Un même disque physique peut être découpé en 4 partitions maximum :

- **Primaire** (ou principale)
- **Étendue**



Il ne peut y avoir qu'une seule partition étendue par disque physique. Afin de bénéficier de lecteur supplémentaire, la partition étendue peut être découpée en partitions logiques

## Partitions principales seulement

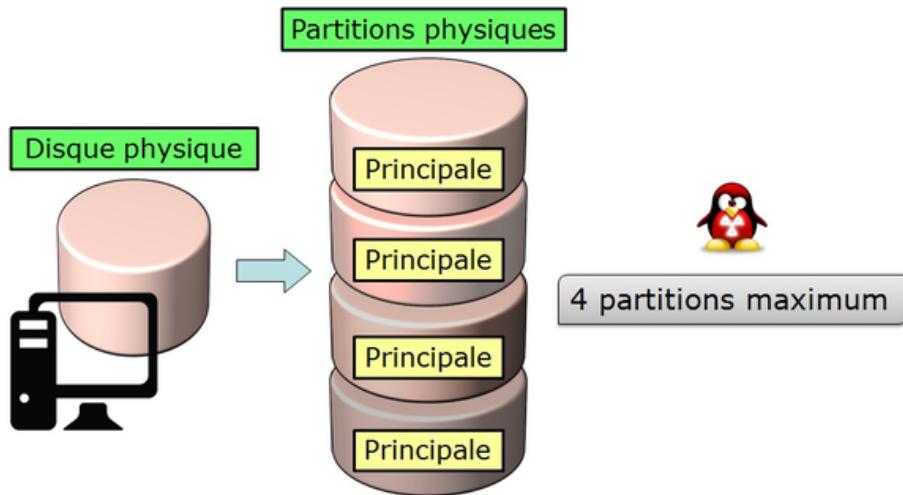


Figure 4.1. Découpage en quatre partitions principales seulement

## Partitions principales et étendue

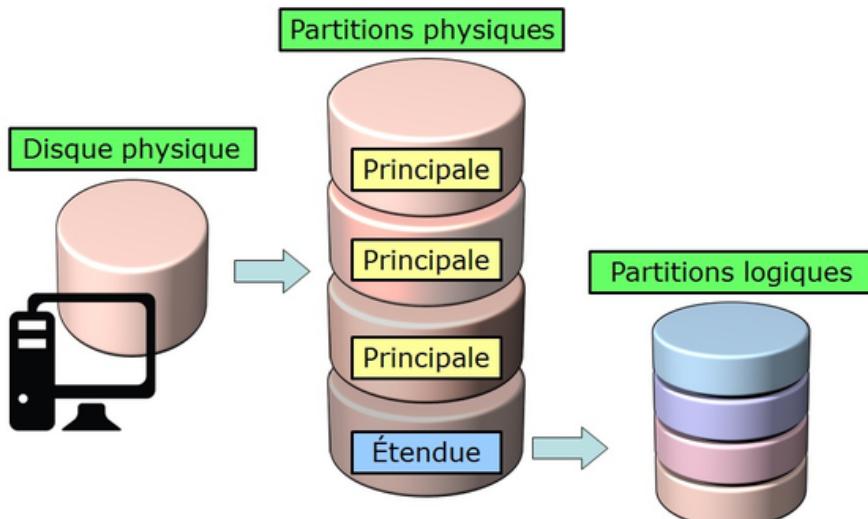


Figure 4.2. Découpage en trois partitions principales et une partition étendue

Les devices, ou périphériques, sont les fichiers identifiant les différents matériels détectés par la carte mère. Ces fichiers sont stockés sans `/dev`. Le service qui détecte les nouveaux périphériques et leur donne des noms s'appelle "udev".

Ils sont identifiés en fonction de leur type.

Les périphériques de stockage se nomment **hd** pour les disques durs IDE et **sd** pour les autres supports. Vient ensuite une lettre qui commence par **a** pour le premier périphérique, puis **b, c, ...**

Enfin nous allons trouver un chiffre qui définit le volume partitionné : **1** pour la première partition primaire, ...



Attention, la partition étendue, qui ne supporte pas de système de fichier, porte quand même un numéro.

### Exemple d'identification de disque IDE

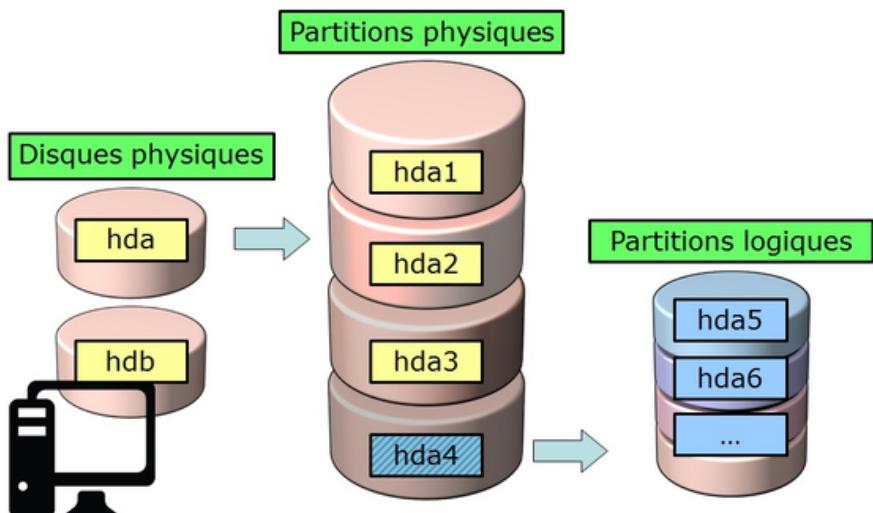


Figure 4.3. Identification des partitions

Il existe deux commandes permettant le partitionnement d'un disque : **fdisk** et **cfdisk**. Ces deux commandes possèdent un menu interactif. **cfdisk** étant plus fiable et mieux optimisée, il est préférable de l'utiliser.

La seule raison d'utiliser **fdisk** est lorsqu'on souhaite lister tous les périphériques logiques avec l'option **-l**.

```
[root]# fdisk -l  
[root]# fdisk -l /dev/sdc  
[root]# fdisk -l /dev/sdc2
```

#### 4.1.1. Commande cfdisk

La commande cfdisk permet de gérer les partitions

##### Syntaxe de la commande cfdisk.

```
cfdisk device
```

Exemple :

```
[root]# cfdisk /dev/sda  
      cfdisk (util-linux-ng 2.17.2)  
      Unité disque : /dev/sda  
      Taille: 10737418240 octets, 10.7 Go  
      Têtes: 255 Secteurs par piste: 63 Cylindres: 1305  
      Nom   Fanions  Part Type Sys.Fic  Étiq. Taille  
      -----  
      ...  
[aide] [nouvelle] [afficher] [quitter] [unités] [écrire]
```

La préparation, sans LVM, du support physique passe par cinq étapes :

- Mise en place du disque physique ;
- Partitionnement des volumes (découpage géographique du disque, possibilité d'installer plusieurs systèmes, ...) ;
- Création des systèmes de fichiers (permet au système d'exploitation de gérer les fichiers, l'arborescence, les droits, ...) ;
- Montage des systèmes de fichiers (inscription du système de fichiers dans l'arborescence) ;
- Gérer l'accès aux utilisateurs.

## 4.2. LVM

### Logical Volume Manager

La gestion de volume crée une couche abstraite sur un stockage physique offrant des avantages par rapport à l'utilisation directe du stockage physique :

- Capacité du disque plus flexible ;
- Déplacement des données en ligne ;
- Disques en mode “stripe” (découpage) ;
- Volumes miroirs (recopie) ;
- Instantanés de volumes (snapshot).

L'inconvénient est que si un des volumes physiques devient HS, alors c'est l'ensemble des volumes logiques qui utilisent ce volume physique qui sont perdus. Il faudra utiliser LVM sur des disques raid.

Le LVM est disponible sous linux à partir de la version 2.4 du noyau.



LVM est uniquement géré par le système d'exploitation. Par conséquent le BIOS a besoin d'au moins une partition sans LVM pour démarrer.

### 4.2.1. Les groupes de volume

Les volumes physiques **PV** (issus des partitions) sont combinés en des groupes de volumes **VG**. Chaque **VG** représente un espace disque pouvant être découpé en volumes logiques **LV**. **L'extension** est la plus petite unité d'espace de taille fixe pouvant être allouée.

- **PE** : Physical Extension
- **LE** : Logical Extension

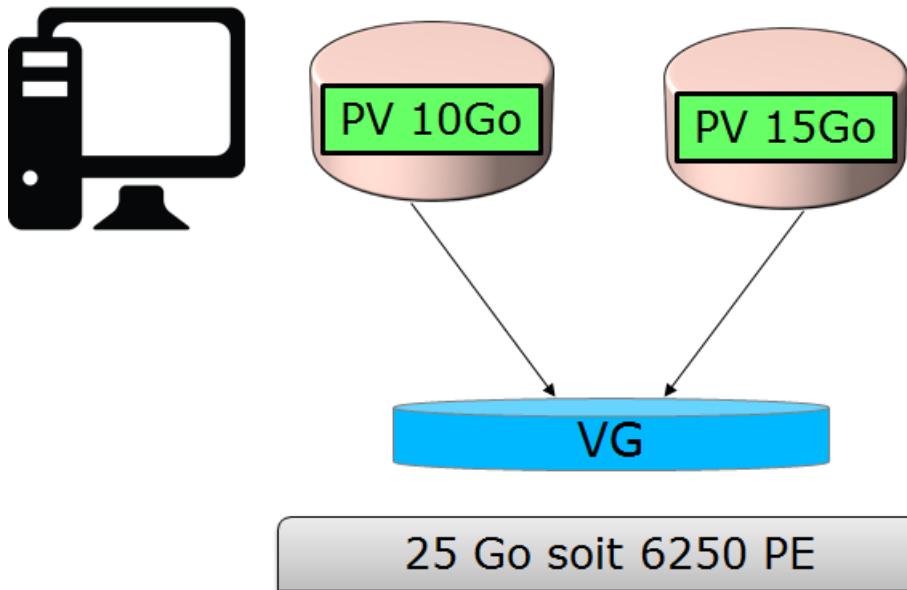


Figure 4.4. Groupe de volumes, taille de PE égale à 4Mo

#### 4.2.2. *Les volumes logiques*

Un groupe de volume **VG** est divisé en volumes logiques **LV** offrant différents modes de fonctionnement :

- Volumes linéaires ;
- Volumes en mode stripe ;
- Volumes en miroirs.

## Les volumes linéaires

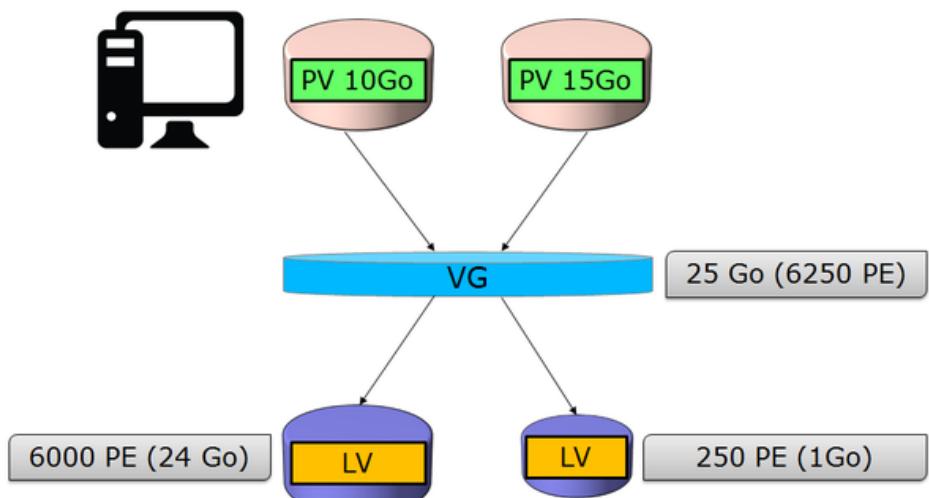


Figure 4.5. Volumes linéaires

## Les volumes en mode « stripe »

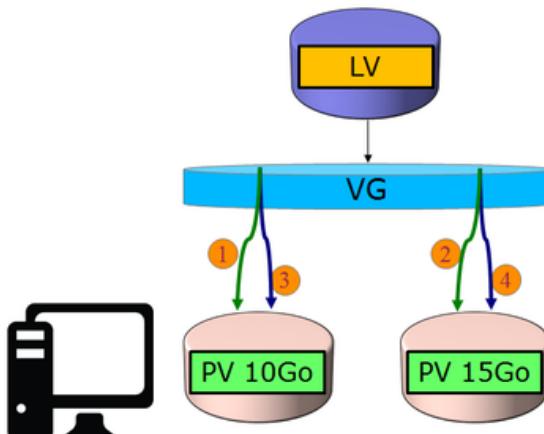


Figure 4.6. Volumes en mode stripe



Le “striping” améliore les performances en écrivant des données sur un nombre prédéterminé de volumes physiques avec une technique de round-robin.

## Les volumes miroirs

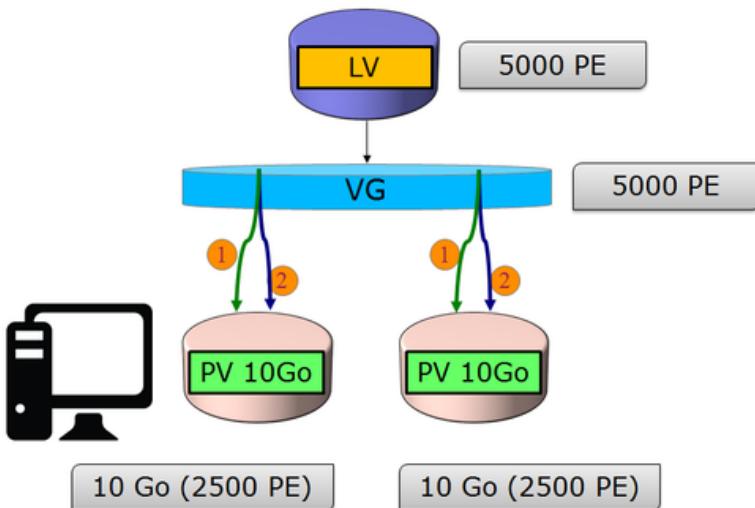


Figure 4.7. Volumes en miroirs

### 4.2.3. Commandes LVM pour la gestion des volumes

#### Commande **pvcreate**

La commande **pvcreate** permet de créer des volumes physiques. Elle transforme des partition Linux en volumes physiques.

#### Syntaxe de la commande **pvcreate**.

```
pvcreate [-options] partition
```

Exemple :

```
[root]# pvcreate /dev/hdb1
pvcreate -- physical volume « /dev/hdb1 » successfully created
```

Tableau 4.1. Option de la commande **pvcreate**

Option	Description
-f	Force la création du volume (disque déjà transformé en volume physique).

## Commande **vgcreate**

La commande **vgcreate** permet de créer des groupes de volumes. Elle regroupe un ou plusieurs volumes physiques dans un groupe de volumes.

### Syntaxe de la commande **vgcreate**.

```
vgcreate volume physical_volume [PV...]
```

Exemple :

```
[root]# vgcreate volume1 /dev/hdb1
...
vgcreate - volume group « volume1 » successfully created and activated
```

## Commande **lvcreate**

La commande **lvcreate** permet de créer des volumes logiques. Le système de fichiers est ensuite créé sur ces volumes logiques.

### Syntaxe de la commande **lvcreate**.

```
lvcreate -L taille [-n nom] nom_VG
```

Exemple :

```
[root]# lvcreate -L 600M -n VolLog1 volume1
lvcreate -- logical volume « /dev/volume1/VolLog1 » successfully created
```

**Tableau 4.2. Options de la commande **lvcreate****

Option	Description
-L taille	Taille du volume logique en K, M ou G
-n nom	Nom du LV. Fichier spécial créé dans <b>/dev/nom_volume</b> portant ce nom

#### **4.2.4. Commandes LVM pour visualiser les informations concernant les volumes**

##### **Commande *pvdisplay***

La commande **pvdisplay** permet de visualiser les informations concernant les volumes physiques.

##### **Syntaxe de la commande *pvdisplay*.**

```
pvdisplay /dev/nom_PV
```

Exemple :

```
[root]# pvdisplay /dev/nom_PV
```

##### **Commande *vgdisplay***

La commande **vgdisplay** permet de visualiser les informations concernant les groupes de volumes.

##### **Syntaxe de la commande *vgdisplay*.**

```
vgdisplay nom_VG
```

Exemple :

```
[root]# vgdisplay volume1
```

##### **Commande *lvdisplay***

La commande **lvdisplay** permet de visualiser les informations concernant les volumes logiques.

##### **Syntaxe de la commande *lvdisplay*.**

```
lvdisplay /dev/nom_VG/nom_LV
```

Exemple :

```
[root]# lvdisplay /dev/volume1/VolLog1
```

#### **4.2.5. Préparation du support physique**

La préparation avec LVM du support physique se décompose comme suit :

- Mise en place du disque physique
- Partitionnement des volumes
- **Volume physique LVM**
- **Groupes de volumes LVM**
- **Volumes logiques LVM**
- Création des systèmes de fichiers
- Montage des systèmes de fichiers
- Gérer l'accès aux utilisateurs

### **4.3. Structure d'un système de fichiers**

Un système de fichiers **SF** peut se nommer système de gestion de fichiers **SGF** mais également file system **FS**.

Un système de fichiers est en charge des actions suivantes :

- Sécuriser les droits d'accès et de modification des fichiers ;
- Manipuler des fichiers : créer, lire, modifier et supprimer ;
- Localiser les fichiers sur le disque ;
- Gérer l'espace mémoire.

Le système d'exploitation Linux est capable d'exploiter différents systèmes de fichiers (ext2, ext3, ext4, FAT16, FAT32, NTFS, HFS, Btrfs, JFS, XFS, ...).

#### **4.3.1. Commande mkfs**

La commande mkfs permet de créer un système de fichiers Linux.

**Syntaxe de la commande mkfs.**

```
mkfs [-t fstype] filesys
```

Exemple :

```
[root]# mkfs -t ext4 /dev/sda1
```

Tableau 4.3. Option de la commande mkfs

Option	Description
-t	Indique le type de système de fichiers à utiliser



Sans système de fichiers il n'est pas possible d'utiliser l'espace disque.

Chaque système de fichiers possède une structure qui est identique sur chaque partition. Un Bloc de Boot et un Super Bloc initialisés par le système puis une Table des Inodes et une Zone de Données initialisées par l'administrateur.



La seule exception est concernant la partition **swap**.

### 4.3.2. Bloc de boot

Il occupe le premier bloc sur le disque et est présent sur toutes les partitions. Il contient le programme assurant le démarrage et l'initialisation du système et n'est donc renseigné que pour la partition de démarrage.

### 4.3.3. Super bloc

La taille de sa table est définie à la création. Il est présent sur chaque partition et contient les éléments nécessaires à l'exploitation de celle-ci.

Il décrit le Système de Fichiers :

- Nom du Volume Logique ;
- Nom du Système de Fichiers ;
- Type du Système de Fichiers ;

- État du Système de Fichiers ;
- Taille du Système de Fichiers ;
- Nombre de blocs libres ;
- Pointeur sur le début de la liste des blocs libres ;
- Taille de la liste des inodes ;
- Nombre et la liste des inodes libres.

Une copie est chargée en mémoire centrale dès l'initialisation du système. Cette copie est mise à jour dès modification et le système la sauvegarde périodiquement (commande sync). Lorsque le système s'arrête, il recopie également cette table en mémoire vers son bloc.

#### **4.3.4. Table des inodes**

La taille de la table des inodes est définie à sa création et est stockée sur la partition. Elle se compose d'enregistrements, appelés inodes, correspondant aux fichiers créés. Chaque enregistrement contient les adresses des blocs de données constituant le fichier.



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

Une copie est chargée en mémoire centrale dès l'initialisation du système. Cette copie est mise à jour dès modification et le système la sauvegarde périodiquement (commande sync). Lorsque le système s'arrête, il recopie également cette table en mémoire vers son bloc. Un fichier est géré par son numéro d'inode.



La taille de la table des inodes détermine le nombre maximum de fichiers que peut contenir le SF.

Informations présentes dans la table des inodes :

- Numéro d'inode ;
- Type de fichier et permissions d'accès ;
- Numéro d'identification du propriétaire ;

- Numéro d'identification du groupe propriétaire ;
- Nombre de liens sur ce fichier ;
- Taille du fichier en octets ;
- Date du dernier accès au fichier ;
- Date de la dernière modification du fichier ;
- Date de la dernière modification de l'inode (= création) ;
- Tableau de plusieurs pointeurs (table de blocs) sur les blocs logiques contenant les morceaux du fichier.

#### **4.3.5. Zone de données**

Sa taille correspond au reste de l'espace disponible de la partition. Cette zone contient les catalogues correspondant à chaque répertoire ainsi que les blocs de données correspondant aux contenus des fichiers.

**Afin de garantir la cohérence du système de fichiers**, une image du super-bloc et de la table des inodes est chargée en mémoire (RAM) lors du chargement du système d'exploitation afin que toutes les opérations d'E/S se fassent à travers ces tables du système. Lorsque l'utilisateur crée ou modifie des fichiers, c'est en premier lieu cette image mémoire qui est actualisée. Le système d'exploitation doit donc régulièrement actualiser le super-bloc du disque logique (commande sync).

Ces tables sont inscrites sur le disque dur lors de l'arrêt du système.



En cas d'arrêt brutal, le système de fichiers peut perdre sa cohérence et provoquer des pertes de données.

#### **4.3.6. Réparation du système de fichiers**

Il est possible de vérifier la cohérence d'un système de fichiers à l'aide de la commande **fsck**.

En cas d'erreurs, des solutions sont proposées afin de réparer les incohérences. Après réparation, les fichiers restant sans entrées dans la table des inodes sont rattachés au dossier **/lost+found** du lecteur logique.

## Commande fsck

**fsck** est un outil en mode console de contrôle d'intégrité et réparation pour les systèmes de fichiers Linux.

**Syntaxe de la commande fsck.**

```
fsck [-SACVRTNP] [ -t fstype ] filesys
```

Exemple :

```
[root]# fsck /dev/sda1
```

Pour vérifier la partition racine, il est possible de créer un fichier forcefsck et de redémarrer ou de faire un shutdown avec l'option -F.

```
[root]# touch /forcefsck  
[root]# reboot  
ou  
[root]# shutdown -r -F now
```



La partition devant être vérifiée doit impérativement être démontée.

## 4.4. Organisation d'un système de fichiers

Par définition, un Système de Fichiers est une structure arborescente de répertoires construite à partir d'un répertoire racine (un périphérique logique ne peut contenir qu'un seul système de fichiers).

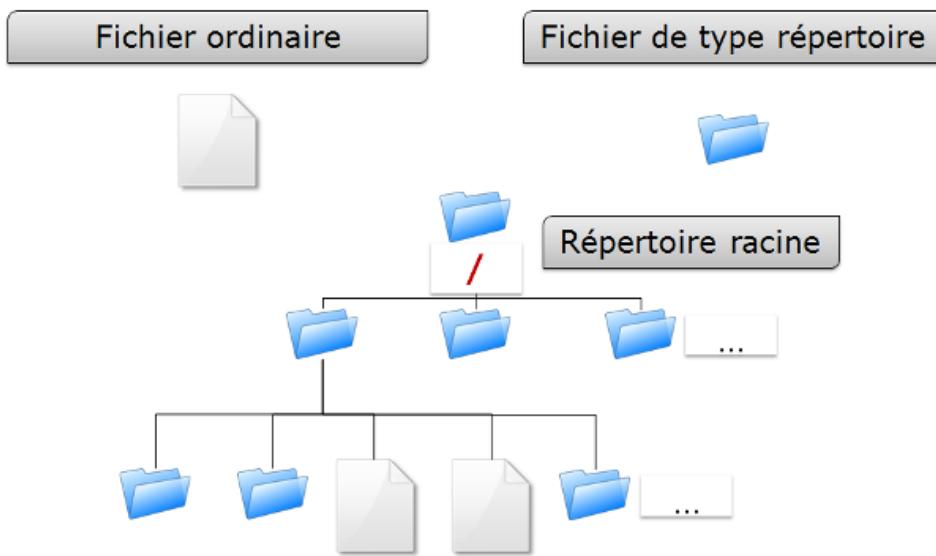


Figure 4.8. Organisation du système de fichiers



Sous Linux, tout est fichier.

Document texte, répertoire, binaire, partition, ressource réseau, écran, clavier, noyau Unix, programme utilisateur, ...

Linux répond à la norme FHS (Filesystems Hierarchy Standard) qui définit le nom des dossiers et leurs rôles.

Tableau 4.4. Organisation standard du système de fichiers

Répertoire	Observation	Abréviation
/	Contient les répertoires spéciaux	
/boot	Fichiers relatifs au démarrage du système	
/sbin	Commandes indispensables au démarrage système	system binaries
/bin	Exécutables des commandes de base du système	binaries
/usr/bin	Commandes d'administration système	
/lib	Librairies partagées et modules du noyau	libraries

Répertoire	Observation	Abréviation
/usr	Tout ce qui n'est pas nécessaire au fonctionnement minimal du système	UNIX System Resources
/mnt	Pour le montage de SF temporaires	mount
/media	Pour le montage de médias amovibles	
/root	Répertoire de connexion de l'administrateur	
/home	Données utilisateurs	
/tmp	Fichiers temporaires	temporary
/dev	Fichiers spéciaux des périphériques	device
/etc	Fichiers de configuration et de scripts	editable text configuration
/opt	Spécifiques aux applications installées	optional
/proc	Système de fichiers virtuel représentant les différents processus	processes
/var	Fichiers variables divers	variables

Montage, démontage...quelques affirmations :

- Pour effectuer un montage ou démontage, au niveau de l'arborescence, il ne faut pas se trouver sous le point de montage.
- Le montage sur un répertoire non vide n'efface pas le contenu. Il est seulement masqué.
- Seul l'administrateur peut effectuer des montages.
- Les points de montage devant être montés automatiquement au démarrage doivent être inscrit dans **/etc/fstab**.

#### 4.4.1. Le fichier /etc/fstab

Ce fichier est lu au démarrage du système et contient les montages à effectuer. Chaque système de fichiers à monter est décrit sur une seule ligne, les champs étant séparés par des espaces ou des tabulations.



Les lignes sont lues séquentiellement (fsck, mount, umount).

### Structure du fichier /etc/fstab.

/dev/mapper/VolGroup-lv_root	/	ext4	defaults	<b>1</b>	<b>1</b>
UUID=46...92	/boot	ext4	defaults	<b>1</b>	<b>2</b>
/dev/mapper/VolGroup-lv_swap	swap	swap	defaults	<b>0</b>	<b>0</b>
tmpfs	/dev/shm	tmpfs	defaults	<b>0</b>	<b>0</b>
devpts	/dev/pts	devpts	gid=5,mode=620	<b>0</b>	<b>0</b>
sysfs	/sys	sysfs	defaults	<b>0</b>	<b>0</b>
proc	/proc	proc	defaults	<b>0</b>	<b>0</b>
				<b>2</b>	<b>3</b>
				<b>4</b>	<b>5</b>
					<b>6</b>

Champ	Description
1	Périphérique du système de fichiers (/dev/sda1, UUID=..., ...)
2	Nom du point de montage, <b>chemin absolu</b> (excepté swap)
3	Type de système de fichiers (ext4, swap, ...)
4	Options particulières pour le montage (defaults, ro, ...)
5	Active ou non la gestion des sauvegardes (0:non sauvegardé, 1:sauvegardé)
6	Ordre de vérification lors du contrôle du SF par la commande fsck (0:pas de contrôle, 1:prioritaire, 2:non prioritaire)

La commande **mount -a** permet de prendre en compte les nouveaux montages sans redémarrage. Ils sont ensuite inscrits dans le fichier **/etc/mtab** qui contient les montages actuels.



Seuls les points de montages inscrits dans **/etc/fstab** seront montés au redémarrage.

Il est possible de faire une copie du fichier **/etc/mtab** ou de copier son contenu vers **/etc/fstab**.

#### **4.4.2. Commandes de gestion des montages**

##### **Commande mount**

La commande **mount** permet de monter et de visualiser les lecteurs logiques dans l'arborescence.

##### **Syntaxe de la commande mount.**

```
mount [-option] [device] [directory]
```

Exemple :

```
[root]# mount /dev/sda7 /home
```

**Tableau 4.5. Options de la commande mount**

Option	Description
-n	Monte sans écrire dans /etc/fstab
-t	Indique le type de système de fichiers à utiliser
-a	Monte tous les systèmes de fichiers mentionnés dans /etc/fstab
-r	Monte le système de fichiers en lecture seule (équivalent -o ro)
-w	Monte le système de fichiers en lecture/écriture, par défaut (équivalent -o rw)
-o	Argument suivi d'une liste d'option(s) séparée(s) par des virgules (remount, ro, ...)



La commande **mount** seule permet de visualiser tous les systèmes de fichiers montés.

##### **Commande umount**

La commande **umount** permet de démonter les lecteurs logiques.

##### **Syntaxe de la commande umount.**

```
umount [-option] [device] [directory]
```

Exemple :

```
[root]# umount /home
[root]# umount /dev/sda
```

**Tableau 4.6. Options de la commande umount**

Option	Description
-n	Démonte sans écrire dans /etc/fstab
-r	Si le démontage échoue, remonte en lecture seule
-f	Force le démontage
-a	Démonte tous les systèmes de fichiers mentionnés dans /etc/fstab



Pour le démontage, il ne faut pas rester en dessous du point de montage. Sinon, le message d'erreur suivant s'affiche : “device is busy”.

## 4.5. Types de fichiers

Comme dans tout système, afin de pouvoir se retrouver dans l’arborescence et la gestion des fichiers, il est important de respecter des règles de nommage des fichiers.

- Les fichiers sont codés sur 255 caractères ;
- Tous les caractères ASCII sont utilisables ;
- Les majuscules et minuscules sont différenciées ;
- Pas de notion d’extension.

Les groupes de mots séparés par des espaces doivent être encadrés par des guillements :

```
[root]# mkdir "repertoire travail"
```



Le . sert seulement à cacher un fichier quand il débute le nom.



Sous Linux, la notion d'extension n'existe pas. Cependant, elle peut être utilisée mais fait alors partie intégrante du nom du fichier.

Exemples de conventions d'extension :

- .c : fichier source en langage C ;
- .h : fichier d'entête C et Fortran ;
- .o : fichier objet en langage C ;
- .tar : fichier de données archivées avec l'utilitaire tar ;
- .cpio : fichier de données archivées avec l'utilitaire cpio ;
- .gz : fichier de données compressées avec l'utilitaire gzip ;
- .html : page web.

#### **4.5.1. Détails du nom d'un fichier**

```
[root]# ls -liah /usr/bin/passwd
18 -rwxr-xr-x. 1 root root 26K 22 févr. 2012 /usr/bin/passwd
1 2 3 4 5 6 7 8 9
```

Champ	Description
1	Numéro d'inode
2	Type de fichiers
3	Droits d'accès
4	Nombre de liens (ordinaire) ou sous-réertoires (réertoires)
5	Nom du propriétaire
6	Nom du groupe
7	Taille (octet, kilo, méga)
8	Date de la dernière mise à jour

Champ	Description
9	Nom du fichier

#### 4.5.2. Différents types de fichiers

On retrouve sur un système les types de fichiers suivants :

- Ordinaires (textes, binaires, ...);
- Répertoires ;
- Spéciaux (imprimantes, écrans, ...);
- Liens ;
- Communications (tubes et socket).

#### Fichiers ordinaires

Ce sont des fichiers textes, programmes (sources), exécutables (après compilation) ou fichiers de données (binaires, ASCII) et multimédias.

```
[root]# ls -l fichier
-rwxr-xr-x  1  root  root  26 nov  31 15:21 fichier
```

Le tiret - au début du groupe de droits indique qu'il s'agit d'un fichier de type ordinaire.

#### Fichiers répertoires

Les fichiers de type répertoire contiennent des références à d'autres fichiers.

Par défaut dans chaque répertoire sont présents . et .. .

Le . représente la position dans l'arborescence.

Le .. représente le père de la position courante.

```
[root]# ls -l repertoire
drwxr-xr-x  1  root  root  26 nov  31 15:21 repertoire
```

La lettre **d** au début du groupe de droits indique qu'il s'agit d'un fichier de type répertoire.

## Fichiers spéciaux

Afin de communiquer avec les périphériques (disques durs, imprimantes, ...), Linux utilise des fichiers d'interface appelés fichiers spéciaux (device file ou special file). Ils permettent donc d'identifier les périphériques.

Ces fichiers sont particuliers car ils ne contiennent pas de données mais spécifient le mode d'accès pour communiquer avec le périphérique.

Ils sont déclinés en deux modes :

- mode **bloc** ;
- mode **caractère**.

Le fichier spécial **mode bloc** permet en utilisant les buffers système de transférer des données vers le périphérique.

```
[root]# ls -l /dev/sda
brw----- 1 root root 8, 0 jan 1 1970 /dev/sda
```

La lettre **b** au début du groupe de droits indique qu'il s'agit d'un fichier spécial bloc.

Le fichier spécial **mode caractère** est utilisé pour transférer des données vers le périphérique sous forme de flux un caractère à la fois sans utiliser de buffer. Ce sont les périphériques comme l'imprimante, l'écran ou les bandes DAT, ...

La sortie standard est l'écran.

```
[root]# ls -l /dev/tty0
crw----- 1 root root 8, 0 jan 1 1970 /dev/tty0
```

La lettre **c** au début du groupe de droits indique qu'il s'agit d'un fichier spécial caractère.

## Fichiers de communication

Il s'agit des fichiers tubes (pipes) et des fichiers sockets.

**Les fichiers tubes** passent les informations entre processus par FIFO (first in first out). Un processus écrit de informations transitoires dans un fichier *pipe* et un autre les lit. Après lecture, les informations ne sont plus accessibles.

**Les fichiers sockets** permettent la communication bidirectionnelle interprocessus (sur système local ou distant). Ils utilisent un inode du système de fichiers.

## Fichiers liens

Ces fichiers donnent la possibilité de donner plusieurs noms logiques à un même fichier physique. Un nouveau point d'accès au fichier est par conséquent créé.

On distingue deux types de fichiers lien :

- Les liens physiques ;
- Les liens symboliques.

### Le lien physique

Le fichier lien et le fichier source ont le même numéro d'inode et le compteur de lien est incrémenté. Il est impossible de lier des répertoires et des fichiers de système de fichiers différents.



Si le fichier source est détruit, le compteur est décrémenté et le fichier lien accède toujours au fichier.

### Commande In

La commande **In** permet de créer des liens

```
[root]# ls -li lettre
666 -rwxr--r-- 1 root root ... lettre

[root]# ln /home/paul/lettre /home/jack/lire

[root]# ls -li /home/*/*1*
666 -rwxr--r-- 2 root root ... lettre
```

```
666 -rwxr--r-- 2 root root ... lire
```

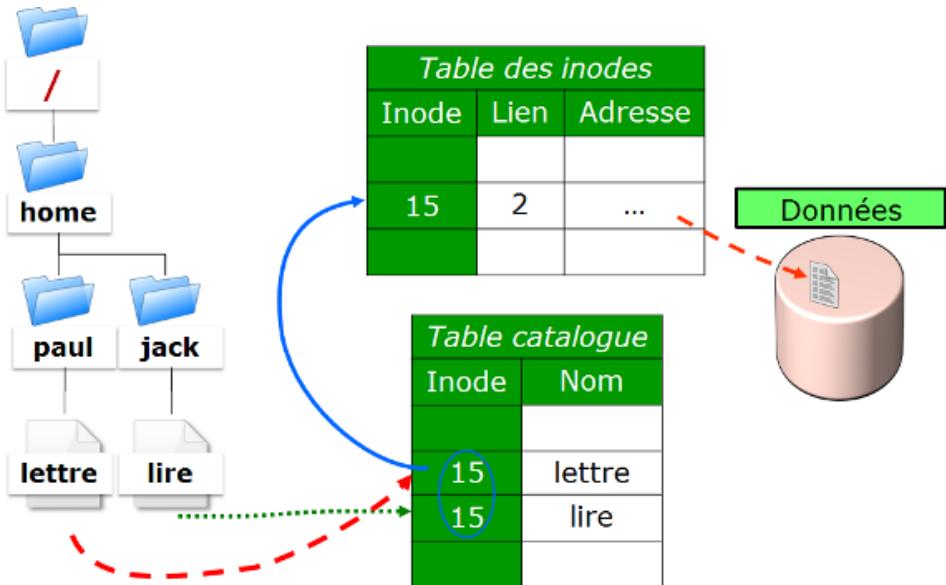


Figure 4.9. Représentation d'un lien physique

## Le lien symbolique

Contrairement au lien physique, le lien symbolique implique la création d'un nouvel **inode**. Au niveau du lien symbolique, seul un chemin d'accès est stocké dans la table des inodes.

Le fichier créé ne contient qu'une indication sur le chemin permettant d'atteindre le fichier. Cette notion n'a plus les limitations des liens physiques et il est désormais possible de lier des répertoires et des fichiers appartenant à des systèmes de fichiers différents.



Si le fichier source est détruit, le fichier lien ne peut plus accéder au fichier.

```
[root]# ls -li lettre
666 -rwxr--r-- 1 root root ... lettre
```

```
[root]# ln -s /home/paul/lettre /tmp/lire
```

```
[root]# ls -li /home/paul/lettre /tmp/lire
666 -rwxr--r--- 1 root root ... lettre
678 lrwxrwxrwx 1 root root ... lire -> lettre
```

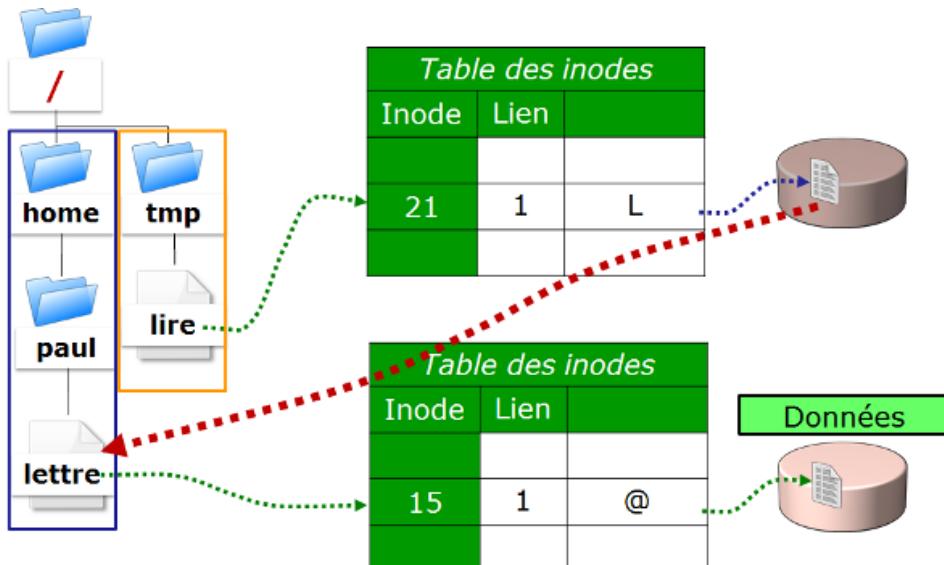


Figure 4.10. Représentation d'un lien symbolique

## 4.6. Attributs des fichiers

Linux est système d'exploitation multi-utilisateurs où l'accès aux fichiers est contrôlé.

Ces contrôles sont fonctions :

- des permissions d'accès au fichier ;
- des utilisateurs (ugo).

La commande **ls -l** permet afficher les attributs.

Il existe 4 droits d'accès aux fichiers :

- **read** (lecture) ;
- **write** (écriture) ;
- **execution** (exécution) ;
- - aucun droit.



Les droits associés aux fichiers diffèrent de ceux associés aux répertoires (voir ci-dessous).

Les types d'utilisateurs associés aux droits d'accès des fichiers sont :

- **user** (propriétaire) ;
- **group** (groupe propriétaire) ;
- **others** (les autres) ;

Dans certaines commandes, il est possible de désigner tout le monde avec **a** (all).

**a = ugo**

#### **4.6.1. Droits associés aux fichiers ordinaires**

- **read** : Permet la lecture d'un fichier (cat, less, ...) et autorise la copie (cp, ...).
- **write** : Autorise la modification du contenu du fichier (cat, », vim, ...).
- **execute** : Considère le fichier comme une commande (binaire, script).
- **-** : Aucune permission.



Déplacer ou renommer un fichier dépend des droits du répertoire cible. Supprimer un fichier dépend des droits du répertoire parent.

#### **4.6.2. Droits associés aux répertoires**

- **read** : Permet la lecture du contenu d'un répertoire (ls -R).
- **write** : Autorise la modification du contenu d'un répertoire (touch) et permet la **création et suppression de fichiers** si la permission **x** est activée.
- **execute** : Permet de descendre dans le répertoire (cd).
- **-** : Aucun droit.

#### **4.6.3. Gestion des attributs**

L'affichage des droits se fait à l'aide de la commande **ls -l**

```
[root]# ls -l /tmp/fichier
-rwxrw-r-x 1 root sys ... /tmp/fichier
 1 2 3     4      5
```

Champ	Description
1	Permissions du propriétaire ( <b>user</b> ), ici <b>rwx</b>
2	Permissions du groupe propriétaire ( <b>group</b> ), ici <b>rw-</b>
3	Permissions des autres utilisateurs ( <b>others</b> ), ici <b>r-x</b>
4	Propriétaire du fichier
5	Groupe propriétaire du fichier



Les permissions s'appliquent sur **user**, **group** et **other (ugo)** en fonction du propriétaire et du groupe.

Par défaut, le propriétaire d'un fichier est celui qui le crée. Le groupe du fichier est le groupe du propriétaire qui a créé le fichier. Les autres sont ceux qui ne sont pas concernés par les cas précédents.

La modification des attributs s'effectue à l'aide de la commande **chmod**

Seuls l'administrateur et le propriétaire d'un fichier peuvent modifier les droits d'un fichier.

### **Commande chmod**

La commande **chmod** permet de modifier les autorisations d'accès à un fichier.

```
chmod [option] mode fichier
```

L'indication de mode peut être une représentation octale (ex : 744) ou une représentation symbolique ([ugoa][+=-][rwxst]).

Plusieurs opérations symboliques peuvent être séparées par des virgules

Exemple :

```
[root]# chmod -R u+rwx,g+wx,o-r /tmp/fichier1
[root]# chmod g=x,o-r /tmp/fichier2
[root]# chmod -R o=r /tmp/fichier3

[root]# ls -l /tmp/fic*
-rwxrwx--- 1 root root ... /tmp/fichier1
-rwx--x--- 1 root root ... /tmp/fichier2
-rwx--xr-- 1 root root ... /tmp/fichier3
```

```
[root]# chmod 741 /tmp/fichier1
[root]# chmod -R 744 /tmp/fichier2

[root]# ls -l /tmp/fic*
-rwxr----x 1 root root ... /tmp/fichier1
-rwxr--r-- 1 root root ... /tmp/fichier2
```

Option	Observation
-R	Modifier récursivement les autorisations des répertoires et de leurs contenus

Il existe deux méthodes pour effectuer les changements de droits :

- La méthode **octale** ;
- La méthode **symbolique**.



Les droits des fichiers et des répertoires ne sont pas dissociés. Pour certaines opérations, il faudra connaître les droits du répertoire contenant le fichier.

Un fichier protégé en écriture peut être supprimé par un autre utilisateur dans la mesure où les droits du répertoire qui le contient autorisent cet utilisateur à effectuer cette opération.

### ***Principe de la méthode octale***

Chaque droit possède une valeur.

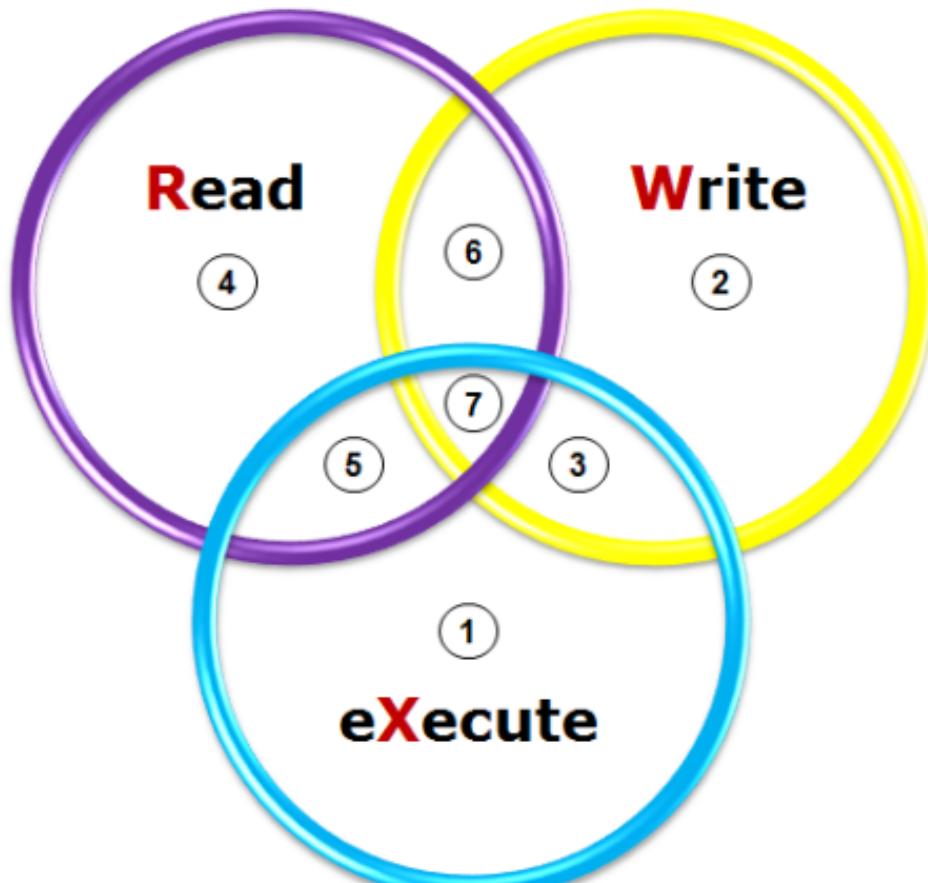


Figure 4.11. Méthode octale

```
[root]# ls -l /tmp/fichier  
-rwxrwxrwx 1 root root ... /tmp/fichier
```

user			group			other		
r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1

Figure 4.12. Droits 777

```
[root]# chmod 741 /tmp/fichier
-rwxr----x 1 root root ... /tmp/fichier
```

user			group			other		
4	2	1	4	0	0	0	0	1
r	w	x	r	-	-	-	-	x

Figure 4.13. Droits 741

### Principe de la méthode symbolique

Cette méthode peut être considérée comme une association “littérale” entre un type d’utilisateur, un opérateur et des droits.

Type utilisateur		Opérateur		Droits
User	u	Ajouter	+	Read r
Group	g	Enlever	-	Write w
Other	o	Remplacer	=	eXecute x
All	a			

Figure 4.14. Méthode symbolique

```
[root]# chmod u+rwx,g+wx,o-r /tmp/fichier
[root]# chmod g=x,o-r /tmp/fichier
[root]# chmod o=r /tmp/fichier
```

```
[root]# ls -l /tmp/fichier
----r--r-- 1 root root ... /tmp/fichier

[root]# chmod u+rwx,g+wx,o-r /tmp/fichier
```

```
[root]# ls -l /tmp/fichier
-rwxrwx--- 1 root root ... /tmp/fichier
```

#### 4.6.4. Les droits particuliers

En complément des droits fondamentaux (rwx), il existe les droits particuliers :

- set-user-ID
- set-group-ID (SGID)
- sticky-bit (SUID)

Comme pour les droits fondamentaux, les droits particuliers possèdent chacun une valeur. Celle-ci se place avant l'ensemble de droits ugo.

suid	sgid	sticky-bit	user	group	other
s	s	t	r w x	r w x	r w x
4	2	1	4 2 1	4 2 1	4 2 1

Figure 4.15. Les droits particuliers

S, S et T en majuscules si le droit n'existe pas.



#### Le Sticky-bit

Une des particularités des droits sous Linux est que le droit d'écrire sur un répertoire permet également de supprimer **tous** les fichiers, propriétaire ou non.

Le sticky-bit positionné sur le répertoire ne permettra aux utilisateurs d'effacer que les fichiers dont ils sont propriétaires.

La mise en place du sticky-bit peut s'effectuer comme ci-dessous :

Méthode octale :

```
[root]# chmod 1777 repertoire
```

Méthode symbolique :

```
[root]# chmod o+t repertoire
```

```
[root]# ls -l  
drwxrwxrwt ... repertoire
```

## SUID et SGID

Ces droits permettent d'exécuter une commande suivant les droits positionnés sur la commande et non plus suivant les droits de l'utilisateur.

La commande s'exécute avec l'identité du propriétaire (**suid**) ou du groupe (**sgid**) de la commande.



L'identité de l'utilisateur demandant l'exécution de la commande n'est plus prise en compte.

Il s'agit d'une possibilité supplémentaire de droits d'accès attribués à un utilisateur lorsqu'il est nécessaire qu'il dispose des mêmes droits que ceux du propriétaire d'un fichier ou ceux du groupe concerné.

En effet, un utilisateur peut avoir à exécuter un programme (en général un utilitaire système) mais ne pas avoir les droits d'accès nécessaires. En positionnant les droits adéquats ("s" au niveau du propriétaire et/ou au niveau du groupe), l'utilisateur du programme possède, pour le temps d'exécution de celui-ci, l'identité du propriétaire (ou celle du groupe) du programme.

Exemple :

Le fichier **/usr/bin/passwd** est un fichier exécutable (une commande) qui porte un **SUID**.

Lorsque l'utilisateur bob va le lancer, ce dernier devra accéder au fichier **/etc/shadow**, or les droits sur ce fichier ne permettent pas à bob d'y accéder.

Ayant un **SUID** cette commande sera exécutée avec l'UID de root et le GID de root. Ce dernier étant le propriétaire du fichier **/etc/shadow**, il aura les droits en lecture.

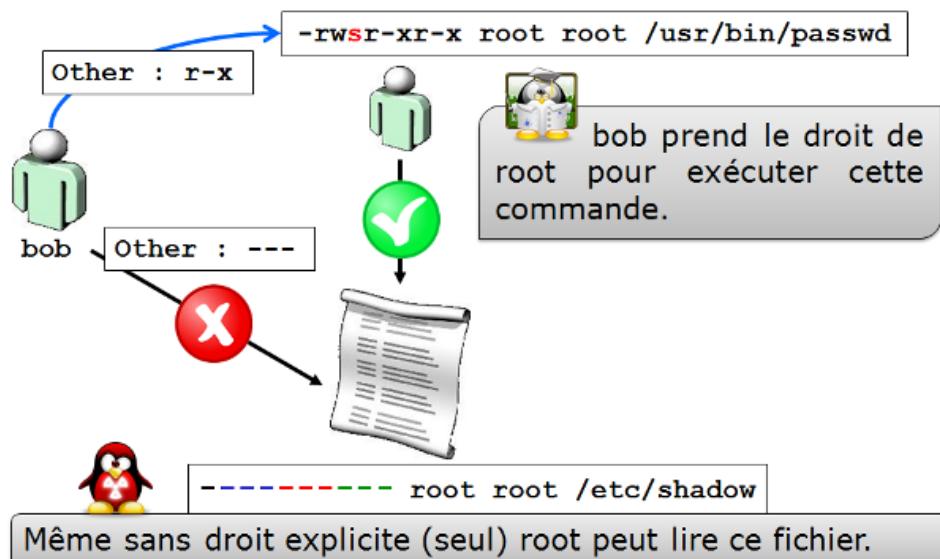


Figure 4.16. Fonctionnement du SUID

La mise en place du SUID et SGID peut s'effectuer comme ci-dessous :

Méthode octale :

```
[root]# chmod 4777 commande1
[root]# chmod 2777 commande2
```

Méthode symbolique :

```
[root]# chmod u+s commande1
[root]# chmod g+s commande2
```

```
[root]# ls -l
-rwsrwxrwx ... commande1
-rwxrwsrwx ... commande2
```

## 4.7. Droits par défaut et masque

Lors de sa création, un fichier ou un répertoire possède déjà des permissions.

- Pour un répertoire : **rwxr-xr-x** soit **755**

- Pour un fichier : **rw-r-r-** soit **644**

Ce comportement est défini par le **masque par défaut**.

Le principe est d'enlever la valeur définie du masque aux droits maximums.

Pour un répertoire :

Droits max	umask	Droits eff.
777	022	755
rwxrwxrwx	----w--w-	rwxr-xr-x

Figure 4.17. Droits par défaut d'un répertoire

Pour un fichier, les droits d'exécution sont retirés :

Droits eff.	Droits exé.	Droits fic.
755	111	644
rwxr-xr-x	--x--x--x	r--r--r--

Figure 4.18. Droits par défaut d'un fichier

#### 4.7.1. Commande umask

La commande **umask** permet d'afficher et de modifier le masque.

```
umask [option] [mode]
```

Exemple :

```
[root]# umask  
0033  
[root]# umask 025  
[root]# umask  
0025
```

**Tableau 4.7. Options de la commande umask**

Option	Description
-S	Affichage symbolique



**umask** n'affecte pas les fichiers existants.



**umask** modifie le masque jusqu'à la déconnexion. Pour garder la valeur, il faut modifier les fichiers de profile suivants :

Pour tous les utilisateurs :

- `/etc/profile`
- `/etc/bashrc`

Pour un utilisateur en particulier :

- `~/.bashrc`

---

# 5

## Gestion des processus

---

### 5.1. Généralités

Un système d'exploitation se compose de processus. Ces derniers sont exécutés dans un ordre bien précis et observent des liens de parenté entre eux. On distingue deux catégories de processus, ceux axés sur l'environnement utilisateur et ceux sur l'environnement matériel.

Lorsqu'un programme s'exécute, le système va créer un processus en plaçant les données et le code du programme en mémoire et en créant **une pile d'exécution**. Un processus est donc une instance d'un programme auquel est associé un environnement processeur (Compteur Ordinal, registres, etc.) et un environnement mémoire.

Chaque processus dispose :

- d'un **PID** : Process IDentifiant, identifiant unique de processus ;
- d'un **PPID** : Parent Process IDentifiant, identifiant unique de processus parent.

Par filiations successives, le processus **init** est le père de tous les processus.

- Un processus est toujours créé par un processus père ;
- Un processus père peut avoir plusieurs processus fils.

Il existe une relation père / fils entre les processus, un processus fils est le résultat de l'appel système de la primitive fork() par le processus père qui duplique son propre code pour créer un fils. Le PID du fils est renvoyé au processus père pour qu'il puisse dialoguer avec. Chaque fils possède l'identifiant de son père, le **PPID**.

Le numéro PID représente le processus au moment de son exécution. À la fin de celui-ci, le numéro est de nouveau disponible pour un autre processus. Exécuter plusieurs fois la même commande produira à chaque fois un PID différent.

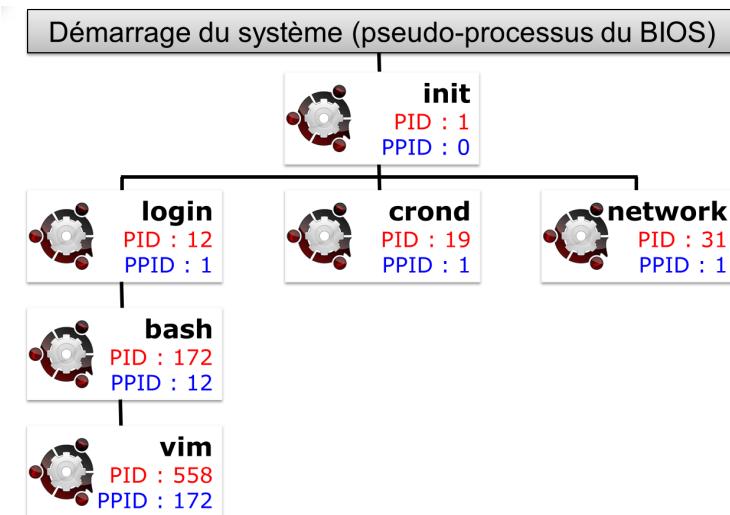


Figure 5.1. Filiation des processus



Les processus ne sont pas à confondre avec les threads. Chaque processus possède son propre contexte mémoire (ressources et espace d'adressage) alors que les threads issus d'un même processus partagent ce même contexte.

## 5.2. Visualisation des processus

La commande **ps** affiche l'état des processus en cours.

### Syntaxe de la commande ps.

```
ps [-e] [-f] [-u login]
```

Exemple :

```
[root]# ps -fu root
```

**Tableau 5.1. Options principales de la commande ps**

<b>Option</b>	<b>Description</b>
-e	Affiche tous les processus.
-f	Affiche des informations supplémentaires.
-u login	Affiche les processus de l'utilisateur.

Quelques options supplémentaires :

**Tableau 5.2. Options supplémentaires de la commande ps**

<b>Option</b>	<b>Description</b>
-g	Affiche les processus du groupe.
-t tty	Affiche les processus exécutés à partir du terminal.
-p PID	Affiche les informations du processus.
-H	Affiche les informations sous forme d'arborescence.
-l	Affiche des informations supplémentaires.

Sans option précisée, la commande ps n'affiche que les processus exécutés à partir du terminal courant.

Le résultat est affiché par colonnes :

```
[root]# ps -ef
UID  PID  PPID  C STIME   TTY TIME      CMD
root  1    0     0 Jan01 ?  00:00/03 /sbin/init
```

**Tableau 5.3. Descriptions des colonnes du résultat de la commande ps -ef**

<b>Colonne</b>	<b>Description</b>
UID	Utilisateur propriétaire.
PID	Identifiant du processus.
PPID	Identifiant du processus parent.
C	Priorité du processus.
STIME	Date et heure d'exécution.
TTY	Terminal d'exécution.

Colonne	Description
TIME	Durée de traitement.
CMD	Commande exécutée.

### 5.3. Types de processus

Le processus utilisateur :

- il est démarré depuis un terminal associé à un utilisateur ;
- il accède aux ressources via des requêtes ou des démons.

Le processus système (démon) :

- il est démarré par le système ;
- il n'est associé à aucun terminal et son propriétaire est un utilisateur système (souvent root) ;
- il est chargé lors du démarrage, il réside en mémoire et est en attente d'un appel ;
- il est généralement identifié par la lettre "d" associé au nom du processus.

Les processus système sont donc appelés démons de l'anglais **daemon** (Disk And Execution MONitor).

### 5.4. Permissions et droits

À l'exécution d'une commande, les identifiants de l'utilisateur sont transmis au processus créé.

Par défaut, les **UID** et **GID** effectifs (du processus) sont donc identiques aux **UID** et **GID réels** (les UID et GID de l'utilisateur qui a exécuté la commande).

Lorsqu'un **SUID** (et/ou un **SGID**) est positionné sur une commande, les UID (et/ou GID) effectifs deviennent ceux du propriétaire (et/ou du groupe propriétaire) de la commande et non plus celui de l'utilisateur ou du groupe utilisateur qui a lancé la commande. **UID effectifs et réels** sont donc **différents**.

À chaque accès à un fichier, le système vérifie les droits du processus en fonction de ses identifiants effectifs.

## 5.5. Gestion des processus

Un processus ne peut pas être exécuté indéfiniment, cela se ferait au détriment des autres processus en cours et empêcherait de faire du multitâche.

Le temps de traitement total disponible est donc divisé en petites plages et chaque processus (doté d'une priorité) accède au processeur de manière séquencée. Le processus prendra plusieurs états au cours de sa vie parmi les états :

- prêt : attend la disponibilité du processus ;
- en exécution : accède au processeur ;
- suspendu : en attente d'une E/S (entrée/sortie) ;
- arrêté : en attente d'un signal d'un autre processus ;
- zombie : demande de destruction ;
- mort : le père du processus tue son fils.

Le séquencement de fin de processus est le suivant :

1. Fermeture des fichiers ouverts ;
2. Libération de la mémoire utilisée ;
3. Envoi d'un signal au processus père et aux processus fils.

Lorsqu'un processus meurt, ses processus fils sont dits orphelins. Ils sont alors adoptés par le processus `init` qui se chargera de les détruire.

### 5.5.1. La priorité d'un processus

Le processeur travaille en temps partagé, chaque processus occupe un quantum de temps du processeur.

Les processus sont classés par priorité dont la valeur varie de -20 (la priorité la plus élevée) à +20 (la priorité la plus basse).

La priorité par défaut d'un processus est 0.

### 5.5.2. Modes de fonctionnements

Les processus peuvent fonctionner de deux manières :

- **synchrone** : l'utilisateur perd l'accès au shell durant l'exécution de la commande. L'invite de commande réapparaît à la fin de l'exécution du processus.
- **asynchrone** : le traitement du processus se fait en arrière-plan, l'invite de commande est réaffichée immédiatement.

Les contraintes du mode asynchrone :

- la commande ou le script ne doit pas attendre de saisie au clavier ;
- la commande ou le script ne doit pas retourner de résultat à l'écran ;
- quitter le shell termine le processus.

## 5.6. Les commandes de gestion des processus

### 5.6.1. La commande kill

La commande kill envoie un signal d'arrêt à un processus.

**Syntaxe de la commande kill.**

```
kill [-signal] PID
```

Exemple :

```
[root]# kill -9 1664
```

Tableau 5.4. Codes numériques des principaux signaux d'arrêt des processus

Code	Signal	Description
2	SIGINT	Interruption du processus (CTRL + D)
9	SIGKILL	Terminaison immédiate du processus
15	SIGTERM	Terminaison propre du processus
18	SIGCONT	Reprise du processus
19	SIGSTOP	Suspension du processus

Les signaux sont les moyens de communication entre les processus. La commande kill permet d'envoyer un signal à un processus.



La liste complète des signaux pris en compte par la commande kill est disponible en tapant la commande :

```
[root]# man 7 signal
```

### **5.6.2. La commande nohup**

La commande **nohup** permet de lancer un processus indépendant d'une connexion.

#### **Syntaxe de la commande nohup.**

```
nohup commande
```

Exemple :

```
[root]# nohup MonProgramme.sh 0</dev/null &
```

Nohup ignore le signal SIGHUP envoyé lors de la déconnexion d'un utilisateur.



Nohup gère les sorties et erreur standards, mais pas l'entrée standard, d'où la redirection de cette entrée vers /dev/null.

### **[CTRL] + [Z]**

En appuyant simultanément sur les touches **CTRL+Z**, le processus synchrone est temporairement suspendu. L'accès au prompt est rendu après affichage du numéro du processus venant d'être suspendu.

### **5.6.3. Instruction &**

L'instruction **&** exécute la commande en mode asynchrone (la commande est alors appelée job) et affiche le numéro de job. L'accès au prompt est ensuite rendu.

Exemple :

```
[root]# time ls -lR / > list.ls 2> /dev/null &
```

```
[1] 15430
[root]#
```

Le numéro de **job** est obtenu lors de la mise en tâche de fond et est affiché entre crochets, suivi du numéro de PID.

#### **5.6.4. Les commandes fg et bg**

La commande **fg** place le processus au premier plan :

```
[root]# time ls -lR / > list.ls 2>/dev/null &
[root]# fg 1
time ls -lR / > list.ls 2>/dev/null
```

tandis que la commande **bg** le place à l'arrière plan :

```
[CTRL]+[Z]
^Z
[1]+ Stopped
[root]# bg 1
[1] 15430
[root]#
```

Qu'il ait été mis à l'arrière plan lors de sa création grâce à l'argument **&** ou plus tard avec les touches **CTRL+Z**, un processus peut être ramené au premier plan grâce à la commande **fg** et à son numéro de job.

#### **5.6.5. La commande jobs**

La commande **jobs** affiche la liste des processus tournant en tâche de fond et précise leur numéro de job.

Exemple :

```
[root]# jobs
[1]- Running      sleep 1000
[2]+ Running      find / > arbo.txt
```

Les colonnes représentent :

1. le numéro de job ;
2. ordre de passage des processus
  - un + : le processus est le prochain processus à s'exécuter par défaut avec fg ou bg ;
  - un - : le processus est le prochain processus qui prendra le + ;
3. Running (en cours de traitement) ou Stopped (processus suspendu).
4. la commande

### **5.6.6. Les commandes nice/renice**

La commande **nice** permet d'exécuter une commande en précisant sa priorité.

#### **Syntaxe de la commande nice.**

```
nice priorité commande
```

Exemple :

```
[root]# nice -n+15 find / -name "fichier"
```

Contrairement à root, un utilisateur standard ne peut que réduire la priorité d'un processus. Seules les valeurs entre +0 et +19 seront acceptées.

La commande **renice** permet de modifier la priorité d'un processus en cours d'exécution.

#### **Syntaxe de la commande renice.**

```
renice priorité [-g GID] [-p PID] [-u UID]
```

Exemple :

```
[root]# renice +15 -p 1664
```

Tableau 5.5. Options principales de la commande renice

Option	Description
-g	GID du groupe propriétaire du processus.

Option	Description
-p	PID du processus.
-u	UID du propriétaire du processus.

La commande **renice** agit sur des processus déjà en cours d'exécution. Il est donc possible de modifier la priorité d'un processus précis, mais aussi de plusieurs processus appartenant à un utilisateur ou à un groupe.

### 5.6.7. La commande top

La commande top affiche les processus et leur consommation en ressources.

```
[root]# top
PID  USER PR NI ... %CPU %MEM   TIME+ COMMAND
2514 root 20 0      15    5.5 0:01.14   top
```

Tableau 5.6. Descriptions des colonnes du résultat de la commande top

Colonne	Description
PID	Identifiant du processus.
USER	Utilisateur propriétaire.
PR	Priorité du processus.
NI	Valeur du nice.
%CPU	Charge du processeur.
%MEM	Charge de la mémoire.
TIME+	Temps d'utilisation processeur.
COMMAND	Commande exécutée.

La commande **top** permet de contrôler les processus en temps réel et en mode interactif.

---

# 6

## Sauvegardes et restaurations

---

La sauvegarde va permettre de répondre à un besoin de conservation et de restauration des données de manière sûre et efficace.

La sauvegarde permet de se protéger des éléments suivants :

- **Destruction** : Volontaire ou involontaire. Humaine ou technique. Virus, ...
- **Suppression** : Volontaire ou involontaire. Humaine ou technique. Virus, ...
- **Intégrité** : Données devenues inutilisables.

Aucun système n'est infaillible, aucun humain n'est infaillible, aussi pour éviter de perdre des données, il faut les sauvegarder pour être en mesure de les restaurer suite à un problème.

Les supports de sauvegarde sont conservés dans une autre pièce (voire bâtiment) que le serveur afin qu'un sinistre ne vienne pas détruire le serveur et les sauvegardes.

De plus, l'administrateur devra régulièrement vérifier que les supports soient toujours lisibles.

### 6.1. Généralités

Il existe deux principes, la **sauvegarde** et l'**archive**.

- L'archive détruit la source d'information après l'opération.
- La sauvegarde conserve la source d'information après l'opération.

Ces opérations consistent à enregistrer des informations dans un fichier, sur un périphérique ou un support (bandes, disques, ...).

### 6.1.1. La démarche

La sauvegarde nécessite de l'administrateur du système beaucoup de discipline et une grande rigueur. Il est nécessaire de se poser les questions suivantes :

- Quel est le support approprié ?
- Que faut-il sauvegarder ?
- Nombre d'exemplaires ?
- Durée de la sauvegarde ?
- Méthode ?
- Fréquence ?
- Automatique ou manuelle ?
- Où la stocker ?
- Délai de conservation ?

### 6.1.2. Méthodes de sauvegardes

- **Complète** : un ou plusieurs **systèmes de fichiers** sont sauvegardés (noyau, données, utilitaires, ...).
- **Partielle** : un ou plusieurs **fichiers** sont sauvegardés (configurations, répertoires, ...).
- **Incrémentale** : uniquement les **fichiers modifiés** depuis la dernière sauvegarde sont sauvegardés.

### 6.1.3. Périodicité

- **Ponctuelle** : à un instant donné (avant une mise à jour du système, ...).
- **Périodique** : Journalière, hebdomadaire, mensuelle, ...



Avant une modification du système, il peut être utile de faire une sauvegarde. Cependant, il ne sert à rien de

sauvegarder tous les jours des données qui ne sont modifiées que tous les mois.

#### **6.1.4. Méthodes de restauration**

En fonction des utilitaires disponibles, il sera possible d'effectuer plusieurs types de restaurations.

- **Restauration complète** : arborescences, ...
- **Restauration sélective** : partie d'arborescences, fichiers, ...

Il est possible de restaurer la totalité d'une sauvegarde mais il est également possible d'en restaurer uniquement une partie. Toutefois, lors de la restauration d'un répertoire, les fichiers créés après la sauvegarde ne sont pas supprimés.



Pour récupérer un répertoire tel qu'il était au moment de la sauvegarde il convient d'en supprimer complètement le contenu avant de lancer la restauration.

#### **6.1.5. Les outils**

Il existe de nombreux utilitaires pour réaliser les sauvegardes.

- **outils éditeurs** ;
- **outils graphiques** ;
- **outils mode de commande** : **tar**, **cpio**, **pax**, **dd**, **dump**, ...

Les commandes que nous verrons ici sont tar et cpio.

- tar :
  - simple d'utilisation ;
  - permet l'ajout de fichiers à une sauvegarde existante.
- cpio :
  - conserve les propriétaires ;
  - groupes, dates et droits ;
  - saute les fichiers endommagés ;

- système de fichiers complet.



Ces commandes sauvegardent dans un format propriétaire et standardisé.

### **6.1.6. Convention de nommage**

L'emploi d'une convention de nommage permet de cibler rapidement le contenu d'un fichier de sauvegarde et d'éviter ainsi des restaurations hasardeuses.

- nom du répertoire ;
- utilitaire employé ;
- options utilisées ;
- date.



Le nom de la sauvegarde doit être un nom explicite.



La notion d'extension sous Unix n'existe pas.

### **6.1.7. Contenu d'une sauvegarde**

Une sauvegarde contient généralement les éléments suivants :

- le fichier ;
- le nom ;
- le propriétaire ;
- la taille ;
- les permissions ;
- date d'accès.



Le numéro d'inode est absent.

### **6.1.8. Modes de stockage**

Deux modes de stockage se distinguent :

- fichier sur le disque ;
- périphérique.

## 6.2. Tape ArchiveR - tar

La commande tar permet la sauvegarde sur plusieurs supports successifs (options multi-volumes).

Il est possible d'extraire tout ou partie d'une sauvegarde.

Tar sauvegarde implicitement en mode relatif même si le chemin des informations à sauvegarder est mentionné en mode absolu.

### 6.2.1. Consignes de restauration

Il faut se poser les bonnes questions

- quoi : Partielle ou complète ;
- où : Lieu où les données seront restaurées ;
- comment : Absolu ou relatif.



Avant une restauration, il faut prendre le temps de la réflexion et déterminer la méthode la mieux adaptée afin d'éviter toutes erreurs.

Les restaurations s'effectuent généralement après un problème qui doit être résolu rapidement. Une mauvaise restauration peut dans certains cas aggraver la situation.

### 6.2.2. La sauvegarde avec tar

L'utilitaire par défaut pour créer des archives dans les systèmes UNIX est la commande tar. Ces archives peuvent être compressées avec une compression gzip ou bzip.

Tar permet d'extraire aussi bien un seul fichier ou un répertoire d'une archive, visualiser son contenu ou valider son intégrité, etc.

#### **Créer une archive**

Créer une archive non-compressée s'effectue avec les clefs cvf :

**Syntaxe de la commande tar pour créer une archive.**

```
tar c[vf] [support] [fichiers(s)]
```

Exemple :

```
[root]# tar cvf /sauvegardes/home.133.tar
```

Tableau 6.1. Clefs principales de la commande tar

Clef	Description
c	Crée une sauvegarde.
v	Affiche le nom des fichiers traités.
f	Permet d'indiquer le nom de la sauvegarde (support).



Il n'y a pas de tiret '-' devant les clefs de tar !

## ***Créer une sauvegarde en mode absolu***

**Syntaxe de la commande tar pour créer une archive en mode absolu.**

```
tar c[vf]P [support] [fichiers(s)]
```

Exemple :

```
[root]# tar cvfP /sauvegardes/home.133.P.tar /home
```

Clef	Description
P	Créer une sauvegarde en mode absolu.



Avec la clef **P**, le chemin des fichiers à sauvegarder doit être renseigné en **absolu**. Si les deux conditions (clef **P** et chemin **absolu**) ne sont pas indiquées, la sauvegarde est en mode relatif.

## ***Créer une archive compressée avec gzip***

Créer une archive compressée en gzip s'effectue avec les clefs **cvzf** :

```
[root]# tar cvzf archive.tar.gz dirname/
```

Clef	Description
z	Comprime l'archive en gzip.



L'extension .tgz est une extension équivalente à .tar.gz



Conserver les clefs 'cvf' ('tvf' ou 'xvf') inchangée pour toutes les manipulations d'archives et simplement ajouter à la fin des clefs celle de compression simplifie la compréhension de la commande (par exemple 'cvfz' ou 'cvfj', etc.).

## ***Créer une archive compressée avec bzip***

Créer une archive compressée en bzip s'effectue avec les clefs cvfj :

```
[root]# tar cvfj archive.tar.bz2 dirname/
```

Clef	Description
j	Comprime l'archive en bzip2.



Les extensions .tbz et .tb2 sont des extensions équivalentes à .tar.bz2

## ***gzip vs bzip2***

bzip2 nécessite plus de temps pour compresser ou décompresser que gzip mais offre des ratios de compression supérieurs.

## ***Extraire (untar) une archive***

Extraire une archive \*.tar s'effectue avec les clefs xvf :

```
[root]# tar xvf /sauvegardes/etc.133.tar etc/exports
[root]# tar xvfh /sauvegardes/home.133.tar.b2
[root]# tar xvfp /sauvegardes/etc.133.P.tar
```



Se placer au bon endroit.

Vérifier le contenu de la sauvegarde.

Clef	Description
x	Extrait des fichiers de l'archive, compressée ou non.

Extraire une archive tar-gzippée (\*.tar.gz) s'effectue avec les clefs xvfz

```
[root]# tar xvfz archive.tar.gz
```

Extraire une archive tar-bzippée (\*.tar.bz2) s'effectue avec les clefs xvfb

```
[root]# tar xvfb archive.tar.bz2
```

### ***Lister le contenu d'une archive***

Visualiser le contenu d'une archive sans l'extraire s'effectue avec les clefs tvf :

```
[root]# tar tvf archive.tar
[root]# tar tvfz archive.tar.gz
[root]# tar tvfb archive.tar.bz2
```

Lorsque le nombre de fichiers contenus dans une archive devient important, il est possible de passer à la commande less le résultat de la commande tar par un pipe ou en utilisant directement la commande less :

```
[root]# tar tvf archive.tar | less
[root]# less archive.tar
```

### ***Extraire uniquement un fichier d'une archive .tar, tar.gz ou tar.bz2***

Pour extraire un fichier spécifique d'une archive tar, spécifier le nom du fichier à la fin de la commande tar xvf.

```
[root]# tar xvf archive.tar /path/to/file
```

La commande précédente permet de n'extraire que le fichier file de l'archive archive.tar.

```
[root]# tar xvfz archive.tar.gz /path/to/file  
[root]# tar xvfj archive.tar.bz2 /path/to/file
```

### ***Extraire uniquement un dossier d'une archive tar, tar.gz, tar.bz2***

Pour n'extraire qu'un seul répertoire (ses sous-répertoires et fichiers inclus) d'une archive, spécifier le nom du répertoire à la fin de la commande tar xvf.

```
[root] tar xvf archive.tar /path/to/dir/
```

Pour extraire plusieurs répertoires, spécifier chacun des noms les uns à la suite des autres :

```
[root] tar xvf archive_file.tar /path/to/dir1/ /path/to/dir2/  
[root] tar xvfz archive_file.tar.gz /path/to/dir1/ /path/to/dir2/  
[root] tar xvfj archive_file.tar.bz2 /path/to/dir1/ /path/to/dir2/
```

### ***Extraire un groupe de fichiers d'une archive tar, tar.gz, tar.bz2 grâce à des expressions régulières (regex)***

Spécifier une regex pour extraire les fichiers correspondants au pattern spécifié.

Par exemple, pour extraire tous les fichiers avec l'extension .conf :

```
[root] tar xvf archive_file.tar --wildcards '*.*.conf'
```

Clefs :

- --wildcards \*.conf correspond aux fichiers avec l'extension .conf.

### ***Ajouter un fichier ou un répertoire à une archive existante***

Il est possible d'ajouter des fichiers à une archive existante avec la clef r.

Par exemple, pour ajouter un fichier :

```
[root]# tar rvf archive.tar filetoadd
```

Le fichier filetoadd sera ajouté à l'archive tar existante. Ajouter un répertoire est similaire :

```
[root]# tar rvf archive_name.tar dirtoadd
```



Il n'est pas possible d'ajouter des fichiers ou des dossiers à une archive compressée.

```
[root]# tar rvfz archive.tgz filetoadd
tar: Cannot update compressed archives
Try `tar --help' or `tar --usage' for more
information.
```

## Vérifier l'intégrité d'une archive

L'intégrité d'une archive peut être testée avec la clef W au moment de sa création :

```
[root]# tar cvfw file_name.tar dir/
```

La clef W permet également de comparer le contenu d'une archive par rapport au système de fichiers :

```
[root]# tar tvfw file_name.tar
Verify 1/file1
1/file1: Mod time differs
1/file1: Size differs
Verify 1/file2
Verify 1/file3
```

La vérification avec la clef W ne peut pas être effectuée avec une archive compressée. Il faut utiliser la clef d :

```
[root]# tar dfz file_name.tgz
[root]# tar dfj file_name.tar.b2z
```

## ***Estimer la taille d'une archive***

La commande suivante estime la taille d'un fichier tar en KB avant de la créer :

```
[root]# tar cf - /directory/to/archive/ | wc -c
20480
[root]# tar czf - /directory/to/archive/ | wc -c
508
[root]# tar cjf - /directory/to/archive/ | wc -c
428
```

## ***Ajout d'éléments à une sauvegarde existante***

**Syntaxe de la commande tar pour ajouter un élément à une sauvegarde existante.**

```
tar {r|A}[clé(s)] [support] [fichiers(s)]
```

Exemple :

```
[root]# tar rvf /sauvegardes/home.133.tar /etc/passwd
```

Clef	Description
r	Ajoute un ou plusieurs fichiers à la fin d'une sauvegarde sur support à accès direct (disque dur).
A	Ajoute un ou plusieurs fichiers à la fin d'une sauvegarde sur support à accès séquentiel (bande).



Si la sauvegarde a été réalisée en mode relatif, ajoutez des fichiers en mode relatif. Si la sauvegarde a été réalisée en mode absolu, ajoutez des fichiers en mode absolu. En mélangeant les modes, vous risquez d'avoir des soucis au moment de la restauration.

## ***Lire le contenu d'une sauvegarde***

**Syntaxe de la commande tar pour lire le contenu d'une sauvegarde.**

```
tar t[clé(s)] [support]
```

Exemple :

```
[root]# tar tvf /sauvegardes/home.133.tar  
[root]# tar tvfj /sauvegardes/home.133.tar.bz2
```

Clef	Description
t	Affiche le contenu d'une sauvegarde (compressée ou non).



Toujours vérifier le contenu d'une sauvegarde.

Tableau 6.2. Convention d'écriture de la commande Tar

Clés	Fichiers	Suffixe
cvf	home	home.tar
cvfP	/etc	etc.P.tar
cvfz	usr	usr.tar.gz
cvfj	usr	usr.tar.bz2
cvfPz	/home	home.P.tar.gz
cvfPj	/home	home.P.tar.bz2

## 6.3. CoPy Input Output - cpio

La commande cpio permet la sauvegarde sur plusieurs supports successifs sans indiquer d'options.

Il est possible d'extraire tout ou partie d'une sauvegarde.



cpio ne permet pas de sauvegarder directement une arborescence. L'arborescence ou fichiers sont donc transmis sous forme de liste à cpio.

Il n'y a aucune option, comme pour la commande tar, permettant de sauvegarder et de compresser en même temps. Cela s'effectue donc en deux temps : la sauvegarde puis la compression.

Pour effectuer une sauvegarde avec cpio, il faut préciser une liste des fichiers à sauvegarder.

Cette liste est fourni avec les commandes find, ls ou cat.

- find : parcourt une arborescence, récursif ou non ;
- ls : liste un répertoire, récursif ou non ;
- cat : lit un fichier contenant les arborescences ou fichiers à sauvegarder.



ls ne peut pas être utilisé avec -l (détails) ou -R (récursif).

Il faut une liste simple de noms.

### 6.3.1. Créer une sauvegarde

#### Syntaxe de la commande cpio.

```
[cde de fichiers |] cpio {-o| --create} [-options] [<fic-liste>
[>support]
```

Exemple :

```
[root]# find /etc | cpio -ov > /sauvegardes/etc.cpio
```

Le résultat de la commande **find** est envoyé en entrée de la commande **cpio** par l'intermédiaire du signe “!” (**AltGr+6**). Ici, la commande find /etc renvoie une liste de fichiers correspondant au contenu du répertoire /etc (en récursif) à la commande cpio qui en effectue la sauvegarde. Ne surtout pas oublier le signe > lors de la sauvegarde.

**Tableau 6.3. Options principales de la commande cpio**

Options	Description
-o	Crée une sauvegarde (output).
-v	Affiche le nom des fichiers traités.
-F	Désigne la sauvegarde à modifier (support).

Sauvegarde vers un support :

```
[root]# find /etc | cpio -ov > /dev/rmt0
```

Le support peut être de plusieurs types :

- /dev/rmt0 : lecteur de bande ;
- /dev/sda5 : une partition.

### 6.3.2. Type de sauvegarde

- Sauvegarde avec chemin relatif

```
[root]# cd /
[root]# find etc | cpio -o > /sauvegardes/etc.cpio
```

- Sauvegarde avec chemin absolu

```
[root]# find /etc | cpio -o > /sauvegardes/etc.A.cpio
```



Si le chemin indiqué au niveau de la commande “find” est en **absolu** alors la sauvegarde sera réalisée en absolu.

Si le chemin indiqué au niveau de la commande “find” est en **relatif** alors la sauvegarde sera réalisée en relatif.

### 6.3.3. Ajouter à une sauvegarde

**Syntaxe de la commande cpio pour ajouter un contenu.**

```
[cde de fichiers |] cpio {-o| --create} -A [-options] [<fic-liste>] {F|>support}
```

Exemple :

```
[root]# find /etc/shadow | cpio -o -AF FicSyst.A.cpio
```

L’ajout de fichiers n’est possible que sur un support à accès direct.

Option	Description
-A	Ajoute un ou plusieurs fichiers à une sauvegarde sur disque.
-F	Désigne la sauvegarde à modifier.

### 6.3.4. Compresser une sauvegarde

- Sauvegarder **puis** compresser

```
[root]# find /etc | cpio -o > etc.A.cpio
[root]# gzip /sauvegardes/etc.A.cpio
[root]# ls /sauvegardes/etc.A.cpio*
/sauvegardes/etc.A.cpio.gz
```

- Sauvegarder **et** compresser

```
[root]# find /etc | cpio -o | gzip > /sauvegardes/etc.A.cpio.gz
```

Il n’y a aucune option, comme pour la commande tar, permettant de sauvegarder et de compresser en même temps. Cela s’effectue donc en deux temps : la sauvegarde puis la compression.

La syntaxe de la première méthode est plus facile à comprendre et à retenir, car elle s'effectue en deux temps.

Pour la première méthode, le fichier de sauvegarde est automatiquement renommé par l'utilitaire gzip qui rajoute .gz à la fin du nom de ce fichier. De même l'utilitaire bzip2 rajoute automatiquement .bz2.

### **6.3.5. Lire le contenu d'une sauvegarde**

**Syntaxe de la commande cpio pour lire le contenu d'une sauvegarde cpio.**

```
cpio -t [-options] [<fic-liste>]
```

Exemple :

```
[root]# cpio -tv </sauvegardes/etc.152.cpio | less
```

Options	Description
-t	Lit une sauvegarde.
-v	Affiche les attributs des fichiers.

Après avoir réalisé une sauvegarde, il faut lire son contenu pour être certain qu'il n'y a pas eu d'erreur.

De la même façon, avant d'effectuer une restauration, il faut lire le contenu de la sauvegarde qui va être utilisée.

### **6.3.6. Restaurer une sauvegarde**

**Syntaxe de la commande cpio pour restaurer une sauvegarde.**

```
cpio {-i| --extract} [-E fichier] [-options] [<support>]
```

Exemple :

```
[root]# cpio -iv </sauvegardes/etc.152.cpio | less
```

Options	Description
-i	Restauration complète d'une sauvegarde .

Options	Description
-E fichier	Restaure uniquement les fichiers dont le nom est contenu dans fichier.
-d	Reconstruit l'arborescence manquante.
-u	Remplace tous les fichiers même s'ils existent.
--no-absolute-filenames	Permet de restaurer une archive effectuée en mode absolu de manière relative.



Par défaut, au moment de la restauration, les fichiers sur le disque dont la date de dernière modification est plus récente ou égale à la date de la sauvegarde ne sont pas restaurés (afin d'éviter d'écraser des informations récentes par des informations plus anciennes).

L'option -u permet au contraire de restaurer d'anciennes versions des fichiers.

#### Exemples :

- Restauration en absolu d'une sauvegarde absolue :

```
[root]# cpio -iv <home.A.cpio
```

- Restauration en absolu sur une arborescence existante :

```
[root]# cpio -iuv <home.A.cpio
```

L'option "u" permet d'écraser des fichiers existants à l'endroit où s'effectue la restauration. \* Restauration en relatif d'une sauvegarde absolue :

```
[root]# cpio -iv --no-absolute-filenames <home.A.cpio
```

L'option longue "--no-absolute-filenames" permet une restauration en mode relatif. En effet le "/" en début de chemin est enlevé.

- Restauration en relatif d'une sauvegarde relative :

```
[root]# cpio -iv <etc.cpio
```

- Restauration en absolu du fichier « passwd » :

```
echo "/etc/passwd" > tmp;cpio -iuE tmp <etc.A.cpio; rm -f tmp
```

## 6.4. Utilitaires de compression - décompression

Le fait d'utiliser la compression au moment d'une sauvegarde peut présenter un certain nombre d'inconvénients :

- Allonge le temps de la sauvegarde ainsi que celui de la restauration.
- Rend impossible l'ajout de fichiers à cette sauvegarde.



Il vaut donc mieux effectuer une sauvegarde et la compresser qu'effectuer la compression lors de la sauvegarde.

### 6.4.1. Compresser avec gzip

**Syntaxe de la commande gzip.**

```
gzip [options] [fichier ...]
```

Exemple :

```
[root]# gzip usr.tar  
[root]# ls  
usr.tar.gz
```

Le fichier reçoit l'extension .gz.

Il conserve les mêmes droits et les mêmes dates de dernier accès et de modification.

### 6.4.2. Compresser avec bunzip2

**Syntaxe de la commande bzip2.**

```
bzip2 [options] [fichier ...]
```

Exemple :

```
[root]# bzip2 usr.cpio  
[root]# ls
```

```
usr.cpio.bz2
```

Le nom du fichier reçoit l'extension .bz2.

La compression par “bzip2” est meilleure que celle par “gzip” mais dure plus longtemps.

#### **6.4.3. Décompresser avec gunzip**

**Syntaxe de la commande gunzip.**

```
gunzip [options] [fichier ...]
```

Exemple :

```
[root]# gunzip usr.tar.gz
[root]# ls
usr.tar
```

Le nom du fichier est tronqué par gunzip et se voit enlever l'extension .gz .

Gunzip décomprime également les fichiers portant les extensions suivantes :

- .Z ;
- -Z ;
- \_Z.

#### **6.4.4. Décompresser avec bunzip2**

**Syntaxe de la commande bzip2.**

```
bzip2 [options] [fichier ...]
```

Exemple :

```
[root]# bunzip2 usr.cpio.bz2
[root]# ls
usr.cpio
```

Le nom du fichier est tronqué par bunzip2 et se voit enlever l'extension .bz2 .

bunzip2 décomprime également le fichier portant les extensions suivantes :

- -bz ;
- .tbz2 ;
- tbz.

# Démarrage du système

## 7.1. Démarrage de l'ordinateur

### 7.1.1. Séquence de démarrage

La séquence de démarrage du système est variable en fonction du système mais peut d'une manière générale être découpée selon les étapes définies ci-dessous.

- démarrage de l'ordinateur ou amorçage ;
- exécution du chargeur de démarrage ;
- démarrage du noyau ;
- lancement du processus **init** ;
- lancement des scripts de démarrage.

### 7.1.2. Amorçage matériel

Après la mise sous tension, un programme stocké en mémoire morte sur la carte mère prend le contrôle. Sur les PC, ce programme est appelé le BIOS ou UEFI pour les dernières générations de carte mère. Pour la suite du cours nous ne verrons que le BIOS.

- **BIOS** : Basic Input Output System ;
- **UEFI** : Unified Extensible Firmware Interface ;
- **POST** : Power On Self Test.

**Séquence d'amorçage matériel :**

- mise sous tension de l'ordinateur ;
- lecture **BIOS/UEFI** stocké en mémoire morte ;
- BIOS effectue un autotest : **POST** ;
- lecture des paramètres (périphériques de boot, ...) ;
- lancement du chargeur de démarrage trouvé.

### 7.1.3. Le chargeur de démarrage

Le chargeur de démarrage **localise le noyau du système d'exploitation** sur le disque, le **charge et l'exécute**.

La majorité des chargeurs sont interactifs, ils permettent :

- de **spécifier un noyau** ;
- de positionner des **paramètres optionnels**.



Spécification d'un noyau alternatif : un noyau de sauvegarde dans le cas où la dernière version compilée ne fonctionne pas.

### 7.1.4. Types de chargeurs

On peut retrouver généralement sous Linux :

- **Grub** : GRand Unified Bootloader (le plus répandu) ;
- **Lilo** : LIinux LOader (désaffecté par les développeurs) ;
- **Eliilo** : pour UEFI.



Pour la suite du cours nous utiliserons GRUB.

## 7.2. Le chargeur GRUB

Le chargeur GRUB est un programme de **multiboot** permettant de choisir entre plusieurs systèmes d'exploitation lors du démarrage.

La technique du **chain-loading** lui permet de démarrer une grande variété de systèmes d'exploitation. Il peut donc aussi bien charger des systèmes compatibles avec le multiboot que des systèmes non compatibles.

**La configuration du GRUB est lue au démarrage du système.**



Le **Chain-loading** est une technique qui permet à un chargeur de lancer un autre chargeur en l'encapsulant.

GRUB reconnaît nativement divers systèmes de fichiers.

Il fournit un interpréteur de commandes permettant le chargement manuel d'un système d'exploitation et le changement de la configuration au boot.

GRUB permet d'agir en interactif sur le démarrage.

GRUB peut être utilisé avec différentes interfaces. Beaucoup de distributions GNU/Linux utilisent le support graphique de GRUB pour afficher au démarrage de l'ordinateur un menu avec une image de fond et parfois un support de la souris.

GRUB peut télécharger des images de systèmes d'exploitations depuis un réseau et supporte donc les ordinateurs sans disques. Il peut donc décompresser ces images pour les charger ensuite.

### **7.2.1. Choix du système**

GRUB fournit une interface avec menu à partir de laquelle vous pouvez choisir un système d'exploitation qui sera ensuite amorcé.

Ce menu est basé sur le fichier de configuration **grub.conf** qui se trouve dans le répertoire **/boot/grub/**.

En fonction des distributions, **/etc/grub.conf** peut être un lien symbolique vers ce fichier.



Lors des sélections dans le GRUB, le clavier est en **qwerty**.

**Tableau 7.1. Les touches de sélection du menu de Grub**

Touche	Action
[ENTREE]	Démarre le système sélectionné.
[e]	Édite la configuration du système.
[a]	Permet de modifier les arguments.
[c]	Permet d'utiliser l'interface Shell de GRUB.

La touche **[ENTREE]** permet de lancer l'initialisation du serveur.

Une fois le démarrage terminé, le système affiche les messages placés dans le fichier **/etc/issue** et propose de saisir un login puis un mot de passe pour se connecter.

La touche **[e]** permet d'éditer la configuration avant de démarrer. Il est alors possible de modifier les lignes en les éditant une à une avec la touche **[e]**.

Après avoir édité la ligne sélectionnée, procéder à la modification puis valider la ligne modifiée en utilisant la touche **[ENTREE]**.

Pour initialiser le système en tenant compte de ces modifications, utiliser la touche **[b]** (comme 'boot').

La touche **[a]** / **[q]** permet de modifier les arguments du noyau.

La touche **[c]** permet d' obtenir l'interface Shell de Grub.

### 7.2.2. Fichier /boot/grub/grub.conf



```

default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
Hiddenmenu
title CentOS (2.6.32-220.el6.i686)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-220.el6.i686 ro
        root=LABEL=/ rhgb quiet
    initrd  /initramfs-2.6.32-220.el6.i686.img

```

La première partie est constituée de lignes de commentaires décrivant la structure du fichier.

Aucune compilation de ce fichier de configuration ne sera nécessaire.

### Explications des options :

Tableau 7.2. Variables du fichier /boot/grub/grub.conf

Variable	Observation
default=0	Correspond au système d'exploitation lancé par défaut. La première rubrique <b>title</b> porte le numéro 0.
timeout=5	GRUB amorcera automatiquement le système par défaut au bout de 5 secondes, à moins d'être interrompu.
splashimage	Déclaration de l'image qui s'affiche avec le chargeur Grub. Il faut indiquer l'emplacement de cette image.  Les systèmes de fichiers n'étant pas encore montés, indiquer le disque et la partition de /boot ( <b>hd0,0</b> ), le chemin jusqu'à l'image <b>grub/</b> et enfin le nom de votre image <b>splash.xpm.gz</b> .
hiddenmenu	Sert à masquer le menu Grub et après le délai du timeout, le système se lancera automatiquement en fonction de l'option <b>default</b> .
title	Il s'agit en fait du nom qui apparaîtra dans le menu Grub (exemple "ma distrib Linux préférée"). En règle générale, le nom du système est choisi : exemple Fedora, Suse, Ubuntu, Vista, Xp, Frugal, etc,... et éventuellement la version du noyau.  Un seul nom par rubrique <b>title</b> , il faut donc déclarer autant de lignes 'title' qu'il y a d'options de démarrage ou de systèmes installés.
root	Indique le disque puis la partition (hdx,y) où se trouvent les fichiers permettant l'initialisation du système (exemple : (hd0,0), correspondant à la partition /boot) pour ce "title".
kernel	Indique le nom du noyau à charger, son emplacement et les options utiles à son démarrage pour ce "title".

Variable	Observation
initrd	initrd (INITial RamDisk) charge un ramdisk initial pour une image de démarrage au format Linux et définit les paramètres adéquats dans la zone de configuration de Linux en mémoire pour ce "title".

## 7.3. Sécuriser GRUB

GRUB est par défaut très permissif. Certaines opérations interactives ne nécessitent pas d'authentification. Ainsi, il est possible d'exécuter des commandes root sans s'être authentifié !



L'accès au menu interactif doit être protégé.

### 7.3.1. Définir un mot de passe

Il faut définir un mot de passe avec la directive **password** dans le fichier **/boot/grub/grub.conf**.

```

default=0
timeout=5
splashimage=(hd0,0)/grub/TuxInBlueAbyss.xpm.gz
Hiddenmenu
password mot_de_passe_en_clair

```

La directive **password** est ajoutée dans la partie globale avant la directive **title**.

Quand la directive **password** est saisie dans **/boot/grub/grub.conf**, GRUB interdit tout contrôle interactif, jusqu'à ce que la touche **[p]** soit pressée et qu'un mot de passe correct soit entré.

### 7.3.2. Chiffrer le mot de passe

Le mot de passe peut être chiffré.

À l'aide d'un shell traditionnel :

```
[root]# grub-crypt >> /boot/grub/grub.conf
```

Dans le fichier **grub.conf** l'option **[--encrypted]** devra être ajoutée entre la directive **password** et le mot de passe chiffré.

Vous pouvez chiffrer votre mot de passe avec la commande **grub-crypt**.

Copiez le mot de passe chiffré dans votre fichier de configuration et indiquez dans la rubrique qu'il est chiffré.

```
[root]# grub-crypt
Password : mdpauser1
Retype password : mdpauser1
$6$uK6Bc/$90LR/0QW.14G4473EaEND
```



Le hash du mot de passe fourni par la commande grub-crypt commence par un \$6\$ car l'algorythme utilisé est le SHA-512.

**Tableau 7.3. Liste des algorythme de hashage des mots de passe**

Préfixe	Algorythme utilisé
	DES
\$1\$	MD5
\$2\$, \$2a\$, \$2x\$, \$2y\$	bcrypt
\$3\$	NTHASH
\$5\$	SHA-256
\$6\$	SHA-512

Dans le but de récupérer ce mot de passe et de l'insérer directement dans le fichier **grub.conf**, utilisez la commande suivante :

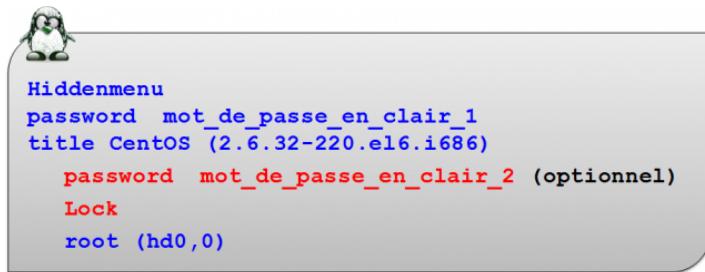
```
[root]# grub-crypt >>/boot/grub/grub.conf
```

### 7.3.3. Menu interactif verrouillé

Pour agir sur l'interactivité du démarrage, utiliser la touche **[p]** pour saisir le mot de passe et ensuite disposer des options permettant d'agir sur le lancement du noyau.

### 7.3.4. Lancement verrouillé

Le lancement d'un système peut être verrouillé avec la directive **lock** positionnée en dessous de la ligne **title** à verrouiller.



- Le mot de passe sera systématiquement demandé.
- Pourquoi verrouiller le lancement d'un système ? L'administrateur peut prévoir une rubrique qui lancera son système en mode **single**. Aucun utilisateur ne devra donc utiliser cette rubrique.
- Il est possible d'affecter un mot de passe différent pour chaque menu. Pour cela il suffira de remettre une directive **password** avec un nouveau mot de passe dans chaque rubrique **title**.

## 7.4. Démarrage du noyau

Au démarrage du système, GRUB apparaît.

**[ENTREE]** active la configuration par défaut.

Une autre touche fait apparaître le menu du GRUB.

### 7.4.1. Niveaux de démarrage

Tableau 7.4. Les 6 niveaux de démarrage

s ou single	Le processus <b>init</b> démarre le système en mode mono-utilisateur. Par défaut l'utilisateur est connecté en tant que <b>root</b> sans fournir de mot de passe.
1 - 5	Le processus <b>init</b> démarre le système avec le niveau demandé.

### 7.4.2. Étapes du démarrage

Principales étapes du démarrage :

- chargement du noyau (processus 0) ;
- installation des périphériques via leur pilote ;
- démarrage du gestionnaire de swap ;
- montage du système de fichiers racine ;
- création par le noyau du premier processus qui porte le numéro 1 ;
- ce processus exécute le programme **/sbin/init** en lui passant les paramètres qui ne sont pas déjà gérés par le noyau.

## 7.5. Le processus init (généralités)

### 7.5.1. Les différents niveaux d'exécution

Tableau 7.5. Les 8 niveaux d'exécution (détailés)

0	Arrête le système.
1	Mode mono-utilisateur (console).
2	Mode multi-utilisateurs. Les systèmes de fichiers sont montés. Le service réseau est démarré.
3	Sur-ensemble du niveau 2. Il est associé au démarrage des services de partage à distance.
4	Mode multi-utilisateurs spécifique au site informatique.
5	Sur-ensemble du niveau 3. Interface X-Window (graphique).

6	Redémarre le système.
s, S, single	Mode mono-utilisateur (single). Les systèmes de fichiers sont montés. Seuls les processus fondamentaux pour le bon fonctionnement du système sont activés. Un shell en mode <b>root</b> est activé sur une console. Le répertoire <b>/etc</b> n'est pas indispensable.

Il n'y a qu'un niveau d'exécution actif à la fois.

### 7.5.2. La commande init

La commande init permet de changer le niveau d'exécution courant .

#### Syntaxe de la commande init.

```
init [-options] [0123456Ss]
```

Exemple :

```
[root]# init 5
```

### 7.5.3. La commande runlevel

La commande runlevel permet de connaître le niveau d'exécution courant.

#### Syntaxe de la commande runlevel.

```
runlevel
```

Exemple :

```
[root]# runlevel
N 3
```

Dans cet exemple, le système se trouve au niveau d'exécution 3 - Multi-users.

Le N indique que le niveau d'exécution précédent était le démarrage du système.  
Après un init 5, le résultat de la commande serait alors :

```
[root]# runlevel
```

3 5

Il n'y a qu'un niveau d'exécution actif à la fois.

#### 7.5.4. Le fichier /etc/inittab

Lors du démarrage, à la création du processus **init**, le niveau est celui défini dans **GRUB** ou sinon celui dans **/etc/inittab**.

Le niveau défini dans **GRUB** est prioritaire à celui défini dans **inittab**.

**Aperçu du fichier /etc/inittab.**

```
# For information on how to write upstart event handlers, or how
# upstart works, see init(5), init(8), and initctl(8).
# Default runlevel. The runlevels used are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have
#      networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
id:5:initdefault:
```

#### 7.5.5. Changement de niveau

Lorsqu'un changement de niveau est effectué, le processus **init** envoie un signal **SIGTERM (15)** à tous les processus non concernés par le nouveau niveau.

Un délai de 5 secondes est accordé afin que les processus se terminent correctement. Après ce délai, le processus **init** envoie un deuxième signal **SIGKILL (9)** à tous les processus non terminés.

**init** démarre ensuite les processus concernés par le nouveau niveau d'exécution.

#### 7.5.6. Activer ou désactiver les terminaux

Pour activer ou désactiver les terminaux, il faut modifier la variable **ACTIVE\_CONSOLES** dans le fichier **/etc/sysconfig/init**.

Exemples :

```
ACTIVE_CONSOLES="/dev/tty[1-6]" #Active les terminaux de 1 à 6  
ACTIVE_CONSOLES="/dev/tty[1-6] /dev/tty8 /dev/tty9" #Active les  
terminaux de 1 à 6, le 8 et le 9
```

Ne pas oublier les “ “.

Le système doit être redémarré pour la prise en compte.



Au niveau de démarrage 5, le système ne prend pas en compte le fichier `/etc/sysconfig/init`.

En cas d'erreur de manipulation dans ce fichier, un moyen de redémarrer un serveur est de force le redémarrage en init 5 !

### 7.5.7. Autoriser à root l'accès aux terminaux

Par défaut, une console déclarée dans `/etc/sysconfig/init` n'est accessible que par les utilisateurs.

Il faut renseigner le fichier `/etc/securetty` en ajoutant le nom de ce nouveau terminal pour autoriser root à s'y connecter.

Exemple suivant, à la dernière ligne (tty6) le `#` est retiré permettant l'accès à root sur ce terminal.

```
console  
#vc/1  
#vc/2  
#vc/3  
#vc/4  
#vc/5  
#vc/6  
#tty1  
#tty2  
#tty3  
#tty4
```

```
#tty5  
tty6
```

## 7.6. Le processus init (démon)

### 7.6.1. Démarrage des démons

Init lance le script **/etc/rc.d/rc.sysinit** quelque soit le niveau.

Init exécute ensuite le script **/etc/rc.d/rc** en lui passant en paramètre le niveau d'exécution demandé

### 7.6.2. Script de démarrage des services

Pour chaque service, il y a un script de démarrage stocké dans **/etc/rc.d/init.d**.

Chaque script accepte au minimum en argument :

- stop : pour arrêter le service ;
- start : pour démarrer le service ;
- restart : pour redémarrer le service ;
- status : pour connaître l'état du service.

### 7.6.3. Répertoires d'ordonnancement

Pour chaque niveau d'exécution, il existe un répertoire correspondant : **/etc/rc.d/rc[0-6].d/**.

Ces répertoires contiennent les liens symboliques vers les scripts placés dans **/etc/rc.d/init.d**.

L'avantage du lien : il n'existe qu'un seul exemplaire du script du service.

### 7.6.4. Nom des liens

Mise en route (**S**tart) : **SXXnom**

Arrêt (**K**ill) : **KYYnom**

**XX et YY** : nombre de 00 à 99 qui guide l'ordre d'exécution (Start ou Kill).

**nom** : nom exact du service à démarrer ou à arrêter tel qu'écrit dans /etc/rc.d/init.d/.

La somme des nombres est un complément à 100 : XX + YY = 100.

Cette méthode permet d'ordonnancer le démarrage et l'arrêt des services. Un service qui est démarré en premier doit être le dernier à s'arrêter. La liste étant lue dans l'ordre alphabétique.

Exemple :

Dans /etc/rc.d/rc3.d/, nous avons :

- K15httpd
- S10network
- S26acpid

Donc, pour le niveau d'exécution 3 (rc3.d) :

- le service httpd doit être arrêté (lettre K),
- les services network et acpid doivent être lancés (lettre S) dans cet ordre (numéro du service network 10 plus petit que celui de acpid 26)

### **7.6.5. Le programme /etc/rc.d/rc**

Ce programme est lancé par **init** avec le niveau d'exécution en paramètre.

**Il comporte deux boucles.**

**Init** lance le script **rc** avec le niveau **X** en paramètre.

Première boucle : lecture des scripts d'arrêt **K...** présents dans **rcX.d**.

Deuxième boucle : lecture des scripts de démarrage **S...** présents dans **rcX.d**.

### 7.6.6. Architecture de démarrage

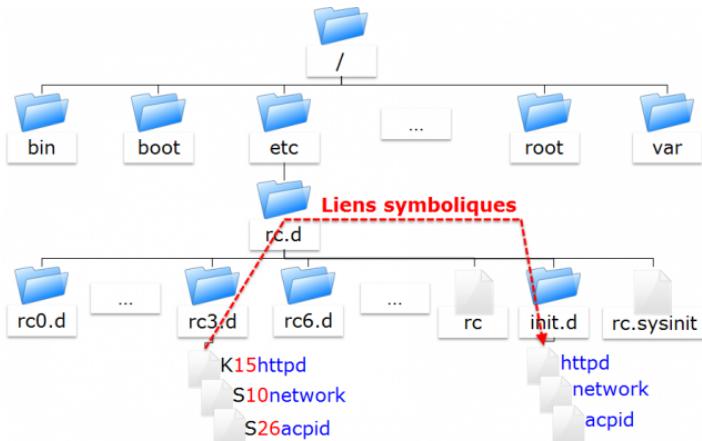


Figure 7.1. Synthèse de l'arborescence système liée au démarrage

## 7.7. La gestion des services

Comme il n'existe qu'un seul exemplaire du fichier script par service sous **/etc/rc.d/init.d**, leur gestion est facilitée. La gestion des liens s'effectue soit :

- manuellement avec la commande `ln` ;
- avec la commande de gestion des services `chkconfig`.

### 7.7.1. La commande `ln`

Créer un lien symbolique manuellement.

#### Syntaxe de la commande `ln`.

```
ln -s source destination
```

Exemple :

```
[root]# cd /etc/rc.d/rc2.d
[root]# ln -s ../init.d/numlock S85numlock
```

Il faut alors créer tous les liens (**K** ou **S**) pour **chaque niveau de démarrage**.

### 7.7.2. La commande **chkconfig**

La commande chkconfig permet de gérer un service.

Il faut les deux lignes suivantes au début de chaque script.

```
# chkconfig: [niveau_exécution] [num_start] [num_kill]
# description: [descriptif du script]
```

Exemple 1 :

```
# chkconfig: 2345 10 90
# description: Commentaires libres
```

Exemple 2 :

```
# chkconfig: -
# description: Commentaires libres
```

Le - après **chkconfig:** signifie que le service ne doit jamais être démarré.

### Gérer et visualiser l'état d'un service

#### Syntaxe de la commande **chkconfig**.

```
chkconfig [--options] [service]
chkconfig --level service on|off|reset
```

Exemple :

```
[root]# chkconfig --list network
network 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt
```

Tableau 7.6. Options de la commande **chkconfig**

Option longue	Description
--list	Visualise l'état des services (--list seul liste tous les services créés).
--add	Crée des liens symboliques.

Option longue	Description
--del	Supprime des liens symboliques.
--level	Modifie les liens symboliques.

La commande **chkconfig --list** lit les niveaux définis dans l'en-tête du service et affiche la configuration de démarrage du service.

Cette commande ne donne pas un état actuel du service.

**Arrêt** ne signifie pas que le service est arrêté, mais qu'il ne sera pas démarré au niveau spécifié.

Un autre moyen de visualiser les liens symboliques en une seule commande :

```
[root]# ls -l /etc/rc.d/rc*.d/*
```

### **Créer les liens symboliques**

```
chkconfig --add service
```

Exemple :

```
[root]# chkconfig --add network
[root]# chkconfig --list network
network 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt
```

chkconfig --add lit les niveaux définis dans l'en-tête du service et crée les liens correspondants.

Exemple :

#### **Fichier /etc/rc.d/init.d/nomduService.**

```
# chkconfig: 235 90 10
```

chkconfig crée les liens **S90...** dans les répertoires définis /etc/rc.d/rc2.d, rc3.d et rc5.d et les liens **K10...** dans les répertoires restants /etc/rc.d/rc0.d, rc1.d, rc4.d et rc6.d

Afin d'éviter les incohérences, faire un **chkconfig --del nomduservice** avant le **chkconfig --add**.

## ***Supprimer des liens symboliques***

```
chkconfig --del nomduservice
```

Exemple :

```
[root]# chkconfig --del network
[root]# chkconfig --list network
network 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt
```

## ***Modifier des liens symboliques***

### **Syntaxe de la commande chkconfig.**

```
chkconfig [--level niveaux] service <on | off>
```

Exemple :

```
[root]# chkconfig --level 235 atd on
[root]# chkconfig --level 0146 atd off
```

--level	Spécification du niveau auquel est créé le lien symbolique.
on	Le lien permettant de lancer le service est créé (Sxx...).
off	Le lien permettant d'arrêter le service est créé (Kxx...).

## ***Démarrage d'un service***

Manuellement :

- Avec un **script** de lancement :

```
/chemin/script [status|start|stop|restart]
```

Exemple :

```
[root]#/etc/rc.d/init.d/crond start
Démarrage de crond : [ OK ]
[root]#/etc/rc.d/init.d/crond status
crond (pid 4731) en cours d'exécution
```

- Avec la commande **service** :

#### Syntaxe de la commande **service**.

```
service script [status|start|stop|restart]
```

Exemple :

```
[root]# service crond start
Démarrage de crond : [ OK ]
[root]# service crond status
crond (pid 4731) en cours d'exécution
```

La commande **service** prend en compte tous les scripts placés dans **/etc/rc.d/init.d**.



La commande **service** n'existe que dans le monde des distributions RHEL !

## 7.8. Arrêt du système

Les opérations de maintenance, diagnostics, modifications de logiciels, ajouts et retraits de matériel, tâches administratives, coupures électriques, ... nécessitent parfois l'arrêt du système.

Cet arrêt peut être planifié, périodique ou impromptu et demander une réactivité immédiate.

Tous les systèmes Unix, y compris ceux fonctionnant sur PC doivent être mis hors service en utilisant les commandes décrites dans cette section.

Ceci garantit l'**intégrité** du disque et la **terminaison propre** des différents services du système.

**Arrêt programmé du système :**

- utilisateurs prévenus de l'arrêt ;
- applications arrêtées proprement ;
- intégrité des systèmes de fichiers assurée ;
- sessions utilisateurs stoppées.

En fonction des options, le système :

- passe en mode mono-utilisateur ;
- est arrêté ;
- est redémarré.

Commandes de mise hors service :

- init ;
- shutdown ;
- halt ;
- reboot.

**7.8.1. Commande shutdown****Syntaxe de la commande shutdown.**

```
shutdown [-t sec] [options] heure [message-avertissement]
```

Exemples :

```
[root]# shutdown -r +2 "arrêt puis reboot dans 2 minutes"
[root]# shutdown -r 10:30 "arrêt puis reboot à 10h30"
[root]# shutdown -h now "arrêt électrique"
```

**Tableau 7.7. Options de la commande shutdown**

Options	Commentaires
-t sec	Attendre <b>sec</b> entre le message d'avertissement et le signal de fin aux processus

-r	Redémarrer la machine après l'arrêt du système
-h	Arrêter la machine après l'arrêt du système
-P	Éteindre l'alimentation
-f	Ne pas effectuer de fsck en cas de redémarrage
-F	Forcer l'utilisation de fsck en cas de redémarrage
-c	Annuler un redémarrage en cours

**heure** : Quand effectuer le shutdown (soit une heure fixe **hh:mm**, soit un délai d'attente en minute **+mm**).

**message-avertissement** : Message à envoyer à tous les utilisateurs.

### 7.8.2. Commande halt

Arrêt immédiat

```
[root]# halt
```

Cette commande appelle le processus **init**

**halt** ⇒ **init 0**

### 7.8.3. Commande reboot

Redémarrage immédiat

```
[root]# reboot
```

Cette commande appelle le processus **init**

**reboot** ⇒ **init 6**



---

# 8

## Démarrage du système sous CentOS

7

---

### 8.1. Le processus de démarrage

Il est important de comprendre le processus de démarrage de Linux pour pouvoir résoudre les problèmes qui peuvent y survenir.

Le processus de démarrage comprend :

#### 8.1.1. Le démarrage du BIOS

Le **BIOS** (Basic Input/Output System) effectue le test **POST** (power on self test) pour détecter, tester et initialiser les composants matériels du système.

Il charge ensuite le **MBR** (Master Boot Record).

#### 8.1.2. Le Master boot record (MBR)

Le Master Boot Record correspond aux 512 premiers bytes du disque de démarrage. Le MBR découvre le périphérique de démarrage et charge le chargeur de démarrage **GRUB2** en mémoire et lui transfert le contrôle.

Les 64 bytes suivants contiennent la table de partition du disque.

#### 8.1.3. Le chargeur de démarrage GRUB2 (Bootloader)

Le chargeur de démarrage par défaut de la distribution CentOS 7 est **GRUB2** (GRand Unified Bootloader). GRUB2 remplace l'ancien chargeur de démarrage Grub (appelé également GRUB legacy).

Le fichier de configuration de GRUB 2 se situe sous **/boot/grub2/grub.cfg** mais ce fichier ne doit pas être directement édité.

Les paramètres de configuration du menu de GRUB2 se trouvent sous **/etc/default/grub** et servent à la génération du fichier grub.cfg.

#### Exemple de fichier /etc/default/grub file.

```
# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/
root rhgb quiet net.ifnames=0"
GRUB_DISABLE_RECOVERY="true"
```

Si des changements sont effectués à un ou plusieurs de ces paramètres, il faut lancer la commande **grub2-mkconfig** pour regénérer le fichier **/boot/grub2/grub.cfg**.

```
[root] # grub2-mkconfig -o /boot/grub2/grub.cfg
```

- GRUB2 cherche l'image du noyau compressé (le fichier vmlinuz) dans le répertoire /boot.
- GRUB2 charge l'image du noyau en mémoire et extrait le contenu du fichier image initramfs dans un dossier temporaire en mémoire en utilisant le système de fichier tmpfs.

#### 8.1.4. Le noyau

Le noyau démarre le processus **systemd** avec le PID 1.

```
root      1      0  0 02:10 ?          00:00:02 /usr/lib/systemd/
systemd --switched-root --system --deserialize 23
```

#### 8.1.5. **systemd**

Systemd est le père de tous les processus du système. Il lit la cible du lien **/etc/systemd/system/default.target** (par exemple **/usr/lib/systemd/system/multi-**

user.target) pour déterminer la cible par défaut du système. Le fichier définit les services à démarrer.

Systemd positionne ensuite le système dans l'état défini par la cible en effectuant les tâches d'initialisations suivantes :

1. Paramétriser le nom de machine
2. Initialiser le réseau
3. Initialiser SELinux
4. Afficher la bannière de bienvenue
5. Initialiser le matériel en se basant sur les arguments fournis au kernel lors du démarrage
6. Monter les systèmes de fichiers, en incluant les systèmes de fichiers virtuels comme /proc
7. Nettoyer les répertoires dans /var
8. Démarrer la mémoire virtuelle (swap)

## 8.2. Protéger le chargeur de démarrage GRUB2

Pourquoi protéger le chargeur de démarrage avec un mot de passe ?

1. Prévenir les accès au mode utilisateur **Single** – Si un attaquant peut démarrer en mode single user, il devient l'utilisateur root.
2. Prévenir les accès à la console GRUB – Si un attaquant parvient à utiliser la console GRUB, il peut changer sa configuration ou collecter des informations sur le système en utilisant la commande cat.
3. Prévenir les accès à des systèmes d'exploitation non sécurisés. S'il y a un double boot sur le système, un attaquant peut sélectionner au démarrage un système d'exploitation comme DOS qui ignore les contrôles d'accès et les permissions des fichiers.

Pour protéger par mot de passe le GRUB2 :

- Retirer **-unrestricted** depuis la déclaration principale **CLASS=** dans le fichier **/etc/grub.d/10\_linux**.

- Si un utilisateur n'a pas encore été configuré, utiliser la commande **grub2-setpassword** pour fournir un mot de passe à l'utilisateur root :

```
# grub2-setpassword
```

Un fichier **/boot/grub2/user.cfg** va être créé s'il n'était pas encore présent. Il contient le mot de passe hashé du GRUB.



Cette commande ne supporte que les configurations avec un seul utilisateur root.

### Exemple de fichier **/boot/grub2/user.cfg**.

```
[root]# cat /boot/grub2/user.cfg
GRUB2_PASSWORD=grub.pbkdf2.sha512.10000.CC6F56....A21
```

- Recréer le fichier de configuration avec la commande **grub2-mkconfig** :

```
[root]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-327.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-
f9725b0c842348ce9e0bc81968cf7181
Found initrd image: /boot/initramfs-0-rescue-
f9725b0c842348ce9e0bc81968cf7181.img
done
```

- Redémarrer le serveur et vérifier.

Toutes les entrées définies dans le menu du GRUB vont maintenant nécessiter la saisie d'un utilisateur et d'un mot de passe à chaque démarrage. Le système ne démarrera pas de noyau sans intervention directe de l'utilisateur depuis la console.

- Lorsque l'utilisateur est demandé, saisir "root" ;
- Lorsqu'un mot de passe est demandé, saisir le mot de passe fourni à la commande grub2-setpassword.

Pour ne protéger que l'édition des entrées du menu de GRUB et l'accès à la console, l'exécution de la commande grub2-setpassword est suffisante.

### 8.3. Systemd

**Systemd** est un gestionnaire de service pour les systèmes d'exploitation Linux.

Il est développé pour :

- rester compatible avec les anciens scripts d'initialisation SysV,
- fournir de nombreuses fonctionnalités comme le démarrage en parallèle des services système au démarrage du système, l'activation à la demande de démons, le support des instantanés ou la gestion des dépendances entre les services.



Systemd est le système d'initialisation par défaut depuis la RedHat/CentOS 7.

Systemd introduit le concept d'unités systemd.

Tableau 8.1. Principaux types d'unités systemd disponibles

Type	Extension du fichier	Observation
Unité de service	.service	Service système
Unité cible	.target	Un groupe d'unités systemd
Unité automount	.automount	Un point de montage automatique pour système de fichiers



Il existe de nombreux types d'unités : Device unit, Mount unit, Path unit, Scope unit, Slice unit, Snapshot unit, Socket unit, Swap unit, Timer unit.

- Systemd supporte les instantanés de l'état du système et leur restauration.
- Les points de montage peuvent être configurés comme des cibles systemd.
- Au démarrage, systemd crée des sockets en écoute pour tous les services système qui supportent ce type d'activation et passe ces sockets à ces

services aussitôt qu'elles sont démarrées. Cela rend possible la relance d'un service sans perdre un seul message qui lui est envoyé par le réseau durant son indisponibilité. La socket correspondante reste accessible et tous les messages sont mis en file d'attente.

- Les services systèmes qui utilisent D-BUS pour leurs communications inter-process peuvent être démarrés à la demande dès la première utilisation par un client.
- Systemd stoppe ou relance uniquement les services en cours de fonctionnement. Les versions précédentes de CentOS tentaient directement de stopper les services sans vérifier leur état en cours.
- Les services systèmes n'héritent d'aucun contexte (comme les variables d'environnements HOME et PATH). Chaque service fonctionne dans son propre contexte d'exécution.

Toutes les opérations des unités de service sont soumises à un timeout par défaut de 5 minutes pour empêcher un service malfonctionnant de geler le système.

### 8.3.1. Gérer les services systèmes

Les unités de service se terminent par l'extension de fichier .service et ont un but similaire à celui des scripts init. La commande systemctl est utilisée pour afficher, lancer, arrêter, redémarrer, activer ou désactiver des services système.



Les commandes service et chkconfig sont toujours disponibles dans le système et fonctionnent comme prévu, mais sont uniquement incluses pour des raisons de compatibilité et doivent être évitées.

Tableau 8.2. Comparaison des utilitaires service et systemctl

service	systemctl	Description
service <i>name</i> start	systemctl start <i>name.service</i>	Lancer un service
service <i>name</i> stop	systemctl stop <i>name.service</i>	Stoppe un service
service <i>name</i> restart	systemctl restart <i>name.service</i>	Relance un service

<b>service</b>	<b>systemctl</b>	<b>Description</b>
service <i>name</i> reload	systemctl reload <i>name.service</i>	Recharge une configuration
service <i>name</i> status	systemctl status <i>name.service</i>	Vérifie si un service fonctionne
service <i>name</i> condrestart	systemctl try-restart <i>name.service</i>	Relance un service seulement s'il fonctionne
service --status-all	systemctl list-units --type service --all	Affiche le status de tous les services

Tableau 8.3. Comparaison des utilitaires chkconfig et systemctl

<b>chkconfig</b>	<b>systemctl</b>	<b>Description</b>
chkconfig <i>name</i> on	systemctl enable <i>name.service</i>	Active un service
chkconfig <i>name</i> off	systemctl disable <i>name.service</i>	Désactive un service
chkconfig --list <i>name</i>	systemctl status <i>name.service</i>	Vérifie si un service fonctionne
chkconfig --list	systemctl list-unit-files --type service	Liste tous les services et vérifie s'ils fonctionnent
chkconfig --list	systemctl list-dependencies --after	Liste les services qui démarrent avant l'unité spécifiée
chkconfig --list	systemctl list-dependencies --before	Liste les services qui démarrent après l'unité spécifiée

Exemples :

```
systemctl stop nfs-server.service
# ou
systemctl stop nfs-server
```

Pour lister toutes les unités chargées actuellement :

```
systemctl list-units --type service
```

Pour lister toutes les unités pour vérifier si elles sont activées :

```
systemctl list-unit-files --type service
```

```
systemctl enable httpd.service  
systemctl disable bluetooth.service
```

### **8.3.2. Exemple de fichier .service pour le service postfix**

```
postfix.service Unit File  
What follows is the content of the /usr/lib/systemd/system/  
postfix.service unit file as currently provided by the postfix package:  
  
[Unit]  
Description=Postfix Mail Transport Agent  
After=syslog.target network.target  
Conflicts=sendmail.service exim.service  
  
[Service]  
Type=forking  
PIDFile=/var/spool/postfix/pid/master.pid  
EnvironmentFile=-/etc/sysconfig/network  
ExecStartPre=-/usr/libexec/postfix/aliasesdb  
ExecStartPre=-/usr/libexec/postfix/chroot-update  
ExecStart=/usr/sbin/postfix start  
ExecReload=/usr/sbin/postfix reload  
ExecStop=/usr/sbin/postfix stop  
  
[Install]  
WantedBy=multi-user.target
```

### **8.3.3. Utiliser les targets systèmes**

Sur CentOS7/RHEL7, le concept des niveaux d'exécution a été remplacé par les cibles Systemd.

Les cibles Systemd sont représentées par des unités de cible (target units). Les unités de cible se terminent par l'extension de fichier .target et leur unique but consiste à regrouper d'autres unités Systemd dans une chaîne de dépendances.

Par exemple, l'unité **graphical.target**, qui est utilisée pour lancer une session graphique, lance des services systèmes comme le gestionnaire d'affichage GNOME (gdm.service) ou le services des comptes (accounts-daemon.service) et active également l'unité multi-user.target.

De manière similaire, l'unité multi-user.target lance d'autres services système essentiels, tels que NetworkManager (NetworkManager.service) ou D-Bus (dbus.service) et active une autre unité cible nommée basic.target.

Tableau 8.4. Comparaison des targets systemctl et runlevel

Runlevel	Target Units	Description
0	poweroff.target	Arrête le système et l'éteint
1	rescue.target	Active un shell de secours
2	multi-user.target	Active un système multi-utilisateur sans interface graphique
3	multi-user.target	Active un système multi-utilisateur sans interface graphique
4	multi-user.target	Active un système multi-utilisateur sans interface graphique
5	graphical.target	Active un système multi-utilisateur avec interface graphique
6	reboot.target	Arrête puis redémarre le système

## La cible par défaut

Pour déterminer quelle cible est utilisée par défaut :

```
systemctl get-default
```

Cette commande recherche la cible du lien symbolique située à /etc/systemd/system/default.target et affiche le résultat.

```
$ systemctl get-default  
graphical.target
```

La commande systemctl peut également fournir la liste des cibles disponibles :

### Lister toutes les cibles disponibles.

```
sudo systemctl list-units --type target  
UNIT           LOAD   ACTIVE SUB     DESCRIPTION  
basic.target    loaded  active  active  Basic System  
bluetooth.target loaded  active  active  Bluetooth  
cryptsetup.target loaded  active  active  Encrypted Volumes  
getty.target    loaded  active  active  Login Prompts  
graphical.target loaded  active  active  Graphical Interface  
local-fs-pre.target loaded  active  active  Local File Systems (Pre)  
local-fs.target  loaded  active  active  Local File Systems  
multi-user.target loaded  active  active  Multi-User System  
network-online.target loaded  active  active  Network is Online  
network.target   loaded  active  active  Network  
nss-user-lookup.target loaded  active  active  User and Group Name Lookups  
paths.target    loaded  active  active  Paths  
remote-fs.target loaded  active  active  Remote File Systems  
slices.target   loaded  active  active  Slices  
sockets.target  loaded  active  active  Sockets  
sound.target    loaded  active  active  Sound Card  
swap.target     loaded  active  active  Swap  
sysinit.target  loaded  active  active  System Initialization  
timers.target   loaded  active  active  Timers
```

Pour configurer le système afin d'utiliser une cible différente par défaut :

```
systemctl set-default name.target
```

Exemple :

```
[root]# systemctl set-default multi-user.target  
rm '/etc/systemd/system/default.target'  
ln -s '/usr/lib/systemd/system/multi-user.target' '/etc/systemd/system/  
default.target'
```

Pour passer à une unité de cible différente dans la session actuelle :

```
systemctl isolate name.target
```

Le **mode de secours** ("Rescue mode") fournit un environnement simple et permet de réparer votre système dans les cas où il est impossible d'effectuer un processus de démarrage normal.

En mode de secours, le système tente de monter tous les systèmes de fichiers locaux et de lancer plusieurs services système importants, mais n'active pas d'interface réseau ou ne permet pas à d'autres utilisateurs de se connecter au système au même moment.

Sur CentOS7/RHEL7, le mode de secours est équivalent au mode utilisateur seul (single user mode) et requiert le mot de passe root.

Pour modifier la cible actuelle et entrer en mode de secours dans la session actuelle :

```
systemctl rescue
```

Le **mode d'urgence** ("Emergency mode") fournit l'environnement le plus minimaliste possible et permet de réparer le système même dans des situations où le système est incapable d'entrer en mode de secours. Dans le mode d'urgence, le système monte le système de fichiers root uniquement en lecture, il ne tentera pas de monter d'autre système de fichiers locaux, n'activera pas d'interface réseau et lancera quelques services essentiels.

Pour modifier la cible actuelle et entrer en mode d'urgence dans la session actuelle :

```
systemctl emergency
```

## **Arrêt, suspension et hibernation**

La commande systemctl remplace un certain nombre de commandes de gestion de l'alimentation utilisées dans des versions précédentes :

Tableau 8.5. Comparaison entre les commandes de gestion de l'alimentation et systemctl

Ancienne commande	Nouvelle commande	Description
halt	systemctl halt	Arrête le système.
poweroff	systemctl poweroff	Met le système hors-tension.
reboot	systemctl reboot	Redémarre le système.
pm-suspend	systemctl suspend	Suspend le système.
pm-hibernate	systemctl hibernate	Met le système en hibernation.
pm-suspend-hybrid	systemctl hybrid-sleep	Met en hibernation et suspend le système.

### **8.3.4. Le processus journald**

Les fichiers journaux peuvent, en plus de rsyslogd, également être gérés par le démon **journald** qui est un composant de systemd.

Le démon journald capture les messages Syslog, les messages du journal du noyau, les messages du disque RAM initial et du début du démarrage, ainsi que les messages inscrits sur la sortie standard et la sortie d'erreur standard de tous les services, puis il les indexe et les rend disponibles à l'utilisateur.

Le format du fichier journal natif, qui est un fichier binaire structuré et indexé, améliore les recherches et permet une opération plus rapide, celui-ci stocke également des informations de métadonnées, comme l'horodatage ou les ID d'utilisateurs.

### **8.3.5. La commande journalctl**

La commande **journalctl** permet d'afficher les fichiers journaux.

```
journalctl
```

La commande liste tous les fichiers journaux générés sur le système. La structure de cette sortie est similaire à celle utilisée dans `/var/log/messages/` mais elle offre quelques améliorations :

- la priorité des entrées est marquée visuellement ;
- les horodatages sont convertis au fuseau horaire local de votre système ;
- toutes les données journalisées sont affichées, y compris les journaux rotatifs ;
- le début d'un démarrage est marqué d'une ligne spéciale.

### ***Utiliser l'affichage continu***

Avec l'affichage continu, les messages journaux sont affichés en temps réel.

```
journalctl -f
```

Cette commande retourne une liste des dix lignes de journal les plus récentes. L'utilitaire `journalctl` continue ensuite de s'exécuter et attend que de nouveaux changements se produisent pour les afficher immédiatement.

### ***Filtrer les messages***

Il est possible d'utiliser différentes méthodes de filtrage pour extraire des informations qui correspondent aux différents besoins. Les messages journaux sont souvent utilisés pour suivre des comportements erronés sur le système. Pour afficher les entrées avec une priorité sélectionnée ou plus élevée :

```
journalctl -p priority
```

Il faut remplacer `priority` par l'un des mots-clés suivants (ou par un chiffre) :

- `debug` (7),
- `info` (6),
- `notice` (5),
- `warning` (4),
- `err` (3),
- `crit` (2),

- alert (1),
- et emerg (0).

---

# 9

## Gestion des tâches

---

### 9.1. Généralités

La planification des tâches est gérée avec l'utilitaire **cron**. Il permet l'exécution périodique des tâches.

Il est réservé à l'administrateur et sous réserve aux utilisateurs et n'utilise qu'une commande : **crontab**.

Le service **cron** sert notamment pour :

- Les opérations d'administration répétitives ;
- Les sauvegardes ;
- La surveillance de l'activité du système ;
- L'exécution de programme.

**crontab** est le diminutif de **chrono table** : table de planification.



Pour mettre en place une planification, il faut que le système soit à l'heure.

### 9.2. Fonctionnement du service

Le fonctionnement du service **cron** est assuré par un démon **crond** présent en mémoire.

Pour vérifier son statut :

```
[root]# service crond status
```



Si le démon **crond** n'est pas en cours de fonctionnement, il faudra l'initialiser manuellement et/ou automatiquement au démarrage. En effet, même si des tâches sont planifiées, elles ne seront pas lancées.

Initialisation du démon **crond** en manuel :

Depuis l'arborescence /etc/rc.d/init.d :

```
[root]# ./crond {status|start|restart|stop}
```

Avec la commande service :

```
[root]# service crond {status|start|restart|stop}
```

Initialisation du démon **crond** au démarrage :

Lors du chargement du système, il est lancé dans les niveaux d'exécution **2 à 5**.

```
[root]# chkconfig --list crond
crond 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt
```

### 9.3. La sécurité

Afin de mettre en oeuvre une planification, un utilisateur doit avoir la permission de se servir du service **cron**.

Cette permission varie suivant les informations contenues dans les fichiers ci-dessous :

- **/etc/cron.allow**
- **/etc/cron.deny**



Si aucun des deux fichiers n'est présent, tous les utilisateurs peuvent utiliser **cron**.

### 9.3.1. Autorisations

#### /etc/cron.allow

Seuls les utilisateurs contenus dans ce fichier sont autorisés à utiliser **cron**.

S'il est vide, aucun utilisateur ne peut utiliser **cron**.



Si cron.allow est présent, cron.deny est ignoré.

#### /etc/cron.deny

Les utilisateurs contenus dans ce fichier ne sont pas autorisés à utiliser **cron**.

S'il est vide, tous les utilisateurs peuvent utiliser **cron**.

### 9.3.2. Autoriser un utilisateur

Seul **user1** pourra utiliser **cron**

```
[root]# vi /etc/cron.allow
user1
```

### 9.3.3. Interdire un utilisateur

Seul **user2** ne pourra pas utiliser **cron**

```
[root]# vi /etc/cron.deny
user2
```

**cron.allow** ne doit pas être présent.

## 9.4. La planification des tâches

Lorsqu'un utilisateur planifie une tâche, un fichier portant son nom est créé sous **/var/spool/cron/**.

Ce fichier contient toutes les informations permettant au démon **crond** de savoir quelle commande ou quel programme lancer et à quel moment le faire (heure, minute, jour ...).

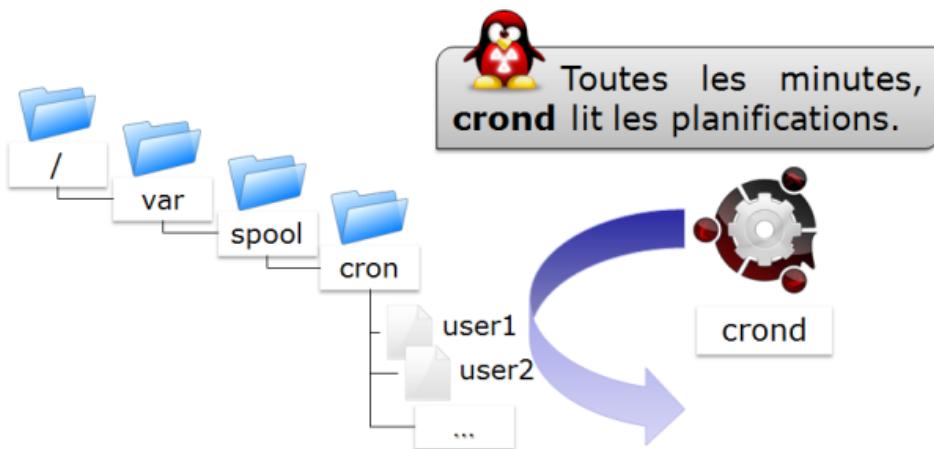


Figure 9.1. Arborescence de cron

#### 9.4.1. Commande crontab

La commande **crontab** permet de gérer le fichier de planification.

```
crontab [-u utilisateur] [-e | -l | -r]
```

Exemple :

```
[root]# crontab -u user1 -e
```

Tableau 9.1. Options de la commande crontab

Option	Description
-e	Edite le fichier de planification avec vi
-l	Affiche le contenu du fichier de planification
-u	Nom de l'utilisateur dont le fichier de planification doit être manipulé
-r	Efface le fichier de planification



crontab sans option efface l'ancien fichier de planification et attend que l'utilisateur rentre de nouvelles lignes. Il faut taper [ctrl] + [d] pour quitter ce mode d'édition.

Seul **root** peut utiliser l'option **-u utilisateur** pour gérer le fichier de planification d'un autre utilisateur.

L'exemple proposé ci-dessus permet à root de planifier une tâche pour l'utilisateur user1.

#### 9.4.2. Intérêts de la planification

Les intérêts de la planification sont multiples et notamment :

- Modifications des fichiers de planification prises en compte immédiatement ;
- Redémarrage inutile.

En revanche, il faut faire attention aux points suivants :

- Le programme doit être autonome ;
- Prévoir des redirections (stdin, stdout, stderr) ;
- Il n'est pas pertinent de lancer des commandes faisant appel à des demandes d'entrée/sortie sur un terminal.



Il faut bien comprendre que le but de la planification est d'effectuer des tâches de façon automatique, donc sans avoir besoin d'une intervention externe.

#### 9.5. Le fichier de planification

Le fichier de planification est structuré et respecte les règles suivantes.

- Chaque ligne de ce fichier correspond à une planification ;
- Chaque ligne comporte six champs, 5 pour le temps et 1 pour la commande ;
- Chaque champs est séparé par un espace ou une tabulation ;
- Chaque ligne se termine par un retour chariot ;
- Un **#** en début de ligne commente celle-ci.

```
[root]# crontab -e  
10 4 1 * * /root/scripts/backup.sh
```

1 2 3 4 5 6

**Tableau 9.2. Champs du fichier de planification**

<b>Champ</b>	<b>Description</b>	<b>Détail</b>
1	Minute(s)	De 0 à 59
2	Heure(s)	De 0 à 23
3	Jour(s) du mois	De 1 à 31
4	Mois de l'année	De 1 à 12
5	Jour(s) de la semaine	De 0 à 7 (0=7=dimanche)
6	Tâche à exécuter	Commande complète ou script



Les tâches à exécuter doivent utiliser des chemins absolus et si possible utiliser des redirections.

Afin de simplifier la notation pour la définition du temps, il est conseillé d'utiliser les symboles spéciaux.

**Tableau 9.3. Métacaractères utilisables**

<b>Métacaractère</b>	<b>Description</b>
*	Toutes les valeurs possibles du champs
-	Indique un intervalle de valeurs
,	Indique une liste de valeurs
/	Définit un pas

Exemples :

Script exécuté le 15 avril à 10h25 :

```
25 10 15 04 * /root/scripts/script > /log/...
```

Exécution à 11h puis à 16h tous les jours :

```
00 11,16 * * * /root/scripts/script > /log/...
```

Exécution toutes les heures de 11h à 16h tous les jours :

```
00 11-16 * * * /root/scripts/script > /log/...
```

Exécution toutes les 10 minutes aux heures de travail :

```
*/10 8-17 * * 1-5 /root/scripts/script > /log/...
```

### 9.5.1. Processus d'exécution d'une tâche

Un utilisateur, Patux, veut éditer son fichier de planification :

- 1 ) crond vérifie s'il est autorisé (/etc/cron.allow et /etc/cron.deny ).
- 2 ) Si c'est le cas, il accède à son fichier de planification ( /var/spool/cron/Pierre ).
- Toutes les minutes crond lit les fichiers de planification.
- 3 ) Il y exécute les tâches planifiées.
- 4 ) Il rend compte systématiquement dans un fichier journal ( /var/log/cron ).

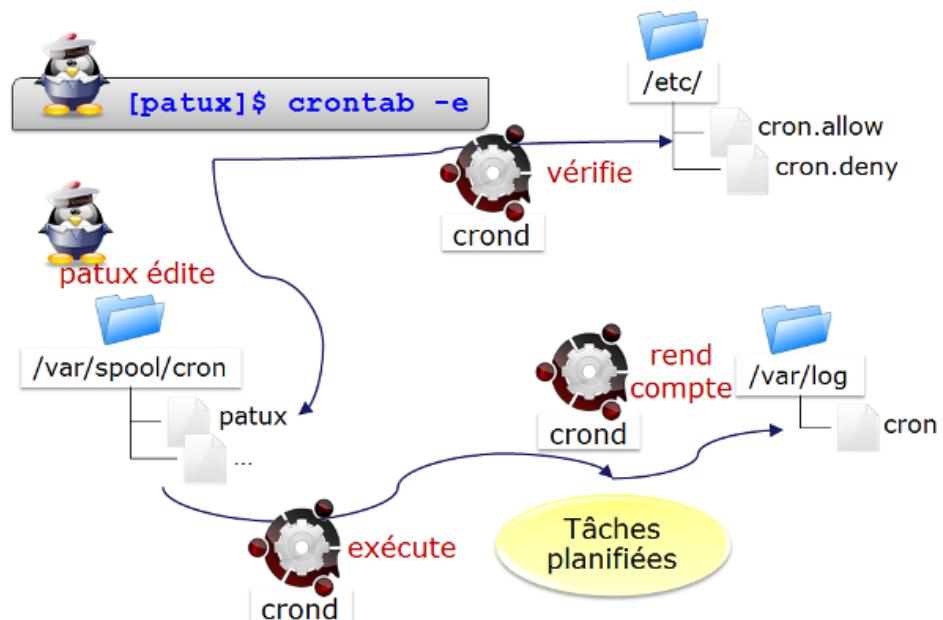


Figure 9.2. Processus d'exécution d'une tâche



# 10

## Mise en oeuvre du réseau

### 10.1. Généralités

Pour illustrer ce cours, nous allons nous appuyer sur l'architecture suivante.

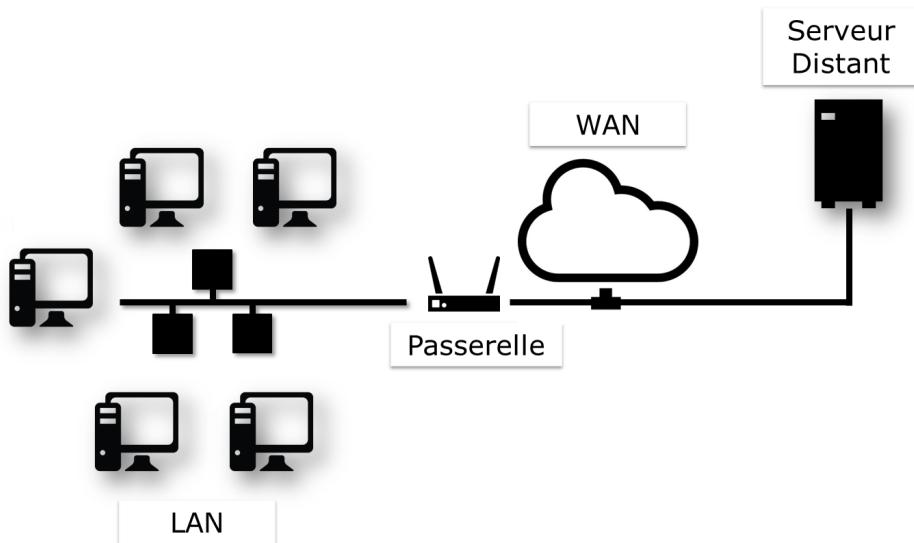


Figure 10.1. Illustration de notre architecture réseau

Elle nous permettra de considérer :

- l'intégration dans un LAN (local area network, ou réseau local) ;
- la configuration d'une passerelle pour joindre un serveur distant ;
- la configuration d'un serveur DNS puis mettre en œuvre la résolution de nom.

Les paramètres minimum à définir propres à la machine sont :

- le nom de la machine ;
- l'adresse IP ;
- le masque de sous-réseau.

Exemple :

- pc-tux ;
- 192.168.1.10 ;
- 255.255.255.0.

La notation appelée CIDR est de plus en plus fréquente : 192.168.1.10/24

Les adresses IP servent au bon acheminement des messages. Elles sont fractionnées en deux parties :

- la partie fixe, identifiant le réseau ;
- l'identifiant de l'hôte dans le réseau.

Le masque de sous-réseau est un ensemble de **4 octets** destiné à isoler :

- l'adresse de réseau (**NetID** ou **SubnetID**) en effectuant un ET logique bit à bit entre l'adresse IP et le masque ;
- l'adresse de l'hôte (**HostID**) en effectuant un ET logique bit à bit entre l'adresse IP et le complément du masque.

Il existe également des adresses spécifiques au sein d'un réseau, qu'il faut savoir identifier. La première adresse d'une plage ainsi que la dernière ont un rôle particulier :

- La première adresse d'une plage est l'**adresse du réseau**. Elle permet d'identifier les réseaux et de router les informations d'un réseau à un autre.
- La dernière adresse d'une plage est l'**adresse de broadcast**. Elle permet de faire de la diffusion à toutes les machines du réseau.

### 10.1.1. Adresse MAC / Adresse IP

Une **adresse MAC** est un identifiant physique inscrit en usine dans une mémoire. Elle est constituée de 6 octets souvent donnée sous forme hexadécimale (par exemple 5E:FF:56:A2:AF:15). Elle se compose de : 3 octets de l'identifiant constructeur et 3 octets du numéro de série.



Cette dernière affirmation est aujourd'hui un peu moins vraie avec la virtualisation. Il existe également des solutions pour changer logiciellement l'adresse MAC.

Une **adresse IP** (Internet Protocol) est un numéro d'identification attribuée de façon permanente ou provisoire à chaque appareil connecté à un réseau informatique utilisant l'Internet Protocol. Une partie définit l'adresse du réseau (NetID ou SubnetID suivant le cas), l'autre partie définit l'adresse de l'hôte dans le réseau (HostID). La taille relative de chaque partie varie suivant le masque de (sous) réseau.

Une adresse IPv4 définit une adresse sur 4 octets. Le nombre d'adresse disponible étant proche de la saturation un nouveau standard a été créé, l'IPv6 définie sur 16 octets.

L'IPv6 est souvent représenté par 8 groupes de 2 octets séparés par un signe deux-points. Les zéro non significatifs peuvent être omis, un ou plusieurs groupes de 4 zéros consécutifs peuvent être remplacés par un double deux-points.

Les masques de sous-réseaux ont de 0 à 128 bits.

(par exemple 21ac:0000:0000:0611:21e0:00ba:321b:54da/64 ou  
21ac::611:21e0:ba:321b:54da/64)

Dans une adresse web ou URL (Uniform Resource Locator), une adresse ip peut être suivi de deux-points, l'adresse de port (qui indique l'application à laquelle les données sont destinées). Aussi pour éviter toute confusion dans une URL, l'adresse IPv6 s'écrit entre crochets [ ], deux-points, adresse de port.

Les adresses IP et MAC doivent être uniques sur un réseau !



Sous VMWare, choisir l'option « I copied it » au lancement d'une VM génère une nouvelle adresse MAC.

### 10.1.2. Domaine DNS

Les postes clients peuvent faire partie d'un domaine DNS (**Domain Name System**, système de noms de domaine, par exemple mondomaine.lan).

Le nom de machine pleinement qualifié (FQDN) devient pc-tux.mondomaine.lan.

Un ensemble d'ordinateurs peut être regroupé dans un ensemble logique, permettant la résolution de nom, appelé domaine DNS. Un domaine DNS n'est pas, bien entendu, limité à un seul réseau physique.

Pour qu'un ordinateur intègre un domaine DNS, il faudra lui spécifier un suffixe DNS (ici mondomaine.lan) ainsi que des serveurs qu'il pourra interroger.

### 10.1.3. Rappel du modèle OSI



Aide mémoire : Pour se souvenir de l'ordre PLRTSPA, retenir la phrase suivante : *Pour Les Réseaux Tu Seras Pas Augmenté*.

Tableau 10.1. Les 7 couches du modèle OSI

Couche	Protocoles
7 - Application	POP, IMAP, SMTP, SSH, SNMP, HTTP, FTP, ...
6 - Présentation	ASCII, MIME, ...
5 - Session	TLS, SSL, NetBIOS, ...
4 - Transport	TLS, SSL, TCP, UDP, ...
3 - Réseau	IPv4, IPv6, ARP, ...
2 - Liaison	Ethernet, WiFi, Token Ring, ...
1 - Physique	Câbles, fibres optiques, ondes radio, ...

**La couche 1** (Physique) prend en charge la transmission sur un canal de communication (Wifi, Fibre optique, câble RJ, etc.). Unité : le bit.

**La couche 2** (Liaison) prend en charge la topologie du réseau (Token-ring, étoile, bus, etc.), le fractionnement des données et les erreurs de transmissions. Unité : la trame.

**La couche 3** (Réseau) prend en charge la transmission de bout en bout des données (routage IP = Passerelle). Unité : le paquet.

**La couche 4** (Transport) prend en charge le type de service (connecté ou non connecté), le chiffrement et le contrôle de flux. Unité : le segment ou le datagramme.

**La couche 7** (Application) représente le contact avec l'utilisateur. Elle apporte les services offerts par le réseau : http, dns, ftp, imap, pop, smtp, etc.

## 10.2. Le nommage des interfaces

**lo** est l'interface “**loopback**” qui permet à des programmes TCP/IP de communiquer entre eux sans sortir de la machine locale. Cela permet de tester si le module « **réseau** » du système fonctionne bien et aussi de faire un ping localhost. Tous les paquets qui entrent par localhost ressortent par localhost. Les paquets reçus sont les paquets envoyés.

Le noyau Linux attribue des noms d'interfaces composés d'un préfixe précis selon le type. Sur des distributions Linux RHEL 6, toutes les interfaces **Ethernet**, par exemple, commencent par **eth**. Le préfixe est suivi d'un chiffre, le premier étant 0 (eth0, eth1, eth2...). Les interfaces wifi se voient attribuées un préfixe **wlan**.

## 10.3. Utiliser la commande IP

Oubliez l'ancienne commande **ifconfig** ! Pensez **ip** !



Commentaire à destination des administrateurs d'anciens systèmes Linux :

La commande historique de gestion du réseau est **ifconfig**. Cette commande a tendance à être remplacée par la commande **ip**, déjà bien connue des administrateurs réseaux.

La commande **ip** est la commande unique pour gérer l'adresse IP, ARP, le routage, etc.

La commande ifconfig n'est plus installée par défaut sous RHEL 7. Il est important de prendre des bonnes habitudes dès maintenant.

## 10.4. Le nom de machine

La commande hostname affiche ou définit le nom d'hôte du système

### Syntaxe de la commande hostname.

```
hostname [-f] [hostname]
```

Tableau 10.2. Options principales de la commande hostname

Option	Description
-f	Affiche le FQDN
-i	Affiche les adresses IP du système



Cette commande est utilisée par différents programmes réseaux pour identifier la machine.

Pour affecter un nom d'hôte, il est possible d'utiliser la commande hostname, mais les changements ne seront pas conservés au prochain démarrage. La commande sans argument permet d'afficher le nom de l'hôte.

Pour fixer le nom d'hôte, il faut modifier le fichier **/etc/sysconfig/network** :

### Le fichier /etc/sysconfig/network.

```
NETWORKING=yes
HOSTNAME=stagiaire.mondomaine.lan
```

Le script de démarrage sous RedHat consulte également le fichier **/etc/hosts** pour résoudre le nom d'hôte du système.

Lors du démarrage du système, Linux vient évaluer la valeur **HOSTNAME** du fichier **/etc/sysconfig/network**.

Il utilise ensuite le fichier **/etc/hosts** pour évaluer l'adresse IP principale du serveur et son nom d'hôte. Il en déduit le nom de domaine DNS.

Il est donc primordiale de bien renseigner ces deux fichiers avant toute configuration de services réseaux.



Pour savoir si cette configuration est bien faîte, les commandes hostname et hostname -f doivent répondre les bonnes valeurs attendues.

## 10.5. Le fichier /etc/hosts

Le fichier **/etc/hosts** est une table de correspondance statique des noms d'hôtes, qui respecte le format suivant :

### Syntaxe du fichier /etc/hosts.

```
@IP <nom d'hôte> [alias] [# commentaire]
```

Exemple de fichier /etc/hosts :

### Exemple de fichier /etc/hosts.

```
127.0.0.1 localhost localhost.localdomain
::1 localhost localhost.localdomain
192.168.1.10 stagiaire.mondomaine.lan stagiaire
```

Le fichier **/etc/hosts** est encore employé par le système, notamment lors du démarrage durant lequel le FQDN du système est déterminé.



RedHat préconise qu'au moins une ligne contenant le nom du système soit renseignée.

Si le service DNS (Domain Name Service) n'est pas en place, vous devez renseigner tous les noms dans le fichier hosts de chacune de vos machines.

Le fichier **/etc/hosts** contient une ligne par entrée, comportant l'adresse IP, le FQDN, puis le nom d'hôte (dans cet ordre) et une suite d'alias (alias1 alias2 ...). L'alias est une option.

## 10.6. Le fichier /etc/nsswitch.conf

Le Name Service Switch (NSS) permet de substituer des fichiers de configuration (par exemple /etc/passwd, /etc/group, /etc/hosts) par une ou plusieurs bases de données centralisées

Le fichier /etc/nsswitch.conf permet de configurer les bases de données du service de noms.

### Le fichier /etc/nsswitch.conf.

```
passwd: files
shadow: files
group: files

hosts: files dns
```

Dans le cas présent, Linux cherchera en premier une correspondance de noms d'hôtes (ligne hosts:) dans le fichier /etc/hosts (valeur files) avant d'interroger le DNS (valeur dns)! Ce comportement peut simplement être changé en éditant le fichier /etc/nsswitch.conf.

Bien évidemment, il est possible d'imaginer interroger un serveur LDAP, Mysql ou autre en configurant le service de noms pour répondre aux requêtes du systèmes sur les hosts, les utilisateurs, les groupes, etc.

La résolution du service de noms peut être testée avec la commande getent que nous verrons plus loin dans ce cours.

## 10.7. Le fichier /etc/resolv.conf

Le fichier /etc/resolv.conf contient la configuration de la résolution de nom DNS.

### /etc/resolv.conf.

```
#Generated by NetworkManager
domain mondomaine.lan
search mondomaine.lan
nameserver 192.168.1.254
```



Ce fichier est historique. Il n'est plus renseigné directement !

Les nouvelles générations de distributions ont généralement intégré le service NetworkManager. Ce service permet de gérer plus efficacement la configuration, que ce soit en mode graphique ou console.

Il permet notamment de configurer les serveurs DNS depuis le fichier de configuration d'une interface réseau. Il se charge alors de renseigner dynamiquement le fichier /etc/resolv.conf qui ne devrait jamais être édité directement, sous peine de perdre les changements de configuration au prochain démarrage du service réseau.

## 10.8. La commande IP

La commande ip du paquet iproute2 permet de configurer une interface et sa table routage.

Afficher les interfaces :

```
[root]# ip link
```

Afficher les informations des interfaces :

```
[root]# ip addr show
```

Afficher les informations d'une interface :

```
[root]# ip addr show eth0
```

Afficher la table ARP:

```
[root]# ip neigh
```

Toutes les commandes historiques de gestion du réseau ont été regroupées sous la commande IP, bien connue des administrateurs réseaux.

## 10.9. Configuration DHCP

Le protocole DHCP (Dynamic Host Control Protocol) permet d'obtenir via le réseau une configuration IP complète. C'est le mode de configuration par défaut d'une interface réseau sous Redhat, ce qui explique qu'un système branché sur le réseau d'une box internet puisse fonctionner sans configuration supplémentaire.

La configuration des interfaces sous RHEL 6 se fait dans le dossier **/etc/sysconfig/network-scripts/**.

Pour chaque interface ethernet, un fichier **ifcfg-ethX** permet de configurer l'interface associée.

**/etc/sysconfig/network-scripts/ifcfg-eth0.**

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
HWADDR=00:0c:29:96:32:e3
```

- Nom de l'interface : (doit être dans le nom du fichier)

```
DEVICE=eth0
```

- Démarrer automatiquement l'interface :

```
ONBOOT=yes
```

- Effectuer une requête DHCP au démarrage de l'interface :

```
BOOTPROTO=dhcp
```

- Spécifier l'adresse MAC (facultatif mais utile lorsqu'il y a plusieurs interfaces) :

```
HWADDR=00:0c:29:96:32:e3
```



Si NetworkManager est installé, les modifications sont prises en compte automatiquement. Sinon, il faut redémarrer le service réseau.

Redémarrer le service réseau :

```
[root]# service network restart
```

et sous RHEL 7 :

```
[root]# systemctl restart network
```

## 10.10. Configuration statique

La configuration statique nécessite à minima :

**/etc/sysconfig/network-scripts/ifcfg-eth0.**

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.1.10
NETMASK=255.255.255.0
```

- Ne pas utiliser DHCP = configuration statique

```
BOOTPROTO=none
```

- Adresse IP :

```
IPADDR=192.168.1.10
```

- Masque de sous-réseau :

```
NETMASK=255.255.255.0
```

- Le masque peut être spécifié avec un préfixe :

```
PREFIX=24
```



Il faut utiliser NETMASK OU PREFIX - Pas les deux !



Pensez à redémarrer le service network !

## 10.11. Routage

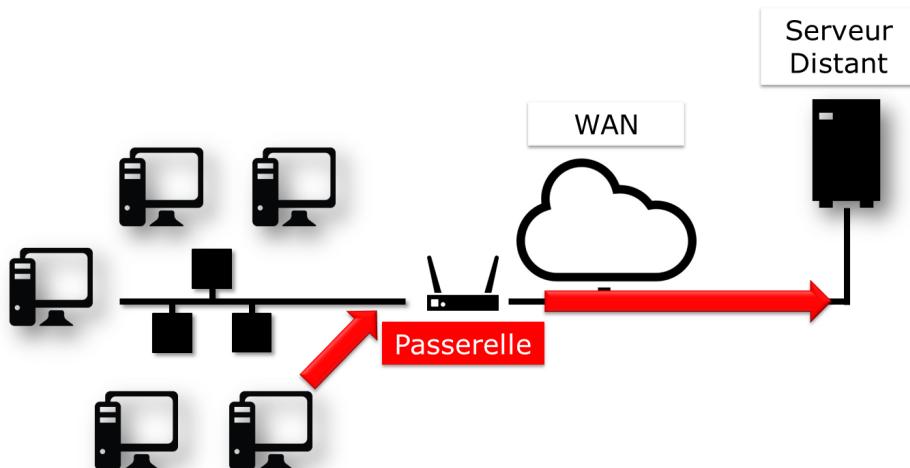


Figure 10.2. Architecture réseau avec une passerelle

/etc/sysconfig/network-scripts/ifcfg-eth0.

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
HWADDR=00:0c:29:96:32:e3
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
```



Pensez à redémarrer le service network !

### La commande ip route.

```
[root]# ip route show
192.168.1.0/24 dev eth0 [...] src 192.168.1.10 metric 1
default via 192.168.1.254 dev eth0 proto static
```

Il est judicieux de savoir lire une table de routage, surtout dans un environnement disposant de plusieurs interfaces réseaux.

- Dans l'exemple présenté, le réseau 192.168.1.0/24 est directement accessible depuis le périphérique eth0, il y a donc une métrique à 1 (ne traverse pas de routeur).
- Tous les autres réseaux que le réseau précédent seront joignables, toujours depuis le périphérique eth0, mais cette fois-ci les paquets seront adressés à une passerelle 192.168.1.254. Le protocole de routage est un protocole statique (bien qu'il soit possible d'ajouter un protocole de routage dynamique sous Linux).

### 10.12. Résolution de noms

Un système a besoin de résoudre :

- des FQDN en adresses IP

```
www.free.fr = 212.27.48.10
```

- des adresses IP en noms

```
212.27.48.10 = www.free.fr
```

- ou d'obtenir des informations sur une zone :

```
MX de free.fr = 10 mx1.free.fr + 20 mx2.free.fr
```

### /etc/sysconfig/network-scripts/ifcfg-eth0.

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
```

```
HWADDR=00:0c:29:96:32:e3
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
DNS1=172.16.1.2
DNS2=172.16.1.3
DOMAIN=mondomaine.lan
```

Dans ce cas, pour joindre les DNS, il faut passer par la passerelle.

#### **/etc/resolv.conf.**

```
#Generated by NetworkManager
domain mondomaine.lan
search mondomaine.lan
nameserver 172.16.1.2
nameserver 172.16.1.3
```

Le fichier a bien été mis à jour par NetworkManager.

### **10.13. Dépannage**

La commande ping permet d'envoyer des datagrammes à une autre machine et attend une réponse.

C'est la commande de base pour tester le réseau car elle vérifie la connectivité entre votre interface réseau et une autre.

#### **La syntaxe de la commande ping.**

```
ping [-c numérique] destination
```

L'option -c (count) permet de stoper la commande au bout du décompte exprimé en seconde.

Exemple :

```
[root]# ping -c 4 localhost
```



Validez la connectivité du plus proche au plus lointain

### 1) Valider la couche logicielle TCP/IP

```
[root]# ping localhost
```

« Pinguer » la boucle interne ne permet pas de détecter une panne matérielle sur l'interface réseau. Elle permet simplement de déterminer si la configuration logicielle IP est correcte.

### 2) Valider la carte réseau

```
[root]# ping 192.168.1.10
```

Pour déterminer que la carte réseau est fonctionnelle, il faut maintenant faire un « ping » de son adresse IP. La carte réseau, si le câble réseau n'est pas connecté, devrait être dans un état « down ».

Si le ping ne fonctionne pas, vérifier dans un premier temps le câble réseau vers votre le commutateur réseau et remonter l'interface (voir la commande if up), puis vérifiez l'interface elle-même.

### 3) Valider la connectivité de la passerelle

```
[root]# ping 192.168.1.254
```

### 4) Valider la connectivité d'un serveur distant

```
[root]# ping 172.16.1.2
```

### 5) Valider le service DNS

```
[root]# ping www.free.fr
```

## **10.13.1. La commande dig**

La commande dig (dig : en français miner, chercher en profondeur) permet d'interroger le serveur DNS.

### Syntaxe de la commande dig.

```
dig [-t type] [+short] [name]
```

Exemples :

```
[root]# dig +short www.formatux.fr  
46.19.120.31  
[root]# dig -t MX +short formatux.fr  
10 smtp.formatux.fr.
```

La commande **DIG** permet d'interroger les **serveurs DNS**. Elle est par défaut très verbeuse, mais ce comportement peut être changé grâce à l'option **+short**.

Il est également possible de spécifier un **type d'enregistrement** DNS à résoudre, comme par exemple un **type MX** pour obtenir des renseignements sur les serveurs de messagerie d'un domaine.

Pour plus d'informations à ce sujet, voir le cours « DNS avec Bind9 »

### **10.13.2. La commande getent**

La commande **getent** (get entry) permet d'obtenir une entrée de NSSwitch (hosts + dns)

**Syntaxe de la commande getent.**

```
getent hosts name
```

Exemple :

```
[root]# getent hosts www.formatux.fr  
46.19.120.31 www.formatux.fr
```

Interroger uniquement un serveur DNS peut renvoyer un résultat erroné qui ne prendrait pas en compte le contenu d'un fichier hosts, bien que ce cas de figure devrait être rare aujourd'hui.

Pour prendre en compte également le fichier **/etc/hosts**, il faut interroger le service de noms NSSwitch, qui se chargera d'une éventuelle résolution DNS.

### 10.13.3. La commande ipcalc

La commande ipcalc (ip calcul) permet de calculer l'adresse d'un réseau ou d'un broadcast depuis une adresse IP et un masque.

#### Syntaxe de la commande ipcalc.

```
ipcalc [options] IP <netmask>
```

Exemple :

```
[root]# ipcalc -b 172.16.66.203 255.255.240.0
BROADCAST=172.16.79.255
```



Cette commande est intéressante suivie d'une redirection pour renseigner automatiquement les fichiers de configuration de vos interfaces :

```
[root]# ipcalc -b 172.16.66.203 255.255.240.0 >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

Option	Description
-b	Affiche l'adresse de broadcast.
-n	Affiche l'adresse du réseau et du masque.

**ipcalc** permet de calculer simplement les informations IP d'un hôte. Les diverses options indiquent quelles informations ipcalc doit afficher sur la sortie standard. Des options multiples peuvent être indiquées. Une adresse IP sur laquelle opérer doit être spécifiée. La plupart des opérations nécessitent aussi un masque réseau ou un préfixe CIDR.

Tableau 10.3. Options principales de la commande ipcalc

Option courte	Option longue	Description
-b	--broadcast	Affiche l'adresse de diffusion de l'adresse IP donnée et du masque réseau.
-h	--hostname	Affiche le nom d'hôte de l'adresse IP donnée via le DNS.

Option courte	Option longue	Description
-n	--netmask	Calcule le masque réseau de l'adresse IP donnée. Suppose que l'adresse IP fait partie d'un réseau de classe A, B, ou C complet. De nombreux réseaux n'utilisent pas les masques réseau par défaut, dans ce cas une valeur incorrecte sera retournée.
-p	--prefix	Indique le préfixe de l'adresse masque/IP.
-n	--network	Indique l'adresse réseau de l'adresse IP et du masque donné.
-s	--silent	N'affiche jamais aucun message d'erreur.

#### 10.13.4. La commande ss

La commande ss (socket statistics) affiche les ports en écoute sur le réseau

**Syntaxe de la commande ss.**

```
ss [-tuna]
```

Exemple :

```
[root]# ss -tuna
tcp    LISTEN  0      128    *:22    *:*
```

Les commandes **SS** et **NETSTAT** (à suivre) vont se révéler très importantes pour la suite de votre cursus Linux.

Lors de la mise en œuvre des services réseaux, il est très fréquent de vérifier avec l'une de ces deux commandes que le service est bien en écoute sur les ports attendus.

#### 10.13.5. La commande netstat

La commande netstat (network statistics) affiche les ports en écoute sur le réseau

## Syntaxe de la commande netstat.

```
netstat -tagn
```

Exemple :

```
[root]# netstat -tagn
tcp  0  0  0.0.0.0:22  0.0.0.0:*  LISTEN  2161/sshd
```

### **10.13.6. Les conflits d'adresses IP ou d'adresses MAC**

Un défaut de configuration peut amener plusieurs interfaces à utiliser la même adresse IP. Cette situation peut se produire lorsqu'un réseau dispose de plusieurs serveurs DHCP ou lorsque la même adresse IP est manuellement assignée plusieurs fois.

Lorsque le réseau fonctionne mal, que des dysfonctionnements ont lieu, et qu'un conflit d'adresses IP pourrait en être à l'origine, il est possible d'utiliser le logiciel arp-scan (nécessite le dépôt EPEL) :

```
$ yum install arp-scan
```

Exemple :

```
$ arp-scan -I eth0 -l

172.16.1.104  00:01:02:03:04:05      3COM CORPORATION
172.16.1.107  00:0c:29:1b:eb:97      VMware, Inc.
172.16.1.250  00:26:ab:b1:b7:f6      (Unknown)
172.16.1.252  00:50:56:a9:6a:ed      VMWare, Inc.
172.16.1.253  00:50:56:b6:78:ec      VMWare, Inc.
172.16.1.253  00:50:56:b6:78:ec      VMWare, Inc. (DUP: 2)
172.16.1.253  00:50:56:b6:78:ec      VMWare, Inc. (DUP: 3)
172.16.1.253  00:50:56:b6:78:ec      VMWare, Inc. (DUP: 4)
172.16.1.232  88:51:fb:5e:fa:b3      (Unknown) (DUP: 2)
```



Comme l'exemple ci-dessus le démontre, il est également possible d'avoir des conflits d'adresses MAC ! Ces

problématiques sont apportées par les technologies de virtualisation et la recopie de machines virtuelles.

## 10.14. Configuration à chaud

La commande ip peut ajouter à chaud une adresse IP à une interface

```
ip addr add @IP dev DEVICE
```

Exemple :

```
[root]# ip addr add 192.168.2.10 dev eth1
```

La commande ip permet d'activer ou désactiver une interface :

```
ip link set DEVICE up  
ip link set DEVICE down
```

Exemple :

```
[root]# ip link set eth1 up  
[root]# ip link set eth1 down
```

La commande ip permet d'ajouter une route :

```
ip route add [default|netaddr] via @IP [dev device]
```

Exemple :

```
[root]# ip route add default via 192.168.1.254  
[root]# ip route add 192.168.100.0/24 via 192.168.2.254 dev eth1
```

## 10.15. En résumé

Les fichiers mis en oeuvre durant ce chapitre sont :

## Fichiers de configuration

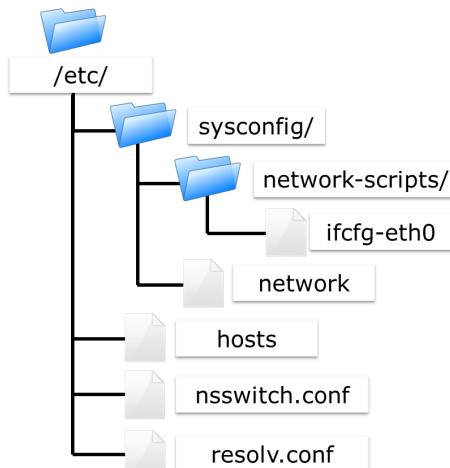


Figure 10.3. Synthèse des fichiers mis en oeuvre dans la partie réseau

Une configuration complète d'une interface pourrait être celle-ci :

**/etc/sysconfig/network-scripts/ifcfg-eth0.**

```

DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
HWADDR=00:0c:29:96:32:e3
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
DNS1=172.16.1.1
DNS2=172.16.1.2
DOMAIN=formatux.fr

```

La méthode de dépannage doit aller du plus proche au plus lointain :

1. ping localhost (test logiciel)
2. ping adresseIP (test matériel)
3. ping passerelle (test connectivité)
4. ping serveur-distant (test routage)
5. Interrogation DNS (dig ou ping)

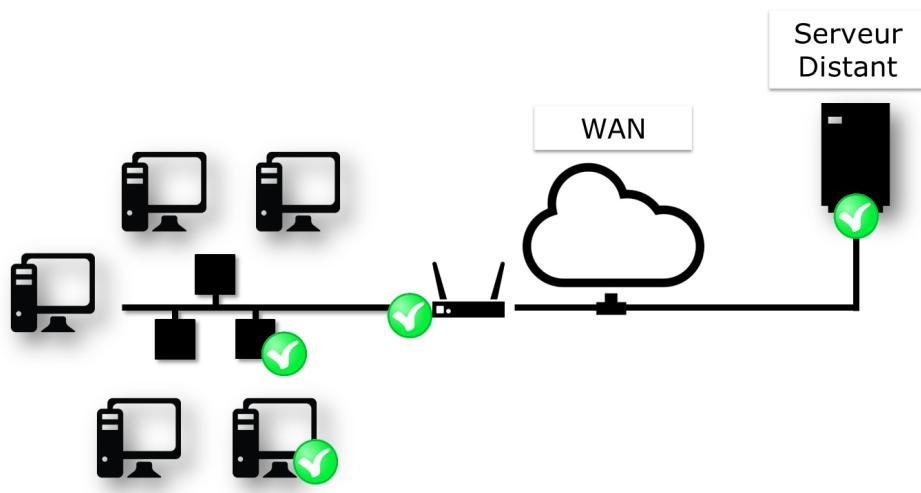


Figure 10.4. Méthode de dépannage ou de validation du réseau

---

# 11

## Gestion des logiciels

---

### 11.1. Généralités

Sur un système Linux, il est possible d'installer un logiciel de deux façons :

- en utilisant un paquet d'installation ;
- en compilant les fichiers sources.

**Le paquet** : Il s'agit d'un unique fichier comprenant toutes les données utiles à l'installation du programme. Il peut être exécuté directement sur le système à partir d'un dépôt logiciel.

**Les fichiers sources** : Certains logiciels ne sont pas fournis dans des paquets prêts à être installés mais via une archive contenant les fichiers sources. Charge à l'administrateur de préparer ces fichiers et de les compiler pour installer le programme.

### 11.2. RPM : RedHat Package Manager

**RPM** (RedHat Package Manager) est un système de gestion des logiciels. Il est possible d'installer, de désinstaller, de mettre à jour ou de vérifier des logiciels contenus dans des paquets.

**RPM** est le format utilisé par toutes les distributions à base RedHat (Fedora, CentOS, SuSe, Mandriva, ...). Son équivalent dans le monde de Debian est DPKG (Debian Package).

Le nom d'un paquet RPM répond à une nomenclature précise :

**Figure 11.1. nom-version-edition.architecture.rpm**

### 11.2.1. Commande rpm

La commande rpm permet d'installer un paquet.

#### Syntaxe de la commande rpm.

```
rpm [-i][-U] paquet.rpm [-e] paquet
```

Exemple :

```
[root]# rpm -ivh paquet.rpm
```

**Tableau 11.1. Options de la commande rpm**

Option	Description
-i <i>paquet.rpm</i>	Installe le paquet.
-U <i>paquet.rpm</i>	Met à jour un paquet déjà installé.
-e <i>paquet.rpm</i>	Désinstalle le paquet.
-h	Affiche une barre de progression.
-v	Informe sur l'avancement de l'opération en cours.
--test	Teste la commande sans l'exécuter.

La commande rpm permet aussi d'interroger la base de données des paquets du système en ajoutant l'option **-q**.

Il est possible d'exécuter plusieurs types de requêtes pour obtenir différentes informations sur les paquets installés. La base de donnée RPM se trouve dans le répertoire **/var/lib/rpm** .

Exemple :

```
[root]# rpm -qa
```

Cette commande interroge tous les paquets installés sur le système.

### Syntaxe de la commande rpm pour requêter.

```
rpm -q [-a][-i][-l] paquet [-f] fichier
```

Exemple :

```
[root]# rpm -qil paquet
[root]# rpm -qf /chemin/fichier
```

Tableau 11.2. Options de la commande rpm pour requêter

Option	Description
-a paquet.rpm	Liste tous les paquets installés sur le système.
-i paquet.rpm	Affiche les informations du paquet.
-l paquet.rpm	Liste les fichiers contenus dans le paquet.
-f	Affiche le nom du paquet contenant le fichier précisé.



Après l'option **-q**, le nom du paquet doit être exact. Les métacaractères ne sont pas gérés.



Il est cependant possible de lister tous les paquets installés et de filtrer avec la commande **grep**.

### 11.3. YUM : Yellow dog Updater Modified

YUM est un gestionnaire de paquets logiciels. Il fonctionne avec des paquets RPM regroupés dans un dépôt (un répertoire de stockage des paquets) local ou distant.

La commande **yum** permet la gestion des paquets en comparant ceux installés sur le système à ceux présents dans les dépôts définis sur le serveur. Elle permet aussi d'installer automatiquement les dépendances, si elles sont également présentes dans les dépôts.

**YUM** est le gestionnaire utilisé par de nombreuses distributions à base RedHat (Fedora, CentOS, ...). Son équivalent dans le monde Debian est APT (Advanced Packaging Tool).

### 11.3.1. Commande yum

La commande yum permet d'installer un paquet en ne spécifiant que le nom court.

#### Syntaxe de la commande yum.

```
yum [install][remove][list all][search][info] paquet
```

Exemple :

```
[root]# yum install tree
```

Seul le nom court du paquet est nécessaire.

Tableau 11.3. Options de la commande yum

Option	Description
<i>install</i>	Installe le paquet.
<i>remove</i>	Désinstalle le paquet.
<i>list all</i>	Liste les paquets déjà présents dans le dépôt.
<i>search</i>	Recherche un paquet dans le dépôt.
<i>provides */ nom_cmde</i>	Recherche une commande.
<i>info</i>	Affiche les informations du paquet.

La commande **yum list** liste tous les paquets installés sur le système et présents dans le dépôt. Elle accepte plusieurs paramètres :

Tableau 11.4. Paramètres de la commande yum list

Paramètre	Description
<i>all</i>	Liste les paquets installés puis ceux disponibles sur les dépôts.
<i>available</i>	Liste uniquement les paquets disponibles pour l'installation.
<i>updates</i>	Liste les paquets pouvant être mis à jour.
<i>obsoletes</i>	Liste les paquets rendus obsolètes par des versions supérieures disponibles.
<i>recent</i>	Liste les derniers paquets ajoutés au dépôt.

Exemple de recherche de la commande semanage:

```
[root]# yum provides */semanage
```

### 11.3.2. Fonctionnement de YUM

Sur un poste client, le gestionnaire YUM s'appuie sur un ou plusieurs fichiers de configuration afin de cibler les dépôts contenant les paquets RPM.

Ces fichiers sont situés dans **/etc/yum.repos.d/** et se terminent obligatoirement par **.repo** afin d'être exploités par YUM.

Exemple :

```
/etc/yum.repos.d/MonDepotLocal.repo
```

Chaque fichier **.repo** se constitue au minimum des informations suivantes, une directive par ligne. Exemple:

```
[DepotLocal]      #Nom court du dépôt
name=Mon dépôt local      #Nom détaillé
baseurl=http://..... ou file:///.....      #Adresse http ou local
enabled=1      #Activation =1, ou non activé =0"
gpgcheck=1      #Dépôt demandant une signature
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6 #Chemin de la clef
publique GPG
```

Par défaut, la directive **enabled** est absente ce qui signifie que le dépôt est activé. Pour désactiver un dépôt, il faut spécifier la directive **enabled=0**.

## 11.4. Gérer son dépôt

La création d'un dépôt permet de disposer de sa propre banque de paquets. Celle-ci peut-être disponible par exemple par point de montage ou mise à disposition sur un serveur web.

Les étapes de la création d'un dépôt sur un serveur sont les suivantes :

- Créer un répertoire qui va accueillir tous les paquets rpm ;

```
[root]# mkdir /MonDepot
```

- Copier tous les paquets rpm nécessaires dans ce dossier ;

```
[root]# cp ../../*.rpm /MonDepot/
```

- Créer la structure et générer le dépôt à l'aide de la commande **createrepo** ;

```
[root]# createrepo /MonDepot
```

- Configurer les fichiers **.repo** des clients afin qu'ils puissent installer les paquets depuis ce serveur (réinitialiser le cache des clients si besoin avec **yum clean all**).

## 11.5. Le dépôt EPEL

Le dépôt **EPEL** (**E**xtra **P**ackages for **E**nterprise **L**inux) est un dépôt contenant des paquets logiciels supplémentaires pour Entreprise Linux, ce qui inclut Red Hat Entreprise Linux (RHEL), CentOS, etc.

### 11.5.1. Installation

Télécharger et installer le rpm du dépôt :

Si vous êtes derrière le proxy internet de l'école

```
[root]# export http_proxy=http://10.10.10.7:8080
```

- Pour une CentOS 6 :

```
[root]# rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Après avoir installé le paquet RPM du dépôt :

```
[root]# yum update
```

---

## **Partie II. Sécurité**



---

# Table des matières

12. Elévation des privilèges (su et sudo) .....	247
12.1. Limiter le compte root .....	248
12.2. La commande su .....	248
12.3. La commande sudo .....	249
12.3.1. Avantages .....	250
12.4. Commande visudo .....	250
12.4.1. Le fichier /etc/sudoers .....	251
12.4.2. Le groupe wheel .....	251
12.4.3. Ne plus utiliser root .....	251
12.4.4. Restreindre le sudo .....	252
13. Les modules d'authentification PAM .....	257
13.1. Généralités .....	257
13.1.1. Syntaxe d'une directive .....	258
13.2. Les mécanismes .....	259
13.2.1. Le mécanisme auth - Authentification .....	259
13.2.2. Le mécanisme account - Gestion de compte .....	259
13.2.3. Le mécanisme session - Gestion de session .....	259
13.2.4. Le mécanisme password - Gestion des mots de passe ..	260
13.3. Les indicateurs de contrôle .....	260
13.3.1. L'indicateur de contrôle required .....	260
13.3.2. L'indicateur de contrôle requisite .....	260
13.3.3. L'indicateur de contrôle sufficient .....	261
13.3.4. L'indicateur de contrôle optional .....	261
13.3.5. En conclusion .....	262
13.4. Les modules de PAM .....	262
13.4.1. Le module pam_unix .....	262
13.4.2. Le module pam_cracklib .....	263
13.4.3. Le module pam_tally .....	264
13.4.4. Le module pam_time .....	265
13.4.5. Le module pam_nologin .....	266
13.4.6. Le module pam_wheel .....	266
13.4.7. Le module pam_mount .....	267
14. Securisation SELinux .....	269
14.1. Généralités .....	269

---

14.1.1. Le contexte SELinux .....	270
14.2. Gestion .....	273
14.2.1. Administrer les objets de type booléens .....	274
14.3. Mode de fonctionnement .....	275
14.3.1. Le fichier /etc/sysconfig/selinux .....	276
14.4. Les jeux de règles (Policy Type) : .....	277
14.5. Contexte .....	277
14.5.1. Aller plus loin avec SELinux .....	279
15. IPtables, le parefeu Linux .....	281
15.1. Gestion du pare-feu .....	282
15.1.1. Démarrer/arrêter ou redémarrer le parefeu .....	282
15.1.2. Afficher les règles .....	282
15.2. Quelques exemples .....	283
15.2.1. Bloquer des adresses IP spécifiques .....	283
15.2.2. Bloquer/accepter des ports spécifiques .....	283
15.2.3. Autoriser un réseau .....	284
15.2.4. Limiter le nombre de connexion d'une adresse .....	284
15.2.5. Bloquer le protocole ICMP (ping) .....	284
15.2.6. Accès à la loopback .....	285
15.2.7. Logger les paquets refusés .....	285
15.2.8. Gérer les connexions établies .....	285
15.2.9. Supprimer les paquets invalides .....	285
15.2.10. Bloquer le trafic SMTP sortant .....	285
15.3. Conclusion .....	286
16. Fail2ban .....	287
16.1. Installation .....	287
16.2. Configuration .....	287
16.3. Lancement du service .....	289
16.4. Vérification du service .....	289
16.5. Interface graphique .....	290
17. Sécuriser le serveur SSH .....	291
17.1. Configuration .....	291
17.2. Changer le port d'écoute et la version du protocole .....	291
17.3. Utilisation de clefs privées/publiques .....	291
17.4. Limiter les accès .....	292
17.5. Interdire l'accès à root !!! .....	292

---

17.6. Sécurité par le parefeu .....	292
18. Autorité de certification TLS avec easy-rsa .....	293
18.1. Installer easy-rsa : .....	293
18.2. Configuration .....	293
18.3. Créer une autorité de certification .....	294
18.4. Créer une biclé serveur .....	295
18.5. Installer le certificat de l'autorité de certification .....	295
Glossaire .....	297



---

# 12

## Elévation des privilèges (su et sudo)

---

Le compte du super-utilisateur (root) possède de nombreuses contraintes car :

- il a tous les droits ;
- il peut causer des dommages irréparables ;
- il est la cible privilégiée des intrus.

L'administrateur qui travaille sur le serveur, se connecte avec son compte (qui dispose de droits restreints), puis au moment où il doit réaliser une tâche d'administration, endosse l'identité de **root** de façon temporaire, exécute ses actions, puis reprend son identité d'origine.

Il convient donc d'effectuer les opérations suivantes :

- interdire le login root ;
- restreindre l'accès à la commande su ;
- privilégier la commande sudo ;
- robustesse du mot de passe :
  - il doit être complexe ;
  - non issu du dictionnaire ;
  - comporter des minuscules, majuscules, lettres, chiffres et caractères spéciaux ;
- changer régulièrement (ne pas être conservé identique indéfiniment) ;
- sécurisation (ne doit pas être inscrit à proximité des postes ou communiqué à des personnes qui n'ont pas à le connaître).



Ces principes sont les premières règles de sécurité.

## 12.1. Limiter le compte root

Le fichier **/etc/security** contient la liste des terminaux accessibles par le login **root**.

Pour interdire un terminal au login **root**, il faut :

- soit mettre la ligne en commentaire ;
- soit supprimer la ligne.

**En fonction des distributions, la syntaxe du fichier /etc/security peut changer.**

Pour une distribution **CentOS**, le fichier est constitué d'une ligne **tty** et d'une ligne **vc**, indispensable pour la connexion d'un utilisateur.

## 12.2. La commande su

**su** signifie **Substitute User** ou **Switch User**. Elle permet d'endosser l'identité d'un autre utilisateur sans se déconnecter. Cette commande utilisée sans login permet par défaut de prendre l'identité de **root**.

**Syntaxe de la commande su.**

```
su [-] [-c commande] [login]
```

Exemple :

```
[root]# su - alain  
[albert]$ su -c "passwd alain"
```

Option	Description
-	Charge l'environnement complet de l'utilisateur.
-c commande	Exécute la commande sous l'identité de l'utilisateur.

Pour la sécurité, les accès par la commande **su** sont répertoriés dans le fichier journal **/var/log/secure**.

Les utilisateurs non **root** devront taper le mot de passe de la nouvelle identité.



Il y a création de couches successives. Pour passer d'un utilisateur à un autre, il faut d'abord taper la commande **exit** pour reprendre son identité puis la commande **su** pour prendre une autre identité.

Il est conseillé de restreindre l'accès à la commande su à certaines personnes définies explicitement. Pour cela, il faut donner les droits à un groupe particulier pour accéder à la commande. Toutes les personnes appartenant à ce groupe obtiendront les droits réservés de celui-ci.



Le principe de restriction mis en place consiste à autoriser seulement un groupe d'utilisateurs particulier à exécuter la commande su.

### 12.3. La commande sudo

La commande **sudo** permet d'exécuter une ou plusieurs commandes avec les priviléges de root. Il n'est pas nécessaire de connaître son mot de passe.

#### Syntaxe de la commande sudo.

```
sudo [-options] commande
```

Cet outil représente une évolution de la commande su.

Exemple :

```
[alain]$ sudo /sbin/route
Mot de passe : *****
Table de routage IP du noyau
Destination   Passerelle   Genmask     Indic Metric Ref Use Iface ....
```

Option	Description
-E	Garde l'environnement du compte appelant.
-u	Désigne le compte qui exécutera la commande.



L'environnement est constitué des paramètres de l'utilisateur comme son répertoire de connexion, ses alias ...

Les accès possibles à la commande et les possibilités offertes aux utilisateurs sont configurables par l'administrateur qui leur affecte des droits explicites.

Vérification de l'existence du paquet :

```
[root] # rpm -qa "sudo*"
sudo-1.8.6p3-12.el6.i686
```

### 12.3.1. Avantages

Par rapport à l'utilisation des restrictions des groupes :

- Le contrôle d'accès aux commandes est centralisé dans le fichier **/etc/sudoers** ;
- Tous les accès et tentatives à partir de sudo sont journalisés dans le fichier **/var/log/secure**;
- L'utilisateur doit s'identifier à sudo en donnant son mot de passe.

Le fichier **/etc/sudoers** contient pour chacun des utilisateurs :

- son login ;
- sur quel ordinateur il est autorisé ;
- quelles commandes il peut exécuter.

## 12.4. Commande visudo

La commande visudo permet d'éditionner le fichier **/etc/sudoers** (qui est en lecture seule) afin de configurer efficacement l'accès à la commande sudo pour les utilisateurs.

### Syntaxe de la commande visudo.

```
visudo [-c] [-f sudoers]
```

Exemple :

```
[root]# visudo
```

Option	Description
-C	Vérifie la syntaxe du fichier.
-f file	Désigne le fichier de configuration de sudo.

### 12.4.1. Le fichier /etc/sudoers

```
# This file must be edited with the
# 'visudo' command.

#
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
```

### 12.4.2. Le groupe wheel

Le mot **wheel** fait référence à un groupe système qui dispose de priviléges lui permettant l'exécution de commandes à accès restreint.

Le groupe **wheel** est présent par défaut sur les distributions RHEL/CentOS.

### 12.4.3. Ne plus utiliser root

1 - Autoriser les utilisateurs du groupe **wheel** à lancer toutes les commandes (via visudo) en supprimant le commentaire de la ligne suivante :

```
%wheel    ALL=(ALL)    ALL
```

2 - Ajouter le compte utilisateur **bob** dans le groupe **wheel** :

```
[root]# usermod -aG wheel bob
```

3 - L'utilisateur **bob** dispose maintenant des pleins pouvoirs :

```
[bob]$ sudo chown root:root /tmp/test  
[sudo] password for bob:
```



Il n'existe plus de raisons valables d'utiliser le compte root.

Le mot de passe **root** peut être séquestré (Keepass !) et son accès via ssh verrouillé.

#### **12.4.4. Restreindre le sudo**

Il est possible de ne pas donner tous les droits à un utilisateur mais de limiter ses accès et les commandes qu'il peut lancer. Par exemple, n'offrir à un compte de supervision que le droit de relancer le serveur (reboot) et uniquement celui-là.

Les accès à l'utilisation de sudo sont enregistrés dans les logs ce qui offre une traçabilité des actions entreprises.

**Les restrictions se gèrent dans les différentes sections du fichier /etc/sudoers par l'intermédiaire de la commande visudo.**

#### **Section Host aliases**

Définir des groupes de machines ou réseaux.

```
Host_Alias HOSTS-GROUP = host1 [,host2 ...]
```

Exemple :

```
Host_Alias FILESERVERS = 192.168.1.1, SF1, SF2
```

Cette section est utilisée pour créer des groupes de ressources réseaux autorisés à utiliser sudo.

#### **Section User aliases**

Définir des alias pour les utilisateurs.

```
User_Alias USER-GROUP = alain [,philippe, ...]
```

Exemple :

```
User_Alias ADMINS = root, AdminSF, adminPrinters
```

Cette section est utilisée essentiellement pour réunir sous un alias unique plusieurs utilisateurs ayant les mêmes besoins de la commande sudo.

## ***Section Command aliases***

Définir des alias pour les commandes.

```
Cmnd_Alias CDE-GROUP = cde1 [,cde2, ...]
```

Exemple :

```
Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/yum  
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig
```

Cette section est utilisée pour réunir plusieurs commandes Linux dans un groupe de commandes sudo.

Il faut alors créer des groupes cohérents.

## ***Section User Specification***

Lier les utilisateurs aux commandes.

```
USER-GROUP HOSTS-GROUP = [(cptc-cible)] CDE-GROUP
```

Exemple :

```
root      ALL=(ALL)          ALL  
AdminSF   FILESERVERS=     SOFTWARE
```

Cette section définit qui a le droit d'utiliser des commandes particulières à partir de postes particuliers.

Il est possible de préciser qui exécute la commande (compte cible).

### **Exemple 1 :**

Grâce au fichier **/etc/sudoers** ci-dessous, les utilisateurs alain, patrick et philippe peuvent désormais exécuter les commandes **ping** et **route** et effectuer des transferts FTP comme s'ils étaient **root**.

```
# Host alias specification
Host_Alias STA = 192.168.1.1, ma.machine
# User alias specification
User_Alias CPTUSER = alain, patrick, philippe
# Cmnd alias specification
Cmnd_Alias NET = /bin/ping, /sbin/route, /usr/bin/ftp
# User privilege specification
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
CPTUSER STA=(root) NET
```

### **Exemple 2 :**

```
# Host alias specification
Host_Alias MACHINE = station1
# User alias specification
User_Alias ADMIN = adminunix
User_Alias UTILISAT = alain, philippe
# Cmnd alias specification
Cmnd_Alias SHUTDOWN = /sbin/shutdown
Cmnd_Alias NET = /usr/bin/ftp
# User privilege specification
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
ADMIN MACHINE=NOPASSWD:ALL
UTILISAT MACHINE=(root) SHUTDOWN, NET
```

Explications de l'exemple 2 :

- Création d'un alias pour la station :

```
Host_Alias MACHINE=station1
```

- Création de deux alias pour deux types d'utilisateurs (adminunix étant un équivalent « root »).

```
User_Alias ADMIN = adminunix  
User_Alias UTILISAT = alain, philippe
```

- Création de deux alias de commandes qui regroupent les commandes exécutables.

```
Cmnd_Alias SHUTDOWN = /sbin/shutdown  
Cmnd_Alias NET = /usr/bin/ftp
```

- L'utilisateur « root » peut exécuter toutes les commandes sur toutes les machines

```
root ALL=(ALL) ALL
```

Les utilisateurs qui font partie de l'alias ADMIN peuvent exécuter toutes les commandes, sur toutes les machines faisant partie de l'alias MACHINE et ce sans entrer de mot de passe (NOPASSWD:).

```
ADMIN MACHINE=NOPASSWD:ALL
```

Les utilisateurs qui font partie de l'alias UTILISAT peuvent exécuter la commande /sbin/shutdown sur toutes les machines faisant partie de l'alias MACHINE. Ils doivent entrer leur mot de passe.

```
UTILISAT MACHINE = SHUTDOWN, NET
```



# 13

## Les modules d'authentification PAM

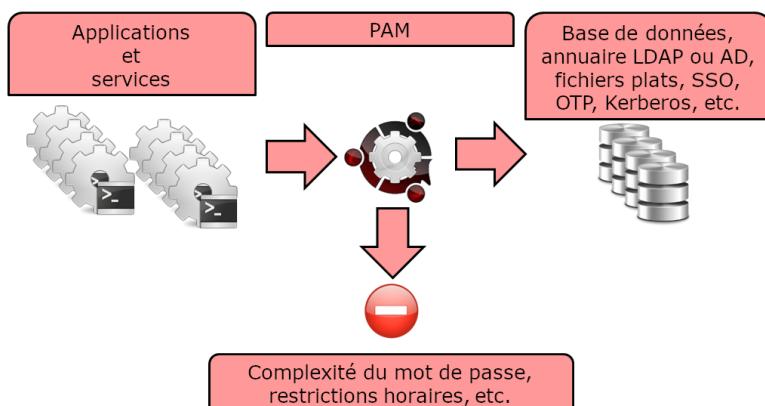
PAM (**Pluggable Authentication Modules**) est le système sous GNU/Linux qui permet à de nombreuses applications ou services d'**authentifier** les utilisateurs de manière **centralisée**.

PAM est un mécanisme permettant d'intégrer différents schémas d'authentification de bas niveau dans une API de haut niveau, permettant de ce fait de rendre indépendants du schéma les logiciels réclamant une authentification.

— Définition de PAM par Wikipedia

### 13.1. Généralités

L'authentification est la phase durant laquelle il est vérifié que vous êtes bien la personne que vous prétendez être. Il existe d'ailleurs d'autres formes d'authentification que l'utilisation des mots de passe.



La mise en place d'une nouvelle méthode d'authentification ne doit pas nécessiter de modifications dans la configuration ou dans le code source d'un programme ou d'un service.

C'est pourquoi les applications s'appuient sur PAM, qui va leur fournir les primitives nécessaires à l'authentification de leurs utilisateurs.

L'ensemble des applications d'un système peuvent ainsi mettre en œuvre en toute transparence des fonctionnalités complexes comme le **SSO** (Single Sign On), l'**OTP** (One Time Password) ou **Kerberos** de manière totalement transparente.

L'administrateur d'un système peut choisir exactement sa politique d'authentification pour une seule application (par exemple pour durcir le service SSH) indépendamment de celle-ci.

À chaque application ou service prenant en charge PAM correspondra un fichier de configuration dans le répertoire «/etc/pam.d». Par exemple, le processus «login» attribue le nom «/etc/pam.d/login» à son fichier de configuration.



Une mauvaise configuration de PAM peut compromettre toute la sécurité de votre système.

PAM est un système d'authentification (gestion des mots de passe). Si PAM est vulnérable alors l'ensemble du système est vulnérable.

### **13.1.1. Syntaxe d'une directive**

Une directive permet de paramétriser une application pour PAM.

#### **Syntaxe d'une directive.**

```
mécanisme [contrôle] chemin-module [argument]
```

Par exemple :

#### **Le fichier /etc/pam.d/sudo.**

```
#%PAM-1.0
auth      include  system-auth
account   include  system-auth
password  include  system-auth
```

```
session optional pam_keyinit.so revoke
session required pam_limits.so
```

Une **directive** (une ligne complète) est composée d'un **mécanisme** (auth, account, password ou session), d'un **contrôle de réussite** (include, optional, required, ...), du chemin d'accès au module et éventuellement d'arguments (comme revoke par exemple).



L'ordre des modules est très important !

Chaque fichier de configuration PAM comprend un ensemble de directives. Les directives des interfaces de modules peuvent être empilées ou placées les unes sur les autres.

De fait, l'ordre dans lequel les modules sont répertoriés est très important au niveau du processus d'authentification.

## 13.2. Les mécanismes

### 13.2.1. Le mécanisme auth - Authentification

Concerne l'authentification du demandeur et établit les droits du compte :

- Authentifie généralement avec un mot de passe en le comparant à une valeur stockée en base de données ou en s'appuyant sur un serveur d'authentification,
- Établit les paramètres du comptes : uid, gid, groupes et limites de ressources.

### 13.2.2. Le mécanisme account - Gestion de compte

Vérifier que le compte demandé est disponible :

- Concerne la disponibilité du compte pour des raisons autres que l'authentification (par exemple pour des restrictions d'horaire).

### 13.2.3. Le mécanisme session - Gestion de session

Concerne la mise en place et la terminaison de la session :

- Accomplir les tâches associées à la mise en place d'une session (par exemple enregistrement dans les logs),
- Accomplir les tâches associées à la terminaison d'une session.

#### **13.2.4. Le mécanisme password - Gestion des mots de passe**

Utilisé pour modifier le jeton d'authentification associé à un compte (expiration ou changement) :

- Modifie le jeton d'authentification et vérifie éventuellement qu'il est assez robuste ou qu'il n'a pas déjà été utilisé.

### **13.3. Les indicateurs de contrôle**

Les **mécanismes PAM** (**auth**, **account**, **session** et **password**) indiquent la réussite ou l'échec. Les **indicateurs de contrôle** (**required**, **requisite**, **sufficient**, **optional**) indiquent à PAM comment traiter ce résultat.

#### **13.3.1. L'indicateur de contrôle required**

La réussite de tous les modules **required** est nécessaire.

- Si le module réussit :

Le reste de la chaîne est exécuté. La requête est autorisée sauf si d'autres modules échouent.

- Si le module échoue :

Le reste de la chaîne est exécuté. Au final la requête est rejetée.

Le module doit être vérifié avec succès pour que l'authentification puisse se poursuivre. Si la vérification d'un module portant l'indication required échoue, l'utilisateur n'en est pas averti tant que tous les modules associés à cette interface n'ont pas été vérifiés.

#### **13.3.2. L'indicateur de contrôle requisite**

La réussite de tous les modules **requisite** est nécessaire.

- Si le module réussit :

Le reste de la chaîne est exécuté. La requête est autorisée sauf si d'autres modules échouent.

- Si le module échoue :

La requête est immédiatement rejetée.

Le module doit être vérifié avec succès pour que l'authentification puisse se poursuivre. Cependant, si la vérification d'un module requisite échoue, l'utilisateur en est averti immédiatement par le biais d'un message lui indiquant l'échec du premier module required ou requisite.

### ***13.3.3. L'indicateur de contrôle sufficient***

La réussite d'un seul module **sufficient** est suffisant.

- Si le module réussit :

La requête est immédiatement autorisée si aucun des précédents modules n'a échoué.

- Si le module échoue :

Le module est ignoré. Le reste de la chaîne est exécutée.

En cas d'échec, les vérifications de modules sont ignorées. Toutefois, si la vérification d'un module portant l'indication sufficient est réussie et qu'aucun module précédent portant l'indicateur required ou requisite n'a échoué, aucun autre module de ce type n'est nécessaire et l'utilisateur sera authentifié auprès du service.

### ***13.3.4. L'indicateur de contrôle optional***

Le module est exécuté mais le résultat de la requête est ignoré.

Si tous les modules de la chaînes étaient marqués optional, toutes les requêtes seraient toujours acceptées.

### 13.3.5. En conclusion

gear	required	X	✓	✓	✓
user	required	✓	✓	✓	✓
gear	requisite	✓	X	✓	✓
user	optional	✓	✓	X	X
gear	sufficient	✓	✓	✓	X
user	requisite			✓	✓

Résultats : X X ✓ ✓

## 13.4. Les modules de PAM

Il existe de nombreux **modules** pour PAM. Voici les plus fréquents :

- pam\_unix
- pam\_ldap
- pam\_wheel
- pam\_cracklib
- pam\_console
- pam\_tally
- pam\_securetty
- pam\_nologin
- pam\_limits
- pam\_time
- pam\_access

### 13.4.1. Le module pam\_unix

Le module **pam\_unix** permet de gérer la politique globale d'authentification.

Fichier **/etc/pam.d/system-auth**.

```
password sufficient pam_unix.so sha512 nullok
```

Des arguments sont possibles pour ce module :

- **nullok** : dans le mécanisme auth autorise un mot de passe de connexion vide.
- **sha512** : dans le mécanisme password, définit l'algorithme de cryptage.
- **debug** : pour transmettre les informations à "syslog".
- **remember=n** : pour se souvenir des n derniers mots de passe utilisés (fonctionne conjointement avec le fichier "/etc/security/opasswd", qui est à créer par l'administrateur).

### 13.4.2. Le module pam\_cracklib

Le module **pam\_cracklib** permet de tester les mots de passe.

Fichier **/etc/pam.d/password-auth**.

```
password sufficient pam_cracklib.so retry=2
```

Ce module utilise la bibliothèque **cracklib** pour vérifier la solidité d'un nouveau mot de passe. Il peut également vérifier que le nouveau mot de passe n'est pas construit à partir de l'ancien. Il ne concerne que le mécanisme password.

Par défaut ce module vérifie les aspects suivants et rejette si tel est le cas :

- le nouveau mot de passe est-il issu du dictionnaire ?
- le nouveau mot de passe est-il un palindrome de l'ancien (ex : azerty <> ytreza) ?
- seule la casse de(s) caractère(s) varie (ex : azerty <>AzErTy) ?

Des arguments possibles pour ce module :

- **retry=n** : impose n demandes (1 par défaut) du nouveau mot de passe.
- **difok=n** : impose au minimum n caractères (10 par défaut) différents de l'ancien mot de passe. De plus si la moitié des caractères du nouveau différent de l'ancien, le nouveau mot de passe est validé.
- **minlen=n** : impose un mot de passe de n+1 caractères minimum non pris en compte en dessous de 6 caractères (module compilé comme tel !).

Autres arguments possibles :

- **dcredit=-n** : impose un mot de passe contenant au moins n chiffres,
- **ucredit=-n** : impose un mot de passe contenant au moins n majuscules,
- **credit=-n** : impose un mot de passe contenant au moins n minuscules,
- **ocredit=-n** : impose un mot de passe contenant au moins n caractères spéciaux.

### **13.4.3. Le module pam\_tally**

Le module **pam\_tally** permet de verrouiller un compte en fonction d'un nombre de tentatives infructueuses de connexion.

**Fichier /etc/pam.d/system-auth.**

```
auth required /lib/security/pam_tally.so onerr=fail no_magic_root
account required /lib/security/pam_tally.so deny=3 reset no_magic_root
```

- Le mécanisme **account** incrémente le compteur.
- Le mécanisme **auth** accepte ou refuse l'authentification et réinitialise le compteur.

Quelques arguments du module pam\_tally sont intéressants à utiliser :

- **onerr=fail** : incrémentation du compteur,
- **deny=n** : une fois le nombre n d'essais infructueux dépassé, le compte est verrouillé,
- **no\_magic\_root** : inclus ou non les démons gérés par root (éviter le verrouillage de root),
- **reset** : remet le compteur à 0 si l'authentification est validée,
- **lock\_time=nsec** : le compte est verrouillé pour n secondes.

Ce module fonctionne conjointement avec le fichier par défaut des essais infructueux */var/log/faillog* (qui peut être remplacé par un autre fichier avec l'argument *file=xxxx*) et la commande associée *faillog*.

**Syntaxe de la commande faillog.**

```
faillog[-m n] [-u login][-r]
```

Options :

- m : pour définir, dans l'affichage de la commande, le nombre maximum d'essais infructueux,
- u : pour spécifier un utilisateur,
- r : déverrouiller un utilisateur.

#### **13.4.4. Le module pam\_time**

Le module **pam\_time** permet de limiter les horaires d'accès à des services gérés par PAM.

**Fichier /etc/pam.d/system-auth.**

```
account required /lib/security/pam_time.so
```

La configuration se fait via le fichier /etc/security/time.conf.

**Fichier /etc/security/time.conf.**

```
login ; * ; users ; MoTuWeThFr0800-2000
http ; * ; users ; A10000-2400
```

La syntaxe d'une directive est la suivante :

```
services ; ttys ; users ; times
```

Dans les définitions suivantes, la liste logique utilise :

- & : et logique,
- | : ou logique,
- ! : négation = « tous sauf »,
- \* : caractère « joker ».

Les colonnes correspondent à :

- **services** : liste logique de services gérés par PAM qui sont concernés,

- **tty**s : liste logique de périphériques concernés,
- **users** : liste logique d'utilisateurs gérés par la règle,
- **times** : liste logique de détermination de l'horaire (jour/plage) autorisé.

Comment gérer les créneaux horaires :

- **les jours** : Mo Tu We Th Fr Sa Su Wk (du L au V) Wd (S et D) Al (du L au D),
- **la plage** : HHMM- HHMM,
- **une répétition annule l'effet** : WkMo = toutes les jours de la semaine (L-V) moins le lundi (répétition).

Exemples :

- Bob, peut se connecter via un terminal tous les jours entre 07h00 et 09h00, sauf le mercredi :

```
login ; tty* ; bob ; alth0700-0900
```

- Pas d'ouvertures de sessions, terminal ou distantes, sauf root, tous les jours de la semaine entre 17h30 et 7h45 le lendemain :

```
login ; tty* | pts/* ; !root ; !wk1730-0745
```

#### **13.4.5. Le module pam\_nologin**

Le module **pam\_nologin** permet de désactiver tous les comptes sauf root :

Fichier **/etc/pam.d/login** :

```
auth required pam_nologin.so
```

Si le fichier **/etc/nologin** existe alors seul root pourra se connecter.

#### **13.4.6. Le module pam\_wheel**

Le module **pam\_wheel** permet de limiter l'accès à la commande su aux membres du groupes wheel.

### Fichier /etc/pam.d/su.

```
auth required pam_wheel.so
```

L'argument **group=mon\_group** limite l'usage de la commande su aux membres du groupe mon\_groupe



Si le groupe mon\_groupe est vide, alors la commande su n'est plus disponible sur le système, ce qui force l'utilisation de la commande sudo.

### 13.4.7. Le module pam\_mount

Le module **pam\_mount** permet de monter un volume pour une session utilisateur.

#### Fichier /etc/pam.d/system-auth.

```
auth optional pam_mount.so
password optional pam_mount.so
session optional pam_mount.so
```

Les points de montage sont configurés dans le fichier /etc/security/pam\_mount.conf :

#### Fichier /etc/security/pam\_mount.conf.

```
<volume fstype="nfs" server="srv" path="/home/%(USER)" mountpoint="~" />
<volume user="bob" fstype="smbfs" server="filesrv" path="public"
mountpoint="/public" />
```



---

# 14

## Securisation SELinux

---

Avec l'arrivée du noyau en version 2.6, un nouveau système de sécurité a été introduit pour fournir un mécanisme de sécurité supportant les stratégies de sécurité de contrôle d'accès.

Ce système s'appelle SELinux (Security Enhanced Linux) et a été créé par la NSA (National Security Administration) pour implémenter dans les sous-systèmes du noyau Linux une architecture robuste de type Mandatory Access Control (MAC).

Si, tout au long de votre carrière, vous avez soit désactivé ou ignoré SELinux, ce chapitre sera pour vous une bonne introduction à ce système qui travaille dans l'ombre de Linux pour limiter les priviléges ou supprimer les risques liés à la compromission d'un programme ou d'un démon.

Avant de débuter, sachez que SELinux est essentiellement à destination des distributions RHEL, bien qu'il soit possible de le mettre en œuvre sur d'autres distributions comme Debian (mais bon courage !). Les distributions de la famille Debian intègrent généralement le système AppArmor, qui pour sa part, fonctionne relativement différemment de SELinux.

### 14.1. Généralités

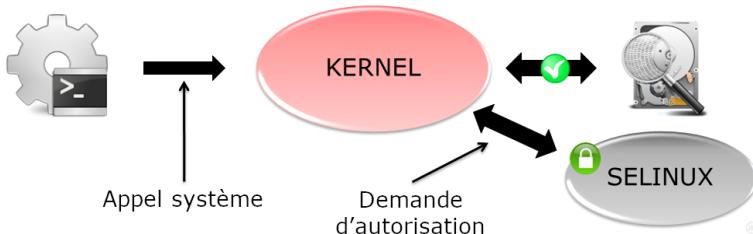
SELinux (Security Enhanced Linux ou Amélioration de la Sécurité Linux) est un système de contrôle d'accès obligatoire (Mandatory Access Control).

Avant l'apparition des systèmes MAC, la sécurité standard de gestion d'accès reposait sur des systèmes DAC (Discretionary Access Control). Une application, ou un démon, fonctionnait avec des droits UID ou SUID (Set Owner User Id), ce

qui permettait d'évaluer les permissions (sur les fichiers, les sockets, et autres processus...) en fonction de cet utilisateur. Ce fonctionnement ne permet pas de limiter suffisamment les droits d'un programme qui est corrompu, ce qui lui permet potentiellement d'accéder aux sous-systèmes du système d'exploitation.

Un système MAC renforce la séparation entre les informations sur la confidentialité et l'intégrité du système pour obtenir un système de confinement. Le système de confinement est indépendant du système de droits traditionnels et il n'existe pas de notion de superutilisateur.

À chaque appel système, le noyau interroge SELinux pour savoir s'il autorise l'action à être effectuée.



SELinux utilise pour cela un ensemble de règles (en anglais *policy*). Un ensemble de deux jeux de règles standards (*targeted* et *strict*) est fourni et chaque application fournit généralement ses propres règles.

### **14.1.1. Le contexte SELinux**

Le fonctionnement de SELinux est totalement différent des droits traditionnels Unix.

Le contexte de sécurité SELinux est défini par le trio **identité+rôle+domaine**.

L'identité d'un utilisateur dépend directement de son compte linux. Une identité se voit attribué un ou plusieurs rôles, mais à chaque rôle correspond un domaine et un seul. C'est en fonction du domaine du contexte de sécurité (et donc du rôle...) que sont évalués les droits d'un utilisateur sur une ressource.

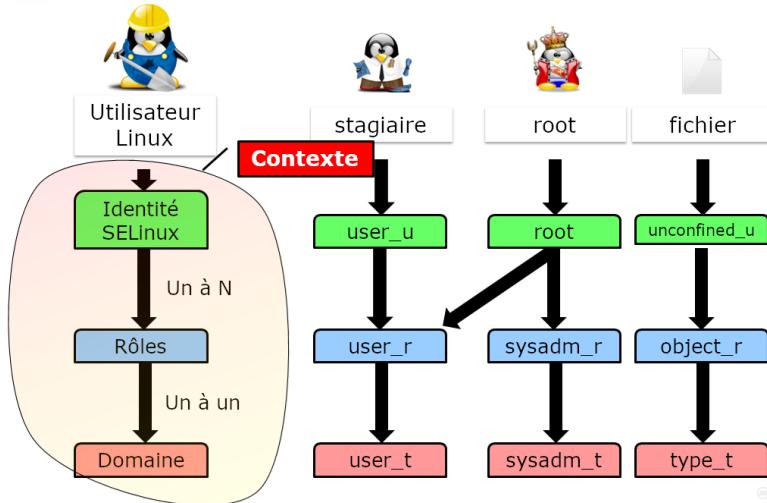


Figure 14.1. Le contexte SELinux

Les termes "domaine" et "type" sont similaires, typiquement "domaine" est utilisé lorsque l'on se réfère à un processus tandis que "type" réfère à un objet.

La convention de nommage est : \_u user ; \_r rôle ; \_t type.

Le contexte de sécurité est attribué à un utilisateur au moment de sa connexion, en fonction de ses rôles. Le contexte de sécurité d'un fichier est quant à lui défini par la commande **chcon** (change context) que nous verrons plus tard dans la suite de ce chapitre.

Considérez les pièces suivantes du puzzle SELinux :

- Les sujets
- Les objets
- Les stratégies
- Le mode

Quand un sujet (une application par exemple) tente d'accéder à un objet (un fichier par exemple), la partie SELinux du noyau Linux interroge sa base de données de stratégies. En fonction du mode de fonctionnement, SELinux autorise l'accès à l'objet en cas de succès, sinon il enregistre l'échec dans le fichier /var/log/messages.

## Le contexte SELinux des processus standards

Les droits d'un processus dépendent de son contexte de sécurité.

Par défaut, le contexte de sécurité du processus est défini par le contexte de l'utilisateur (identité + rôle + domaine) qui le lance.

Un domaine étant un type (au sens SELinux) spécifique lié à un processus et hérité (normalement) de l'utilisateur qui l'a lancé, ses droits s'expriment en termes d'autorisation ou de refus sur des types (liés à des objets) :

Un processus dont le contexte a le domaine de sécurité D peut accéder aux objets de type T.

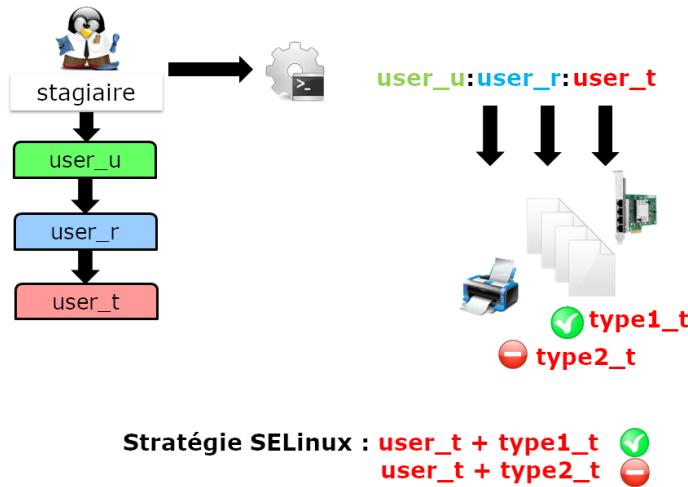


Figure 14.2. Le contexte SELinux d'un processus standard

## Le contexte SELinux des processus importants

La plupart des programmes importants se voient attribuer un domaine dédié.

Chaque exécutable est étiqueté avec un type dédié (ici `sshd_exec_t`) qui fait basculer le processus associé automatiquement dans le contexte `sshd_t` (au lieu de `user_t`).

Ce mécanisme est essentiel puisqu'il permet de restreindre au plus juste les droits d'un processus.

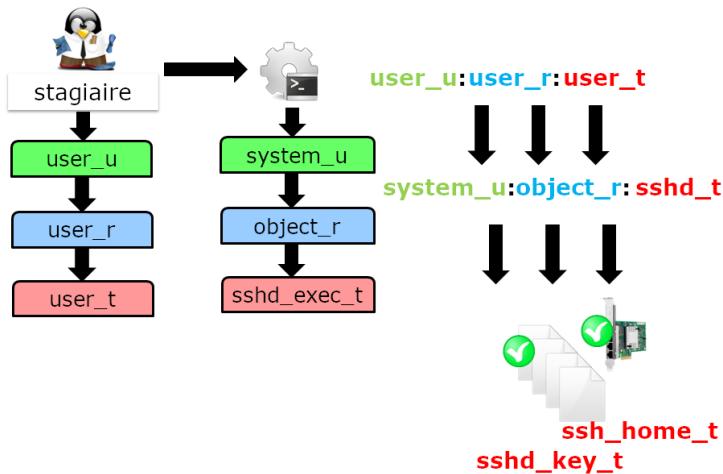


Figure 14.3. Le contexte SELinux d'un processus important - exemple de sshd

## 14.2. Gestion

La commande **semanage** (SE manage) permet d'administrer les règles SELinux.

### Syntaxe de la commande semanage.

```
semanage [type_d_objet] [options]
```

Exemple :

```
[root]# semanage boolean -l
```

Tableau 14.1. Options de la commande semanage

Options	Observations
-a	Ajoute un objet
-d	Supprime un objet
-m	Modifie un objet
-l	Liste les objets

La commande **semanage** n'est pas installée par défaut sous CentOS.

Sans connaître le paquet qui fournit cette commande, il convient de rechercher son nom avec la commande :

```
[root]# yum provides */semanage
```

puis l'installer :

```
[root]# yum install policycoreutils-python
```

### **14.2.1. Administrer les objets de type booléens**

Les booléens permettent le confinement des processus.

#### **Syntaxe de la commande semanage boolean.**

```
semanage boolean [options]
```

Pour lister les booléens disponibles :

```
[root]# semanage boolean -l
Booléen SELinux      State Default  Description
...
httpd_can_sendmail (fermé, fermé)  Allow http
daemon to send mail
...
```

La commande **setsebool** permet de modifier l'état d'un objet de type booléen :

#### **Syntaxe de la commande setsebool.**

```
setsebool [-PV] boolean on|off
```

Exemple :

```
[root]# setsebool -P httpd_can_sendmail on
```

Tableau 14.2. Options de la commande setsebool

Options	Observations
-P	Modifie la valeur par défaut au démarrage (sinon uniquement jusqu'au reboot)

Options	Observations
-V	Supprime un objet

La commande semanage permet d'administrer les objets de type port :

#### Syntaxe de la commande semanage port.

```
semanage port [options]
```

Exemple : autoriser le port 81 aux processus du domaine httpd

```
[root]# semanage port -a -t http_port_t -p tcp 81
```

### 14.3. Mode de fonctionnement

SELinux propose trois modes de fonctionnement :

- Enforcing (Appliqué)

Mode par défaut pour les Linux Redhat. Les accès seront restreints en fonction des règles en vigueur.

- Permissive (Permissif)

Les règles sont interrogées, les erreurs d'accès sont journalisées, mais l'accès ne sera pas bloqué.

- Disabled (Désactivé)

Rien ne sera restreint, rien ne sera journalisé.

Par défaut, la plupart des systèmes d'exploitation sont configurés avec SELinux en mode Enforcing.

La commande **getenforce** retourne le mode de fonctionnement en cours

#### Syntaxe de la commande getenforce.

```
getenforce
```

Exemple :

```
[root]# getenforce  
Enforcing
```

La commande **sestatus** retourne des informations sur SELinux

**Syntaxe de la commande sestatus.**

```
sestatus
```

Exemple :

```
[root]# sestatus  
SELinux status:     enabled  
SELinuxfs mount:   /selinux  
Current mode:      enforcing  
Mode from config file : enforcing  
Policy version:    24  
Policy from config file: targeted
```

La commande **setenforce** modifie le mode de fonctionnement en cours :

**Syntaxe de la commande setenforce.**

```
setenforce 0|1
```

Passer SELinux en mode permissif :

```
[root]# setenforce 0
```

#### **14.3.1. Le fichier /etc/sysconfig/selinux**

Le fichier /etc/sysconfig/selinux permet de modifier le mode de fonctionnement de SELinux.



Désactiver SELinux se fait à vos risques et périls ! Il est préférable d'apprendre le fonctionnement de SELinux plutôt que de le désactiver systématiquement !

Modifier le fichier /etc/sysconfig/selinux

```
SELINUX=disabled
```

Redémarrer le système :

```
[root]# reboot
```



Attention au changement de mode SELinux !

En mode permissif ou désactivé, les nouveaux fichiers créés ne porteront aucune étiquette.

Pour réactiver SELinux, il faudra repositionner les étiquettes sur l'intégralité de votre système.

Labéliser le système entièrement :

```
[root]# touch /.autorelabel  
[root]# reboot
```

#### 14.4. Les jeux de règles (Policy Type) :

SELinux fournit deux types de règles standards :

- Targeted : seuls les démons réseaux sont protégés (dhcpd, httpd, named, nscd, ntpd, portmap, snmpd, squid et syslogd)
- Strict : tous les démons sont protégés

#### 14.5. Contexte

L'affichage des contextes de sécurité se fait avec l'option -Z. Elle est associée à de nombreuses commandes :

Exemples :

```
[root]# id -Z # le contexte de l'utilisateur  
[root]# ls -Z # ceux des fichiers courants  
[root]# ps -eZ # ceux des processus
```

```
[root]# netstat -Z # ceux des connexions réseaux
[root]# lsof -Z # ceux des fichiers ouverts
```

La commande **matchpathcon** retourne le contexte d'un répertoire.

### Syntaxe de la commande matchpathcon.

```
matchpathcon répertoire
```

Exemple :

```
[root]# matchpathcon /root
/root system_u:object_r:admin_home_t:s0

[root]# matchpathcon /
/ system_u:object_r:root_t:s0
```

La commande **chcon** modifie un contexte de sécurité :

### Syntaxe de la commande chcon.

```
chcon [-vR] [-u USER] [-r ROLE] [-t TYPE] fichier
```

Exemple :

```
[root]# chcon -vR -t httpd_sys_content_t /home/SiteWeb
```

Tableau 14.3. Options de la commande chcon

Options	Observations
-v	Passe en mode verbeux
-R	Applique la récursivité
-u,-r,-t	S'applique à un utilisateur, un rôle ou un type

La commande **restorecon** restaure le contexte de sécurité par défaut :

### Syntaxe de la commande restorecon.

```
restorecon [-vR] répertoire
```

Exemple :

```
[root]# restorecon -vR /home/SiteWeb
```

Tableau 14.4. Options de la commande restorecon

Options	Observations
-v	Passe en mode verbeux
-R	Applique la récursivité

La commande **audit2why** indique la cause d'un refus SELinux :

#### Syntaxe de la commande audit2why.

```
audit2why [-vw]
```

Exemple :

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1
audit2why
```

Tableau 14.5. Options de la commande audit2why

Options	Observations
-v	Passe en mode verbeux
-w	Traduit la cause d'un rejet par SELinux et propose une solution pour y remédier (option par défaut)

### 14.5.1. Aller plus loin avec SELinux

La commande **audit2allow** crée à partir d'une ligne d'un fichier "audit" un module pour autoriser une action SELinux :

#### Syntaxe de la commande audit2allow.

```
audit2allow [-mM]
```

Exemple :

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1
audit2allow -M MonModule_Local
```

**Tableau 14.6. Options de la commande audit2allow**

Options	Observations
-m	Crée juste le module (*.te)
-M	Crée le module, le compile et le met en paquet (*.pp)

## ***Exemple de configuration***

Après l'exécution d'une commande, le système vous rend la main mais le résultat attendu n'est pas visible : aucun message d'erreur à l'écran.

- **Étape 1** : Lire le fichier journal sachant que le message qui nous intéresse est de type AVC (SELinux), refusé (denied) et le plus récent (donc le dernier).

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1
```

Le message est correctement isolé mais ne nous est d'aucune aide.

- **Étape 2** : Lire le message isolé avec la commande audit2why pour obtenir un message plus explicite pouvant contenir la solution de notre problème (typiquement un booléen à positionner).

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2why
```

Deux cas se présentent : soit nous pouvons placer un contexte ou renseigner un booléen, soit il faut passer à l'étape 3 pour créer notre propre contexte.

- **Étape 3** : Créer son propre module.

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2allow -M MonModule_Local
Generating type enforcement: MonModule_Local.te
Compiling policy: checkmodule -M -m -o MonModule_Local.mod
MonModule_Local.te
Building package: semodule_package -o MonModule_Local.pp -m
MonModule_Local.mod

[root]# semodule -i MonModule_Local.pp
```

---

# 15

## IPTables, le parefeu Linux

---

Gérer le trafic réseau est certainement la partie la plus difficile du métier d'administrateur système. Il faut impérativement configurer les firewalls sur l'ensemble des éléments actifs du réseau, serveurs inclus, en prenant en compte les besoins des utilisateurs et des systèmes à la fois pour le trafic entrant et sortant, sans laisser des systèmes vulnérables à des attaques.

C'est le rôle du pare-feu **IPTables**, entièrement administrable en ligne de commandes.

IPTables utilise un jeu de tables qui contiennent des chaînes comprenant les règles du firewall.

Il existe 3 types de tables :

- La table **FILTER**, qui est la table par défaut. Elle est composée de 3 chaînes :
  - La chaîne **INPUT** : les paquets sont destinés à une socket locale.
  - La chaîne **FORWARD** : les paquets sont routés par le système.
  - La chaîne **OUTPUT** : les paquets sont générés en local.
- La table **NAT**, qui est consultée lorsque un paquet tente de créer une nouvelle connexion. Elle est composée de 3 chaînes :
  - **PREROUTING** : utilisée pour modifier des paquets dès leur réception par le système.
  - **OUTPUT** : utilisée pour modifier des paquets générés en local.

- **POSTROUTING** : utilisée pour modifier des paquets au moment de leur sortie du système.
- La table **MANGLE** : cette table est utilisée pour modifier des paquets. Il y a 5 chaînes :
  - **PREROUTING** : pour modifier des connexions entrantes.
  - **OUTPUT** : pour modifier des paquets générés en local.
  - **INPUT** : pour modifier les paquets entrants.
  - **POSTROUTING** : pour modifier les paquets avant leur départ du système.
  - **FORWARD** : pour modifier les paquets qui sont routés par le système.

## 15.1. Gestion du pare-feu

### 15.1.1. Démarrer/arrêter ou redémarrer le parefeu

En fonction de la distribution, si elle utilise les scripts de démarrage SystemD (RHEL 7 et +) ou SysVinit, il faudra utiliser les lignes de commandes suivantes :

#### Distribution basée sur SystemD.

```
[admin]$ sudo systemctl start iptables
[admin]$ sudo systemctl stop iptables
[admin]$ sudo systemctl restart iptables
```

#### Distribution basée sur SysVinit.

```
[admin]$ sudo service iptables start
[admin]$ sudo service iptables stop
[admin]$ sudo service iptables restart
```

### 15.1.2. Afficher les règles

Pour afficher les règles IPTables :

```
[admin]$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 927K packets, 219M bytes)
pkts bytes target prot opt in out source destination
...
```

```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
...
```

```
Chain OUTPUT (policy ACCEPT 127 packets, 18M bytes)
pkts bytes target prot opt in out source destination
...
```

Il est possible de spécifier, avec l'option **-t**, quelle table afficher :

```
[admin]$ sudo iptables -t input -L -n -v
Chain INPUT (policy ACCEPT 927K packets, 219M bytes)
pkts bytes target prot opt in out source destination
...
```

## 15.2. Quelques exemples

Voici quelques règles qui mettent en oeuvre les fonctionnalités d'IPTables :

### 15.2.1. Bloquer des adresses IP spécifiques

Pour bloquer une adresse qui aurait un comportement abusif :

```
iptables -A INPUT -s xxx.xxx.xxx.xxx -j DROP
```



Changer xxx.xxx.xxx.xxx par l'adresse IP à bloquer.

pour la débloquer :

```
iptables -D INPUT -s xxx.xxx.xxx.xxx -j DROP
```

L'option **-D** ou **--delete** supprime la règle dans la chaîne spécifiée.

### 15.2.2. Bloquer/accepter des ports spécifiques

Pour bloquer un port spécifique en sortie :

```
iptables -A OUTPUT -p tcp --dport xxx -j DROP
```

alors que pour autoriser une connexion entrante :

```
iptables -A INPUT -p tcp --dport xxx -j ACCEPT
```

Pour bloquer le protocole udp, remplacer simplement tcp par udp.

Plusieurs ports peuvent être spécifiés en même temps :

```
iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
```

### **15.2.3. Autoriser un réseau**

Un réseau complet peut être spécifié, par exemple pour autoriser un accès SSH :

```
iptables -A OUTPUT -p tcp -d 10.10.10.0/24 --dport 22 -j ACCEPT
```

### **15.2.4. Limiter le nombre de connexion d'une adresse**

Pour limiter le nombre de paquets par adresse (protection contre le flooding) :

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 50/minute --limit-burst 100 -j ACCEPT
```

Pour limiter le nombre de connexions actives (ici 3 sur le port ssh) :

```
iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT
```

### **15.2.5. Bloquer le protocole ICMP (ping)**

Bloquer le protocole ICMP peut être considéré comme une mesure de sécurité.

```
iptables -A INPUT -p icmp -i eth0 -j DROP
```

### 15.2.6. Accès à la loopback

L'accès à l'interface de loopback doit toujours être possible. Les lignes suivantes doivent impérativement être présentes :

```
iptables -A INPUT -i lo -j ACCEPT  
iptables -A OUTPUT -o lo -j ACCEPT
```

### 15.2.7. Logger les paquets refusés

Iptables peut envoyer à syslog (/var/log/messages) les paquets qu'il refuse :

```
iptables -A INPUT -i eth0 -J LOG --log-prefix "REFUS IPTABLES : "
```

Cette règle est à mettre en toute dernière, juste avant la suppression des paquets.

### 15.2.8. Gérer les connexions établies

Il est nécessaire d'autoriser les paquets provenant de connexions déjà établies ou en relation avec d'autres connexions.

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT  
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

### 15.2.9. Supprimer les paquets invalides

Certains paquets sont marqués invalides à l'arrivée (duplication, déséquancement, etc.). Il est possible de les logger pour pouvoir éventuellement mener des actions correctrices, puis de les supprimer :

```
iptables -A INPUT -m conntrack --ctstate INVALID -J LOG --log-prefix "REFUS IPTABLES : "  
iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

### 15.2.10. Bloquer le trafic SMTP sortant

Pour empêcher toute sortie de mails depuis un des ports correspondant aux ports SMTP :

```
iptables -A OUTPUT -p tcp --dports 25,465,587 -j REJECT
```

### 15.3. Conclusion

Ces exemples permettent de faire le tour des fonctionnalités offertes par le parefeu iptables. Quelques règles permettent d'offrir un niveau de sécurité plus important.

La mise en place du parefeu n'est donc pas un élément à négliger.

---

# 16

## Fail2ban

---

Fail2ban permet le blocage des adresses IP tentant une intrusion sur votre serveur. Pour cela, il s'appuie sur le parefeu Netfilter ou sur TCP Wrapper.

Il détecte les tentatives d'intrusion en parcourant les journaux du système.

### 16.1. Installation

Fail2ban est disponible dans le dépôt EPEL.

```
yum install fail2ban
```

Le fichier **/etc/fail2ban/jail.conf** est fourni par le paquet **fail2ban**.

Il ne faut le modifier directement, sous peine de voir ses changements perdus lors de la prochaine mise à jour du paquet. Au lieu de cela, il faut créer un fichier **/etc/fail2ban/jail.local**, et y placer sa configuration personnalisée.

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

### 16.2. Configuration

Options générales (section DEFAULT) :

- **bantime** : le temps en secondes durant lequel l'adresse IP ayant tentée une intrusion sera bannie ;
- **findtime** : la plage de temps en secondes utilisée pour analyser les logs. Plus cette plage est grande, plus l'analyse est longue ;

- **maxretry** : le nombre d'échecs de connexion tolérés avant le bannissement.

Il est primordiale d'ajuster au mieux ces trois valeurs. Un attaquant, avec la configuration par défaut, peut faire 5 tentatives toutes les 10 minutes, sans être banni. Cette valeur peut paraître ridiculement petite, mais elle représente par jour  $5 \times 6 \times 24 = 720$  tentatives, et 262 800 par an. Elle est encore à multiplier par le nombre de postes participant à l'attaque.



Même avec un système comme fail2ban, un poste n'est pas à l'abri des attaques. Fail2ban n'empêchera pas un attaquant de prendre possession de votre serveur, mais le retardera. Il est important de respecter les règles de bases de la sécurité informatique : changement fréquent de mot de passe, complexité, etc.

```
[root]# vim /etc/fail2ban/jail.local
[DEFAULT]
bantime = 3600
findtime = 600
maxretry = 5

[ssh-iptables]
enabled = true
filter = sshd
action = iptables[name=SSH, port=ssh, protocol=tcp] sendmail-
whois[name=SSH, dest=root, sender=fail2ban@formatux.fr]
logpath = /var/log/secure
maxretry = 5
```

- section ssh-iptables :
  - enabled : active la règle
  - filter : fichier de log à analyser. Un chemin complet ou un raccourci (comme c'est le cas ici)
  - action : que dois faire fail2ban en cas d'échec de connexion ?
    - iptables : activer une règle dans le parefeu,
    - sendmail-whois : envoyer un mail de rapport.

### 16.3. Lancement du service

- Fail2ban s'appuyant sur le firewall netfilter pour bannir les adresses IP tentant une intrusion, il faut s'assurer que celui-ci soit démarré :

```
service iptables status  
service ip6tables status
```

- Si le parefeu n'est pas actif sur le système :

```
chkconfig iptables on  
chkconfig ip6tables on  
service iptables start  
service ip6tables start
```

- Démarrer le service fail2ban :

```
chkconfig fail2ban on  
service fail2ban start
```

### 16.4. Vérification du service

La commande **fail2ban-client status** permet d'obtenir des informations sur les services surveillés ainsi que le nombre de règles iptables mises en place :

```
[root]# fail2ban-client status  
Status  
|- Number of jail: 2  
`- Jail list: mysqld-iptables, ssh-iptables
```

IPtables doit également renvoyer des informations concernant l'utilisation d'une chaîne fail2ban-SSH :

```
[root]# iptables --list  
  
Chain INPUT (policy ACCEPT)  
target     prot opt source          destination  
fail2ban-SSH  tcp  --  anywhere       anywhere        tcp  
               dpt:ssh
```

```
ACCEPT      all  --  anywhere          anywhere        state
RELATED,ESTABLISHED
...
Chain fail2ban-SSH (1 references)
target      prot opt source           destination
RETURN     all  --  anywhere          anywhere
```

## 16.5. Interface graphique

**Fail2Web** est une interface graphique web pour Fail2Ban, qui communique via **Fail2Rest** (service REST).



Attention à la configuration du serveur Fail2Rest, qui par défaut ne propose pas d'authentification, il faudra le faire via l'authentification apache.

# Sécuriser le serveur SSH

Le serveur openSSH permet l'administration d'un serveur à distance.

## 17.1. Configuration

La configuration du serveur SSH se fait dans le fichier `/etc/ssh/sshd_config`.

À chaque modification, il faut relancer le service :

```
service sshd restart
```

## 17.2. Changer le port d'écoute et la version du protocole

Il est préférable de changer le port par défaut (22) par un port connu de vous seul et de n'utiliser que la dernière version du protocole :

```
Port XXXX  
Protocol 2
```

## 17.3. Utilisation de clefs privées/publiques

Dans la mesure du possible, utilisez un couple de clef privée/publique pour l'accès au serveur, et désactivez les autres possibilités de connexion (authentification par utilisateur + mot de passe) :

```
PasswordAuthentication no  
RSAAuthentication yes  
PubkeyAuthentication yes
```

## 17.4. Limiter les accès

Il est possible de limiter les accès directement dans la configuration du service avec la directive AllowUsers :

```
AllowUsers antoine
```

Il est également possible de limiter les accès par adresse IP via TCP Wrapper. Par exemple, refusez tous les accès dans le fichier /etc/hosts.deny :

```
sshd: ALL
```

et n'acceptez dans le fichier /etc/hosts.allow que les connexions depuis des adresses IP validées :

```
sshd: 192.168.1. 221.10.140.10
```

Voir la configuration de TCP Wrapper pour plus d'informations.

## 17.5. Interdire l'accès à root !!!

La mesure essentielle à prendre est d'interdire l'accès direct à root au serveur ssh :

```
PermitRootLogin no
```

et d'utiliser les sudoers pour permettre aux utilisateurs administrateurs de lancer des commandes admins.

## 17.6. Sécurité par le parefeu

Il est également important de limiter les accès aux services grâce au parefeu et de bannir les adresses IP tentant des attaques par dictionnaire.

---

# 18

## Autorité de certification TLS avec easy-rsa

---

SSL (Secure Socket Layer) est le protocole historique développé par la société Netscape pour sécuriser les échanges entre les clients et les serveurs Web. SSL a été **standardisé par l'IETF** sous le nom de **TLS** (Transport Layer Security). TLS n'est rien d'autre que SSLv3 avec quelques corrections et améliorations.

Une autorité de certification (**CA, Certificate Authority**) agit comme une entité disposant d'une biclé (couple de clé privée/publique) de confiance représentant un "certificat racine". Ce certificat va seulement être employé pour signer d'autres certificats après avoir vérifié l'identité qui y est inscrite. Tout client voulant utiliser un service s'appuyant sur TLS devra disposer du certificat de sa CA pour valider l'authenticité du certificat TLS du serveur.

### 18.1. Installer easy-rsa :



Easy-rsa est disponible dans le dépôt EPEL.

```
[root]# yum install easy-rsa
```

### 18.2. Configuration

- Se déplacer dans le répertoire easy-rsa :

```
[root]# cd /usr/share/easy-rsa/2.0/
```

- Configurer les réponses par défaut :

```
[root]# vim vars
...
export KEY_COUNTRY="FR"
export KEY_PROVINCE="fr"
export KEY_CITY="Rennes"
export KEY_ORG="Formatux"
export KEY_EMAIL="admin@formatux.fr"
export KEY_OU="formatux.fr"

# X509 Subject Field
export KEY_NAME="Formatux"
```

### 18.3. Créer une autorité de certification

- Se déplacer dans le répertoire easy-rsa :

```
[root]# cd /usr/share/easy-rsa/2.0/
[root]# source ./vars
[root]# ./clean-all
[root]# ./build-ca
Generating a 2048 bit RSA private key
.....+++
.....
++++
writing new private key to 'ca.key'

-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [fr]:
Locality Name (eg, city) [Rennes]:
Organization Name (eg, company) [Formatux]:
```

```
Organizational Unit Name (eg, section) [formatux.fr]:  
Common Name (eg, your name or your server's hostname) [Formatux CA]:  
Name [Formatux]:  
Email Address [admin@formatux.fr]:  
  
[root]# ./build-dh
```

## 18.4. Créer une biclé serveur

```
[root]# cd /usr/share/easy-rsa/2.0/  
[root]# source ./vars  
[root]# ./build-key-server servername
```

## 18.5. Installer le certificat de l'autorité de certification

- Installer le package ca-certificates

```
[root]# yum install ca-certificates
```

- Activer la configuration dynamique de l'autorité de certification :

```
[root]# update-ca-trust enable
```

- Ajouter le certificat de l'autorité :

```
[root]# cp foo.crt /etc/pki/ca-trust/source/anchors/  
[root]# update-ca-trust extract
```



---

# Glossaire

**BASH**

Bourne Again SHELL

**BIOS**

Basic Input Output System

**CIDR**

Classless Inter-Domain Routing

**Daemon**

Disk And Execution MONitor

**DHCP**

Dynamic Host Control Protocol

**DNS**

Domain Name Service

**FQDN**

Fully Qualified Domain Name

**LAN**

Local Area Network

**NTP**

Network Time Protocol

**nsswitch**

Name Service Switch

**POSIX**

Portable Operating System Interface

**POST**

Power On Self Test

**SHELL**

En français "coquille". À traduire par "interface système".

---

**SMB**

Server Message Block

**SMTP**

Simple Mail Tranfer Protocol

**SSH**

Secure SHell

**TLS**

Transport Layer Security, un protocole de cryptographie pour sécuriser les communications IP.

**TTY**

teletypewriter, qui se traduit téléscripteur. C'est la console physique.

**UEFI**

Unified Extensible Firmware Interface

---

## **Partie III. Shell**



---

# Table des matières

19. Les scripts SHELL - Niveau 1 .....	303
19.1. Premier script .....	304
19.2. Variables .....	306
19.2.1. Supprimer et verrouiller les variables .....	308
19.2.2. Variables d'environnements .....	308
19.2.3. Exporter une variable .....	309
19.2.4. La substitution de commande .....	309
19.2.5. Améliorations du script de sauvegarde .....	310
19.3. Saisie et manipulations .....	312
19.3.1. La commande read .....	312
19.3.2. La commande cut .....	313
19.3.3. La commande tr .....	314
19.3.4. Extraire le nom et le chemin d'un fichier .....	315
19.3.5. Arguments d'un script .....	315
20. Scripts shell - Instructions de contrôle .....	321
20.1. Tests .....	321
20.1.1. Tester le type d'un fichier .....	322
20.1.2. Comparer deux fichiers .....	323
20.1.3. Tester une variable .....	323
20.1.4. Tester une chaîne de caractères .....	324
20.1.5. Comparaison de numériques entiers .....	324
20.1.6. Combinaison de tests .....	325
20.1.7. Les opérations numériques .....	326
20.1.8. La commande typeset .....	327
20.1.9. La commande let .....	327
20.2. Structures conditionnelles .....	328
20.2.1. Structure alternative conditionnelle case .....	330
20.3. Boucles .....	332
20.3.1. La structure boucle conditionnelle while .....	332
20.3.2. La commande exit .....	333
20.3.3. La commande break / continue .....	334
20.3.4. Les commandes true / false .....	334
20.3.5. La structure boucle conditionnelle until .....	335
20.3.6. La structure choix alternatif select .....	335

---

20.3.7. La structure boucle sur liste de valeurs for .....	336
<b>21. TP Scripting SHELL .....</b>	<b>339</b>
21.1. Etude du besoin .....	339
21.2. Consignes .....	340
21.3. Pistes de travail .....	340
21.4. Proposition de correction .....	341
21.4.1. Création de l'utilisateur .....	341
21.4.2. Menu .....	341
21.4.3. Quelques fonctions utilitaires .....	343
21.4.4. La gestion du réseau .....	345
21.4.5. La gestion des services .....	346
21.4.6. L'affichage des utilisateurs .....	347
<b>Index .....</b>	<b>351</b>

---

# 19

## Les scripts SHELL - Niveau 1

---

Le shell est l'interpréteur de commandes de Linux. Il ne fait pas partie du noyau, mais forme une couche supplémentaire, d'où son nom de "coquille".

Il a un rôle d'analyse des commandes saisies puis les fait exécuter par le système.

Il existe plusieurs shells, tous partageant des points communs :

- bourne again shell (bash),
- korn shell (ksh),
- c shell (csh),
- etc.

Le bash est présent par défaut sur les distributions Linux, il se caractérise par ses fonctionnalités pratiques et conviviales.

Le shell est aussi un langage de programmation basique qui grâce à quelques commandes dédiées permet :

- L'utilisation de variables,
- l'exécution conditionnelle de commandes,
- la répétition de commandes.

Les scripts en shell ont l'avantage d'être réalisables rapidement et de manière fiable, sans compilation ni installation de commandes supplémentaires.

Pour écrire un script shell, il suffit de réunir dans un même fichier toutes les commandes nécessaires, une commande par ligne.

En rendant ce fichier exécutable, le shell le lira séquentiellement et exécutera une à une les commandes le comportant.

Lorsque le shell rencontre une erreur, il affiche un message permettant d'identifier le problème mais continue l'exécution du script.

Les erreurs propres aux commandes sont également affichées à l'écran.

Qu'est ce qu'un bon script ? C'est un script :

- fiable : son fonctionnement est irréprochable même en cas de mauvaise utilisation ;
- commenté : son code est annoté pour en faciliter la relecture et les futures évolutions;
- lisible : le code est indenté à bon escient, une seule commande par ligne, les commandes sont aérées...
- portable : le code fonctionne sur tout système Linux, gestion des dépendances, gestion des droits, etc.

## 19.1. Premier script

Pour commencer l'écriture d'un script shell , il est pratique d'utiliser un éditeur de texte gérant la coloration syntaxique.

**vim** est un outil adapté a cela.

Le nom du script devra respecter quelques règles :

- pas de majuscule,
- pas de nom de commandes existantes,
- extension en .sh pour indiquer qu'il s'agit d'un script shell.

### hello-world.sh.

```
#!/bin/bash
#
# Auteur : Antoine Le Morvan
# Date : 18 septembre 2016
# Version 1.0.0 : Affiche le texte "Hello world !"
#
```

```
# Affiche un texte à l'écran :  
echo "Hello world !"
```

Pour pouvoir exécuter ce script, il est nécessaire de lui donner le droit d'exécution :

```
stagiaire $ chmod u+x ./hello-world.sh  
stagiaire $ ./hello-world.sh  
Hello world !
```

La première ligne à écrire dans tout script permet d'indiquer le shell à utiliser pour l'exécuter. Si vous désirez utiliser le shell ksh ou le langage interprété python, vous remplaceriez la ligne :

```
#!/bin/bash
```

par :

```
#!/bin/ksh
```

ou par :

```
#!/usr/bin/python
```

Tout au long de l'écriture, il faudra penser à la relecture du script en utilisant notamment des commentaires :

- un commentaire général en début pour indiquer le but du script, son auteur, sa version, etc.
- des commentaires au cours du texte pour aider à la compréhension des actions.

Les commentaires peuvent être placés sur une ligne à part ou bien à la fin d'une ligne contenant une commande.

Exemple :

```
# Ce programme affiche la date
```

```
date      # Cette ligne est la ligne qui affiche la date !
```

## 19.2. Variables

Comme dans tout langage de programmation, le script shell utilise des variables. Elles servent à stocker des informations en mémoire pour les réutiliser à volonté au cours du script.

Une variable est créée au moment où elle reçoit son contenu. Elle reste valide jusqu'à la fin de l'exécution du script. Puisque le script est exécuté séquentiellement du début à la fin, il est impossible de faire appel à une variable avant qu'elle ne soit créée.

Le contenu peut être modifié au cours du script, la variable continue d'exister. Si le contenu est supprimé, la variable reste active mais ne contient rien.



Il n'y a pas de notion de type de variable en script shell. Le contenu d'une variable est toujours un caractère ou une chaîne de caractères.

### 01-backup.sh.

```
#!/bin/bash

#
# Auteur : Antoine Le Morvan
# Date : 18 septembre 2016
# Version 1.0.0 : Sauvegarde dans /root les fichiers passwd, shadow,
# group et gshadow
#

# Variables globales
FICHIER1=/etc/passwd
FICHIER2=/etc/shadow
FICHIER3=/etc/group
FICHIER4=/etc/gshadow

# Dossier destination
DESTINATION=/root

# Nettoie l'écran :
clear
```

```
# Lancer la sauvegarde
echo "La sauvegarde de $FICHIER1, $FICHIER2, $FICHIER3, $FICHIER4 vers
$DESTINATION va commencer :"

cp $FICHIER1 $FICHIER2 $FICHIER3 $FICHIER4 $DESTINATION

echo "La sauvegarde est terminée !"
```

Ce script fait usage de variables. Le nom d'une variable doit commencer par une lettre mais peut ensuite contenir n'importe quelle suite de lettres ou de chiffres. Hormis le tiret bas `_`, les caractères spéciaux ne sont pas utilisables.

Par convention, les variables créées par un utilisateur ont un nom en minuscules. Ce nom doit être choisi avec précaution pour n'être ni trop évasif ni trop compliqué. Une variable peut toutefois être nommée avec des majuscules, comme c'est le cas ici, s'il s'agit d'une variable globale qui ne doit pas être modifiée par le programme.

Le caractère `=` affecte du contenu à une variable :

```
variable=valeur
nom_rep="/home"
```

Il n'y a pas d'espace ni avant ni après le signe `=`.

Pour afficher du texte en même temps que le contenu d'une variable, il est obligatoire d'utiliser les guillemets et non les apostrophes.



L'usage des apostrophes inhibe l'interprétation des caractères spéciaux.

```
stagiaire $ message="Bonjour"
stagiaire $ echo "Voici le contenu de la variable
message : $message"
Voici le contenu de la variable message : Bonjour
stagiaire $ echo 'Voici le contenu de la variable
message : $message'
Voici le contenu de la variable message : $message
```

Pour isoler le nom de la variable, il faut utiliser les apostrophes ou les accolades :

```
stagiaire $ touch "$fichier"1  
stagiaire $ touch ${fichier}1
```

### 19.2.1. Supprimer et verrouiller les variables

La commande **unset** permet de supprimer une variable.

Exemple :

```
stagiaire $ nom="NOM"  
stagiaire $ prenom="Prenom"  
stagiaire $ echo "$nom $prenom"  
NOM Prenom  
stagiaire $ unset prenom  
stagiaire $ echo "$nom $prenom"  
NOM
```

La commande **readonly** verrouille une variable.

Exemple :

```
stagiaire $ nom="NOM"  
stagiaire $ readonly nom  
stagiaire $ nom="AUTRE NOM"  
bash: nom: variable en lecture seule  
stagiaire $ unset nom  
bash: nom: variable en lecture seule
```

### 19.2.2. Variables d'environnements

Les variables d'environnements et les variables systèmes sont des variables utilisées par le système pour son fonctionnement. Par convention elles portent un nom en majuscules.

Elles peuvent être affichées ou modifiées dans un script comme n'importe quelle variable. Elles doivent cependant être modifiées avec précaution.

La commande **env** permet d'afficher toutes les variables d'environnement utilisées.

La commande **set** permet d'afficher toutes les variables système utilisées.

Parmi les dizaines de variables d'environnement, plusieurs ont un intérêt à être utilisées dans un script shell :

Tableau 19.1. Variables d'environnement

Variable	Observation
HOSTNAME	Nom d'hôte de la machine
USER, USERNAME et LOGNAME	Nom de l'utilisateur connecté sur la session
PATH	Chemin des commandes
PWD	Répertoire courant, mis à jour à chaque exécution de la commande cd.
HOME	Répertoire de connexion.

### 19.2.3. Exporter une variable

La commande **export** permet d'exporter une variable.

Une variable n'est valable que dans l'environnement du processus du script shell. Pour que les **processus fils** du script puissent connaître les variables et leur contenu, il faut les exporter.

La modification d'une variable exportée dans un processus fils ne peut pas remonter au processus père.



Sans option, la commande **export** affiche le nom et les valeurs des variables exportées dans l'environnement.

### 19.2.4. La substitution de commande

Il est possible de stocker le résultat d'une commande dans une variable.



Cette opération n'est valable que pour les commandes qui renvoient un message à la fin de leur exécution.

La syntaxe pour sous-exécuter une commande est la suivante :

**Syntaxes pour la substitution de commandes.**

```
variable=`commande`
variable=$(commande)
```

Exemples :

```
stagiaire $ jour=`date +%j`
stagiaire $ homedir=$(pwd)
```

**19.2.5. Améliorations du script de sauvegarde****Quelques pistes d'améliorations.**

```
#!/bin/bash

#
# Auteur : Antoine Le Morvan
# Date : 18 septembre 2016
# Version 1.0.0 : Sauvegarde dans /root les fichiers passwd, shadow,
# group et gshadow
# Version 1.0.1 : Création d'un répertoire avec le quantième du jour.
#                 Améliorations diverses

# Variables globales

## Fichiers à sauvegarder
FICHIER1=/etc/passwd
FICHIER2=/etc/shadow
FICHIER3=/etc/group
FICHIER4=/etc/gshadow

## Dossier destination
DESTINATION=/root

## Variables en readonly
readonly FICHIER1
readonly FICHIER2
readonly FICHIER3
readonly FICHIER4
readonly DESTINATION

# Un nom de dossier avec le quatrième du jour
rep="backup-$(date +%j)"
```

```

# Nettoie l'écran :
clear

# Lancer la sauvegarde
echo "*****"
echo "      Script de sauvegarde - Sauvegarde sur la machine $HOSTNAME "
echo "*****"
echo "La sauvegarde sera faite dans le dossier ${rep}."
echo "Création du répertoire..."
mkdir -p $DESTINATION/$rep
echo "                                [ OK ]"
echo "La sauvegarde de ${FICHIER1}, ${FICHIER2}, ${FICHIER3},
${FICHIER4} vers ${DESTINATION}/$rep va commencer :"

cp $FICHIER1 $FICHIER2 $FICHIER3 $FICHIER4 $DESTINATION/$rep

echo "La sauvegarde est terminée !"

# La sauvegarde est notée dans le journal d'évènements du système :
echo "La sauvegarde est renseignée dans syslog :"
logger "Sauvegarde des fichiers systèmes par ${USER} sur la machine
${HOSTNAME} dans le dossier ${DESTINATION}/$rep."
echo "                                [ OK ]"

```

## Exécution de notre script de sauvegarde.

```

root # ./02-backup-enhanced.sh
*****
      Script de sauvegarde - Sauvegarde sur la machine formateur1
*****
La sauvegarde sera faite dans le dossier backup-262.
Création du répertoire...
                                [ OK ]
La sauvegarde de /etc/passwd, /etc/shadow, /etc/group, /etc/gshadow
vers /root/backup-262 va commencer :
La sauvegarde est terminée !
La sauvegarde est renseignée dans syslog :
                                [ OK ]

```

Le lancement de la commande peut être visualisée dans le journal syslog :

## Evènement dans syslog.

```

root # tail -f /var/log/messages

```

```
sept. 18 19:35:35 formateur1 antoine[9712]: Sauvegarde des fichiers
systèmes par antoine sur la machine formateur1 dans le dossier /root/
b...
```

## 19.3. Saisie et manipulations

Selon l'objet du script, il peut être nécessaire d'envoyer des informations en cours d'exécution.

Ces informations, non connues lors de l'écriture du script, peuvent être extraites à partir de fichiers ou saisies par l'utilisateur.

Il est aussi possible d'envoyer ces informations sous forme d'arguments lors de la saisie de la commande du script. C'est le mode de fonctionnement de nombreuses commandes Linux.

### 19.3.1. La commande read

La commande **read** permet de saisir une chaîne de caractère pour la stocker dans une variable.

**Syntaxe de la commande read.**

```
read [-n X] [-p] [-s] [variable]
```

**Exemple de la commande read.**

```
stagiaire $ read nom prenom
stagiaire $ read -p "Veuillez saisir votre nom : " nom
```

Tableau 19.2. Options de la commande read

Option	Observation
-p	Affiche un message de prompt
-n	Limite le nombre de caractères à saisir
-s	Masque la saisie

Lors de l'utilisation de l'option **-n**, le shell valide automatiquement la saisie au bout du nombre de caractères précisés. L'utilisateur n'a pas à appuyer sur la touche [ENTREE].

```
stagiaire $ read -n5 nom
```

La commande **read** permet d'interrompre l'exécution du script le temps que l'utilisateur saisisse des informations. La saisie de l'utilisateur est découpée en mots affectés à une ou plusieurs variables prédefinies. Les mots sont des chaînes de caractères séparées par le séparateur de champs.

La fin de la saisie est déterminée par la frappe sur la touche **[ENTREE]** ou le caractère spécial de fin de ligne.

Une fois la saisie validée, chaque mot sera stocké dans la variable prédefinie.

Le découpage des mots est défini par le caractère séparateur de champs. Ce séparateur est stocké dans la variable système **IFS** (*Internal Field Separator*).

```
[root] # set | grep IFS  
IFS=$' \t\n'
```

Par défaut, l'IFS contient l'espace, la tabulation **\t** et le saut de ligne **\n**.

Utilisée sans préciser de variable, cette commande met simplement le script en pause. Le script continue son exécution lors de la validation de la saisie.

Cette utilisation permet de faire une pause lors du débogage d'un script ou pour inciter l'utilisateur à appuyer sur **[ENTREE]** pour continuer.

```
[root]# echo -n "Appuyer sur [ENTREE] pour continuer..."  
[root]# read
```

### 19.3.2. La commande cut

La commande **cut** permet d'isoler une colonne dans un fichier.

**Syntaxe de la commande cut.**

```
cut [-cx] [-dy] [-fz] fichier
```

**Syntaxe de la commande cut.**

```
[root]# cut -d: -f1 /etc/passwd
```

**Tableau 19.3. Options de la commande cut**

<b>Option</b>	<b>Observation</b>
-c	Spécifie les numéros d'ordre des caractères à sélectionner
-d	Spécifie le séparateur de champs
-f	Spécifie le numéro d'ordre des colonnes à sélectionner

Le principal intérêt de cette commande sera son association avec la commande grep et le pipe |.

La commande **grep** travaille verticalement (*isolation d'une ligne parmi toutes celles du fichier*).

La combinaison des deux commandes permet d'isoler un champ précis du fichier.

#### Syntaxe de la commande cut.

```
[root]# grep '^root:' /etc/passwd | cut -d: -f3
0
```



Les fichiers de configurations comportant une structure unique utilisant le même séparateur de champs sont des cibles idéales pour cette combinaison de commandes.

#### 19.3.3. La commande tr

La commande **tr** permet de convertir une chaîne de caractères

#### Syntaxe de la commande tr.

```
tr [-csd] chaîne1 chaîne2
```

**Tableau 19.4. Options de la commande cut**

<b>Option</b>	<b>Observation</b>
-c	Tous les caractères qui ne sont pas spécifiés dans la première chaîne sont convertis selon les caractères de la seconde.
-d	Efface le caractère spécifié

Option	Observation
-s	Réduire à une seule unité le caractère spécifié

```
[stagiaire]$ tr -s " " < /etc/hosts
```

### **Exercice : extraire le niveau d'exécution du fichier /etc/inittab**

```
#!/bin/bash

#
# Auteur : Antoine Le Morvan
# Date : 18 septembre 2016
# Version 1.0.0 : Extrait le niveau d'exécution du fichier /etc/inittab

# Variables globales

INITTAB=/etc/inittab

niveau=`grep "^id" $INITTAB | cut -d: -f2`

# Affichage du résultat :
echo "Le niveau d'init au démarrage est : $niveau"
```

#### **19.3.4. Extraire le nom et le chemin d'un fichier**

La commande **basename** permet d'extraire le nom du fichier à partir d'un chemin.  
La commande **dirname** permet d'extraire le chemin parent d'un fichier.

Exemple :

```
[root]# echo $FICHIER=/usr/bin/passwd
[root]# basename $FICHIER
passwd
[root]# dirname $FICHIER
/usr/bin
```

#### **19.3.5. Arguments d'un script**

La demande de saisie d'informations grâce à la commande **read** interrompt l'exécution du script tant que l'utilisateur ne fait pas de saisie.

Cette méthode, bien que très conviviale, présente des limites s'il s'agit d'un script à l'exécution programmée la nuit par exemple. Afin de palier ce problème il est possible d'injecter les informations souhaitées via des arguments.

De nombreuses commandes Linux fonctionnent sur ce principe.

Cette façon de faire à l'avantage qu'une fois le script exécuté, il n'aura plus besoin d'intervention humaine pour se terminer.

Son inconvénient majeur est qu'il faudra prévenir l'utilisateur du script de sa syntaxe pour éviter des erreurs.

Les arguments sont renseignés lors de la saisie de la commande du script. Ils sont séparés par un espace.

```
[root]# ./script argument1 argument2
```

Une fois exécuté, le script enregistre les arguments saisis dans des variables prédéfinies : les variables positionnelles.

Ces variables sont utilisables dans le script comme n'importe quelle autre variable, à l'exception faite qu'elles ne peuvent pas être affectées.

Les variables positionnelles non utilisées existent mais sont vides.

Les variables positionnelles sont toujours définies de la même façon :

Tableau 19.5. Les variables positionnelles

Variable	Observation
\$0	contient le nom du script tel qu'il a été saisi.
1 à \$9	contiennent les valeurs du 1er et du 9ème argument.
\${x}	contient la valeur de l'argument x, supérieur à 9.
\$#	contient le nombre d'arguments passés.
\$*	contient en une variable tous les arguments passés

Exemples :

```
[root]# ./script.sh un deux trois
```

```
[root]# echo $3 $2 $1
trois deux un
[root]# echo $0 $# $*
./script.sh 3 un deux trois
```



Attention : il existe une autre variable positionnelle, `$@`, qui contient tous les arguments passés. La confusion avec `$*` est aisée.

La différence se fait au niveau du format de stockage des arguments : `$*` : Contient les arguments au format "`$1 $2 $3 ...`" `$@` : Contient les arguments au format "`$1" "$2" "$3" ...`

## ***La commande shift***

La commande **shift** permet de décaler les variables positionnelles.

Exemples :

```
[root]# ./script.sh un deux trois
[root]# echo $1
un
[root]# shift 2
[root]# echo $1
trois
```



Attention : Lors de l'utilisation de la commande **shift**, les variables `$#` et `$*` sont modifiées en conséquence.

## ***La commande set***

La commande **set** découpe une chaîne en variables positionnelles.

**Syntaxe de la commande set.**

```
set [valeur] [$variable]
```

Exemple :

```
[root]# set un deux trois
```

```
[root]# echo $1 $2 $3 $#
un deux trois 3
[root]# variable="un deux trois"
[root]# set $variable
[root]# echo $1 $2 $3 $#
un deux trois 3
```

Ci-dessous, la version de notre script de sauvegarde mettant en oeuvre les variables positionnelles :

```
#!/bin/bash

#
# Auteur : Antoine Le Morvan
# Date : 18 septembre 2016
# Version 1.0.0 : Sauvegarde dans /root les fichiers passwd, shadow,
# group et gshadow
# Version 1.0.1 : Création d'un répertoire avec le quantième du jour.
# Améliorations diverses
# Version 1.0.2 : Modification pour utiliser les variables
# positionnelles
# Limitation à 5 fichiers

# Variables globales

## Dossier destination
DESTINATION=/root

# Un nom de dossier avec le quantième du jour
rep="backup-$(date +%j)"

# Nettoie l'écran :
clear

# Lancer la sauvegarde
echo ****
echo "      Script de sauvegarde - Sauvegarde sur la machine $HOSTNAME "
echo ****
echo "La sauvegarde sera faite dans le dossier ${rep}."
echo "Création du répertoire..."
mkdir -p $DESTINATION/$rep
echo "                                [ OK ]"
echo "La sauvegarde de ${1} ${2} ${3} ${4} ${5} vers ${DESTINATION}/${rep}
va commencer :"
```

```
cp $1 $2 $3 $4 $5 $DESTINATION/$rep
```

```
echo "La sauvegarde est terminée !"
```



---

# 20

## Scripts shell - Instructions de contrôle

---

### 20.1. Tests

Lorsqu'elles se terminent, toutes les commandes exécutées par le shell renvoient un code de retour (également appelé code de statut ou de sortie).

La convention établie veut que si la commande s'est bien exécutée, le code de statut ait pour valeur zéro.

Si la commande a eu un problème lors de son exécution (les raisons peuvent être nombreuses : manque de droits d'accès, absence de fichier, saisie incorrecte, ...), son code de status aura une valeur différente de zéro.

Il faut se référer au manuel de la commande pour connaître les différentes valeurs du code de retour prévues par les développeurs.

Le code de retour n'est pas visible directement, mais est enregistré dans une variable spéciale : `$?`.

```
[stagiaire] $ mkdir repertoire
[stagiaire] $ echo $?
0
[stagiaire] $ mkdir /repertoire
mkdir: impossible de créer le répertoire
[stagiaire] $ echo $?
1
```

L'affichage du contenu de la variable `$?` avec la commande `echo` se fait immédiatement après la commande que l'on souhaite évaluer. Cette variable étant mise à jour après chaque commande.

Il est également possible de créer des codes de retour dans un script. Il suffit pour cela d'ajouter un argument numérique à la commande `exit`.

```
[stagiaire] $ exit 2
[stagiaire] $ echo $?
2
```

Outre la bonne exécution d'une commande, le shell offre la possibilité d'exécuter des tests sur de nombreux motifs :

- Fichiers (existance, type, droits, comparaison) ;
- Chaînes de caractères (longueur, comparaison) ;
- Numériques entiers (valeur, comparaison).

Le résultat du test :

- `$?=0` : le test s'est correctement exécuté et est vrai ;
- `$?=1` : le test s'est correctement exécuté et est faux ;
- `$?=2` : le test ne s'est pas correctement exécuté.

### **20.1.1. Tester le type d'un fichier**

**Syntaxe de la commande `test` pour un fichier.**

```
test [-d|-e|-f|-L] fichier
```

Tableau 20.1. Options de la commande `test` sur les fichiers

Option	Observation
<code>-e</code>	Teste si le fichier existe
<code>-f</code>	Teste si le fichier existe et est de type normal
<code>-d</code>	Teste si le fichier existe et est de type répertoire
<code>-L</code>	Teste si le fichier existe et est de type lien symbolique
<code>-b</code>	Teste si le fichier existe et est de type spécial mode bloc

Option	Observation
-c	Teste si le fichier existe et est de type spécial mode caractère
-p	Teste si le fichier existe et est de type tube
-S	Teste si le fichier existe et est de type socket
-t	Teste si le fichier existe et est de type terminal
-r	Teste si le fichier existe et est accessible en lecture
-w	Teste si le fichier existe et est accessible en écriture
-x	Teste si le fichier existe et est exécutable
-g	Teste si le fichier existe et est a un SGID positionné
-u	Teste si le fichier existe et est a un SUID positionné
-s	Teste si le fichier existe et est non vide (taille > 0 octets)

### 20.1.2. Comparer deux fichiers

La commande **test** peut également comparer des fichiers :

#### Syntaxe de la commande test pour la comparaison de fichiers.

```
test fichier1 [-nt|-ot|-ef] fichier2
```

Tableau 20.2. Options de la commande test pour la comparaison de fichiers

Option	Observation
-nt	Teste si le premier fichier est plus récent qu le second
-ot	Teste si le premier fichier est plus ancien que le second
-ef	Teste si le premier fichier est un lien physique du second

### 20.1.3. Tester une variable

#### Syntaxe de la commande test pour les variables.

```
test [-z|-n] $variable
```

Tableau 20.3. Options de la commande test pour les variables

Option	Observation
-z	Teste si la variable est vide

Option	Observation
-n	Teste si la variable n'est pas vide

### 20.1.4. Tester une chaîne de caractères

Syntaxe de la commande test pour les chaînes de caractères.

```
test chaîne1 [=| !=] chaîne2
```

Exemple :

```
[stagiaire] $ test "$var" = "Hello world !"
[stagiaire] $ echo $?
0
```

Tableau 20.4. Options de la commande test pour les variables

Option	Observation
=	Teste si la première chaîne de caractères est égale à la seconde
!=	Teste si la première chaîne de caractères est différente de la seconde
<	Teste si la première chaîne de caractères est avant la seconde dans l'ordre ASCII
>	Teste si la première chaîne de caractères est après la seconde dans l'ordre ASCII

### 20.1.5. Comparaison de numériques entiers

Syntaxe de la commande test pour les entiers.

```
test "num1" [-eq|-ne|-gt|-lt] "num2"
```

Exemple :

```
[stagiaire] $ test "$var" -eq "1"
[stagiaire] $ echo $?
0
```

**Tableau 20.5. Options de la commande test pour les entiers**

Option	Observation
-eq	Teste si le premier nombre est égal au second
-ne	Teste si le premier nombre est différent au second
-gt	Teste si le premier nombre est supérieur au second
-lt	Teste si le premier nombre est inférieur au second



Les numériques étant traités par le shell comme des caractères (ou chaînes de caractères) classiques, un test sur un caractère peut renvoyer le même résultat qu'il soit traité en tant que numérique ou non.

```
[stagiaire] $ test "1" = "1"
[stagiaire] $ echo $?
0
[stagiaire] $ test "1" -eq "1"
[stagiaire] $ echo $?
0
```

Mais le résultat du test n'aura pas la même signification :

- Dans le premier cas, il signifiera que les deux caractères ont la même valeur dans la table ASCII.
- Dans le second cas, il signifiera que les deux nombres sont égaux.

### 20.1.6. Combinaison de tests

La combinaison de test permet d'effectuer plusieurs tests en une seule commande. Il est possible de tester plusieurs fois le même argument (fichier, chaîne ou numérique) ou des arguments différents.

```
test option1 argument1 [-a|-o] option2 argument 2
```

```
[stagiaire] $ test -d /etc -a -x /etc
[stagiaire] $ echo $?
```

0

**Tableau 20.6. Options de combinaison de tests**

Option	Observation
-a	ET : Le test sera vrai si tous les motifs le sont.
-o	OU : Le test sera vrai si au moins un motif l'est.

Les tests peuvent ainsi être groupé avec des parenthèses ( ) pour leur donner une priorité.

```
(TEST1 -a TEST2) -a TEST3
```

Le caractère ! permet d'effectuer le test inverse de celui demandé par l'option :

```
[stagiaire] $ test -e /fichier # test si fichier existe
[stagiaire] $ ! test -e /fichier # test si fichier n'existe pas
```

### 20.1.7. Les opérations numériques

La commande **expr** effectue une opération avec des entiers numériques.

```
expr num1 [+] [-] [\*] [/] [%] num2
```

Exemple :

```
[stagiaire] $ expr 2 +2
4
```



Dans le cas d'une multiplication, le caractère joker \* est précédé par \ pour éviter une mauvaise interprétation.

Opérateur	Observation
+	Addition
-	Soustraction
\*	Multiplication

Opérateur	Observation
/	Quotient de la division
%	Modulo de la division

### 20.1.8. La commande typeset

La commande **typeset -i** déclare une variable comme un entier.

Exemple :

```
[stagiaire] $ typeset -i var1
[stagiaire] $ var1=1+1
[stagiaire] $ var2=1+1
[stagiaire] $ echo $var1
2
[stagiaire] $ echo $var2
1+1
```

### 20.1.9. La commande let

La commande **let** teste si un caractère est numérique.

Exemple :

```
[stagiaire] $ var1="10"
[stagiaire] $ var2="AA"
[stagiaire] $ let $var1
[stagiaire] $ echo $?
0
[stagiaire] $ let $var2
[stagiaire] $ echo $?
1
```



La commande **let** ne retourne pas un code retour cohérent lorsqu'elle évalue le numérique 0

```
[stagiaire] $ let 0
[stagiaire] $ echo $?
1
```

La commande **let** permet également d'effectuer des opérations mathématiques :

```
[stagiaire] $ let var=5+5
[stagiaire] $ echo $var
10
```

## 20.2. Structures conditionnelles

Si la variable \$? permet de connaître le résultat d'un test ou de l'exécution d'une commande elle ne peut qu'être affichée et n'a aucune incidence sur le déroulement d'un script.

Mais nous pouvons nous en servir dans une condition. **Si** le test est bon **alors** je fais cette action **sinon** je fais telle autre action.

### Syntaxe de l'alternative conditionnelle if.

```
if commande
then
    commande si $?=0
else
    commande si $?!=0
fi
```

La commande placée après le mot **if** peut être n'importe quelle commande puisque c'est son code de retour, **\$?**, qui sera évalué. Il est souvent pratique d'utiliser la commande **test** pour définir plusieurs actions en fonction du résultat de ce test (fichier existe, variable non vide, droits en écriture positionnés.). Utiliser une commande classique (**mkdir**, **tar**, ...) permet de définir les actions à effectuer en cas de succès ou les messages d'erreur à afficher en cas d'échec.

```
if test -e /etc/passwd
then
    echo "Le fichier existe"
else
    echo "Le fichier n'existe pas"
fi

if mkdir rep
then
```

```
cd rep
fi
```

Si le bloc **else** commence par une nouvelle structure **if**, il est possible de fusionner **else** et **if** :

```
[...]
else
  if test -e /etc/
[...]

[...]
# est équivalent à
elif test -e /etc
[...]
```

La structure **if / then / else / fi** évalue la commande placée après **if** :

- Si le code retour de cette commande est 0 (vrai) le shell exécutera les commandes placées après **then** ;
- Si le code retour est différent de 0 (faux) le shell exécutera les commandes placées après **else**.

Le bloc **else** est facultatif.

Il existe un besoin d'effectuer certaines actions uniquement si l'évaluation de la commande est vraie, et n'avoir rien à faire si elle est fausse.

Le mot **fi** ferme la structure.

Lorsqu'il n'y a qu'une seule commande à exécuter dans le bloc **then**, il est possible d'utiliser une syntaxe plus simple.

La commande à exécuter si **\$?** est vrai est placée après **&&** tandis que la commande à exécuter si **\$?** est faux est placée après **||** (*facultatif*).

Par exemple :

```
[stagiaire]$ test -e /etc/passwd && echo "Le fichier existe" || echo "Le
fichier n'existe pas"
```

```
[stagiaire]$ mkdir repert && echo "Le répertoire est créé"
```

Il est possible d'évaluer et de remplacer une variable avec une structure plus légère que **if**.

Cette syntaxe met en oeuvre les accolades :

- Affiche une valeur de remplacement si la variable est vide :

```
 ${variable:-valeur}
```

- Affiche une valeur de remplacement si la variable n'est pas vide :

```
 ${variable:+valeur}
```

- Affecte une nouvelle valeur à la variable si elle est vide :

```
 ${variable:=valeur}
```

Exemples :

```
[stagiaire]$ nom=""
[stagiaire]$ echo ${nom:-linux}
linux
[stagiaire]$ echo $nom

[stagiaire]$ echo ${nom:=linux}
linux
[stagiaire]$ echo $nom
linux
[stagiaire]$ echo ${nom:+tux}
tux
[stagiaire]$ echo $nom
linux
```

### **20.2.1. Structure alternative conditionnelle case**

Une succession de structures **if** peut vite devenir lourde et complexe. Lorsquelle concerne l'évaluation d'une même variable, il est possible d'utiliser une structure

conditionnelle à plusieurs branches. Les valeurs de la variable peuvent être précisées ou appartenir à une liste de possibilités.

Les caractères jokers sont utilisables.

La structure **case / esac** évalue la variable placée après **case** et la compare aux valeurs définies. À la première égalité trouvée, les commandes placées entre ) et ;; sont exécutées.

La variable évaluée et les valeurs proposées peuvent être des chaînes de caractères ou des résultats de sous-exécutions de commandes.

Placé en fin de structure, le choix \* indique les actions à exécuter pour toutes les valeurs qui n'ont pas été précédemment testées.

### Syntaxe de l'alternative conditionnelle case.

```
case $ in
    valeur1)
        commandes si variable = valeur1
    ;;
    valeur2)
        commandes si variable = valeur2
    ;;
    [..]
    *)
        commandes pour toutes les valeurs de variable != de valeur1 et
        valeur2
    ;;
esac
else
    commande si $?!=0
fi
```

Lorsque la valeur est sujette à variation, il est conseillé d'utiliser les caractères jokers [] pour spécifier les possibilités :

```
[0o][Uu][Ii])
echo "oui"
;;
```

Le caractère | permet aussi de spécifier une valeur ou une autre :

```
"oui" | "OUI")
echo "oui"
;;
```

## 20.3. Boucles

Le shell bash permet l'utilisation de boucles. Ces structures permettent l'exécution d'un bloc de commandes plusieurs fois (de 0 à l'infini) selon une valeur définie statiquement, dynamiquement ou sur condition :

- **while**
- **until**
- **for**
- **select**

Quelle que soit la boucle utilisée, les commandes à repéter se placent entre les mots **do** et **done**.

### 20.3.1. La structure boucle conditionnelle while

La structure **while / do / done** évalue la commande placée après **while**.

Si cette commande est vrai (`$? = 0`), les commandes placées entre **do** et **done** sont exécutées. Le script retourne ensuite au début évaluer de nouveau la commande.

Lorsque la commande évaluée est fausse (`$? != 0`), le shell reprend l'exécution du script à la première commande après **done**.

#### Syntaxe de la structure boucle conditionnelle while.

```
while commande
do
    commande si $? = 0
done
```

Exemple :

```
while test -e /etc/passwd
```

```
do  
    echo "Le fichier existe"  
done
```



Si la commande évaluée ne varie pas, la boucle sera infinie et le shell n'exécutera jamais les commandes placées à la suite dans le script. Cela peut être volontaire, mais aussi être une erreur. Il faut donc faire très attention à la commande qui régit la boucle et trouver un moyen d'en sortir.

Pour sortir d'une boucle while, il faut faire en sorte que la commande évaluée ne soit plus vraie, ce qui n'est pas toujours possible.

Il existe des commandes qui permettent de modifier le comportement d'une boucle :

- **exit**
- **break**
- **continue**

### **20.3.2. La commande exit**

La commande **exit** termine l'exécution du script.

**Syntaxe de la commande exit.**

```
exit [n]
```

Exemple :

```
[stagiaire]$ exit 1  
[stagiaire]$ echo $?  
1
```

La commande exit met fin au script immédiatement. Il est possible de préciser le code de retour du script en le précisant en argument (*de 0 à 255*). Sans argument précisé, c'est le code de retour de la dernière commande du script qui sera transmise à la variable \$?.

Cette commande est utile dans le cas d'un menu proposant la sortie du script dans les choix possibles.

### 20.3.3. La commande break / continue

La commande **break** permet d'interrompre la boucle en allant à la première commande après **done**.

La commande **continue** permet de relancer la boucle en revenant à la première commande après **do**.

```
while test -d /
do
    echo "Voulez-vous continuer ? (oui/non)"
    read rep
    test $rep = "oui" && continue
    test $rep = "non" && break
done
```

### 20.3.4. Les commandes true / false

La commande **true** renvoie toujours vrai tandis que la commande **false** renvoie toujours faux.

```
[stagiaire]$ true
[stagiaire]$ echo $?
0
[stagiaire]$ false
[stagiaire]$ echo $?
1
```

Utilisées comme condition d'une boucle, elles permettent soit d'exécuter une boucle infinie soit de désactiver cette boucle.

Exemple :

```
while true
do
    echo "Voulez-vous continuer ? (oui/non)"
    read rep
```

```
test $rep = "oui" && continue
test $rep = "non" && break
done
```

### 20.3.5. La structure boucle conditionnelle until

La structure **until** / **do** / **done** évalue la commande placée après **until**.

Si cette commande est fausse (`$? != 0`), les commandes placées entre **do** et **done** sont exécutées. Le script retourne ensuite au début évaluer de nouveau la commande.

Lorsque la commande évaluée est vraie (`$? = 0`), le shell reprend l'exécution du script à la première commande après **done**.

**Syntaxe de la structure boucle conditionnelle while.**

```
until commande
do
    commande si $? != 0
done
```

Exemple :

```
until test -e /etc/passwd
do
    echo "Le fichier n'existe pas"
done
```

### 20.3.6. La structure choix alternatif select

La structure **select** / **do** / **done** permet d'afficher rapidement un menu avec plusieurs choix et une demande de saisie.

À chaque élément de la liste correspond un choix numéroté. À la saisie, la valeur choisie est affectée à la variable placée après **select** (*créée à cette occasion*).

Elle exécute ensuite les commandes placées entre **do** et **done** avec cette valeur.

- La variable "PS3" va permettre de demander à l'utilisateur de faire un choix;
- La variable "REPLY" va permettre de récupérer le numéro du choix.

Il faut une commande **break** pour sortir de la boucle.



La structure **select** est très utile pour de petits menus simples et rapides. Pour personnaliser un affichage plus complet, il faudra utiliser les commandes **echo** et **read** dans une boucle **while**.

### Syntaxe de la structure boucle conditionnelle select.

```
PS3="Votre choix :"
select variable in var1 var2 var3
do
    commandes
done
```

Exemple :

```
PS3="Votre choix : "
select choix in café thé chocolat
do
    echo "Vous avez choisi le $REPLY : $choix"
done
```

ce qui donne à l'exécution :

```
1) Café
2) Thé
3) Chocolat
Votre choix : 2
Vous avez choisi le choix 2 : thé
Votre choix :
```

### 20.3.7. La structure boucle sur liste de valeurs for

La structure **for / do / done** affecte le premier élément de la liste à la variable placée après **for** (*créée à cette occasion*).

Elle exécute ensuite les commandes placées entre **do** et **done** avec cette valeur. Le script retourne ensuite au début affecter l'élément suivant de la liste à la variable de travail.

Lorsque le dernier élément a été utilisé, le shell reprend l'exécution à la première commande après **done**.

### Syntaxe de la structure boucle sur liste de valeurs for.

```
for variable in liste
do
    commandes
done
```

Exemple :

```
for fichier in /home /etc/passwd /root/fic.txt
do
    file $fichier
done
```

Toute commande produisant une liste de valeurs peut être placée à la suite du **in** à l'aide d'une sous-exécution. La boucle **for** prendra le résultat de cette commande comme liste d'éléments sur laquelle boucler.

Cela peut être les fichiers d'un répertoire. Dans ce cas, la variable prendra comme valeur chacun des noms des fichiers présents :

```
for fichier in `ls /root`
do
    echo $fichier
done
```

Cela peut être les lignes d'un fichier. Dans ce cas, la variable prendra comme valeur chacune des lignes du fichier parcouru, du début à la fin :

```
for ligne in `more /etc/hosts`
do
    echo $ligne
done
```



---

# 21

---

## TP Scripting SHELL

---

Votre entreprise a besoin d'une solution sécurisée permettant aux personnels de la supervision d'intervenir dans un cadre maîtrisé sur les serveurs.

### 21.1. Etude du besoin

Votre responsable vous demande de développer un outil destiné aux superviseurs. Ils pourront effectuer quelques actions d'administration ainsi que les premiers diagnostics avant de faire intervenir le personnel d'astreinte.

Le personnel doit pouvoir se connecter aux serveurs via un compte générique : **supervision**.

Lorsque l'utilisateur se connecte, un menu est proposé, lui permettant :

- De gérer les utilisateurs :
  - afficher le nombre d'utilisateurs du serveur et les afficher sous forme de 2 listes :
    - les utilisateurs systèmes,
    - les utilisateurs standards ;
  - afficher les groupes du serveur et les afficher sous forme de 2 listes :
    - les groupes systèmes,
    - les groupes standards ;
  - créer un groupe : le superviseur devra fournir le GID ;
  - créer un utilisateur : le superviseur devra fournir l'UID, le GID, etc. ;

- changer le mot de passe d'un utilisateur ; l'utilisateur sera forcé de changer son mot de passe lors de sa prochaine connexion.
- De gérer les services :
  - relancer le serveur apache ;
  - relancer le serveur postfix.
- De tester le réseau :
  - Afficher les informations du réseau (Adresse IP, masque, passerelle, serveurs DNS) ;
  - Tester le réseau (localhost, ip, passerelle, serveur distant, résolution DNS).
- Actions diverses :
  - redémarrer le serveur ;
  - quitter le script (l'utilisateur est déconnecté).

Les actions du superviseur devront être renseignées dans les journaux systèmes.

## 21.2. Consignes

- Les scripts sont stockés dans /opt/supervision/scripts/ ;
- Effectuer tous les tests que vous jugerez nécessaires ;
- Découper le code en plusieurs scripts ;
- Utiliser des fonctions pour organiser le code ;
- Commenter le code.

## 21.3. Pistes de travail

- L'utilisateur supervision aura besoin des droits sudo pour les commandes réservées à root.
- Le système attribue le shell /bin/bash à un utilisateur standard, tentez d'attribuer votre script à la place !
- Utilisez la commande logger pour suivre les actions des superviseurs.
- Visitez le site : <https://www.shellcheck.net/>

## 21.4. Proposition de correction



Le code présenté ci-dessous n'est qu'une ébauche effectuée en TP par des stagiaires après 12 heures de cours de script. Il n'est pas parfait mais peut servir de base de correction ou de départ pour l'élaboration d'un travail plus complet.

### 21.4.1. Crédation de l'utilisateur

L'utilisateur doit être créé en remplaçant son shell (option -s) par le script que nous allons créer :

```
useradd -s /opt/supervision/scripts/supervision.sh -g users supervision
```

Il faut autoriser l'utilisateur supervision à utiliser sudo mais seulement pour les commandes autorisées. Pour cela, nous allons créer un fichier /etc/sudoers.d/supervision contenant les directives nécessaires :

```
# Liste les commandes autorisees aux superviseurs
Cmnd_Alias SUPERVISION = /sbin/reboot, /sbin/ip

# Autorise le superviseur a lancer les commandes precedentes sans saisir
# de mot de passe
supervision    ALL=NOPASSWD:SUPERVISION
```

### 21.4.2. Menu

Créer le fichier /opt/supervision/scripts/supervision.sh et lui donner les droits en exécution :

```
mkdir -p /opt/supervision/scripts
touch /opt/supervision/scripts/supervision.sh
chown supervision /opt/supervision/scripts/*
chmod u+x /opt/supervision/scripts/*
```

La même opération sera effectuée pour chaque script créé.

```
#!/bin/bash
```

```
# Base des scripts

BASE=$(dirname "$0")
readonly BASE

. $BASE/utils.sh

function print_menu {
    while (true)
    do
        clear
        banner
        warning
        echo "Vous pouvez : "
        echo ""
        echo " => 1) Relancer le serveur"
        echo " => 2) Afficher la conf IP"
        echo " => 3) Tester le reseau   "
        echo " => 4) Afficher les utilisateurs"
        echo " => 5) Relancer le service apache"
        echo " => 6) Relancer le service postfix"
        echo " => Q) Quitter ce super programme "
        echo ""
        read -p "Que voulez vous faire : " choix
        echo ""
        case $choix in
            "1")
                sudo reboot
                ;;
            "2")
                $BASE/print-net.sh
                ;;
            "3")
                $BASE/check-net.sh
                ;;
            "4")
                $BASE/affiche-utilisateurs.sh
                ;;
            "5")
                $BASE/gestion-services.sh "httpd"
                ;;
            "6")
                $BASE/gestion-services.sh "postfix"
                ;;
            "q" | "Q" | "quitter" | "quit")
                exit 0
            ;;
        esac
    done
}
```

```
        exit 0
    ;;
*)
    echo "Cette fonction n'est pas encore developpee"

esac
pause
done
}
banner

echo "Bienvenue sur votre console d'administration"
echo ""
echo "Vous pouvez effectuer quelques diagnostics avantd'appeler le
personnel d'astreinte"
echo ""
warning

pause

print_menu

exit 0
```

### 21.4.3. Quelques fonctions utilitaires

Le fichier utils.sh contient des fonctions que nous utiliserons dans chaque script :

```
#!/bin/bash
#
# Fonctions utilitaires et variables globales
# Version 1
#
ok="          [OK]"
nok="         [NOK]"

# Affiche ok ou nok
# Arguments :
# $1 = 0 ou 1
# $2 = message a imprimer
function printOK {
    echo "$1"
    if test "$2" = "0"
    then
```

```

    echo "$ok"
else
    echo "$nok"
fi
}

function banner {
echo "*           Bienvenue dans l'outil de la      *"
echo "           S U P E R V I S I O N      "
echo "
echo " _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ "
echo "/ \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ "
echo "\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ "
echo "| | | | | | | | | | | | | | | | | | | | | | | | | | "
echo "| | | | | | | | | | | | | | | | | | | | | | | | | | "
echo "| | | | | | | | | | | | | | | | | | | | | | | | | | "
echo " | | | | | | | | | | | | | | | | | | | | | | | | | | "
}

function pause {
    echo "Appuyer sur Entrée pour continuer..."
    read
}

function warning {
    echo "ATTENTION !!!"
    echo "Toutes les actions entreprises sont renseignées dans le journal"
    echo "d'évenement !"
    echo ""
}

```

Le fichier net-utils.sh contient les fonctions liées au réseau :

```

#!/bin/bash

#
# Fonction utilitaires du reseau
# Version 1
# Depends de utils.sh

function getGateway {
    gateway=$(sudo ip route | grep default | cut -d" " -f3)
    echo $gateway
}

```

```

function getDNS {
    DNS=$(grep "nameserver" /etc/resolv.conf | tail -1 | cut -d" " -f2)
    echo $DNS
}

# Test une adresse IP
function checkIP {
    ip=$1
    msg="Test de l'adresse ip : $ip"
    ping -c 1 $ip 1>/dev/null 2>&1
    printOK "$msg" "$?"
}

# test une resolution DNS
function checkDNS {
    res=$(dig +short www.free.fr | wc -l)
    if test "$res" -gt "0"
    then
        printOK "La resolution DNS fonctionne" "0"
    else
        printOK "La resolution DNS ne fonctionne pas" "1"
    fi
}

function getPrefix {
    sudo ip add sh | grep " inet " | grep -v "127.0.0.1" | tr -s ' ' | cut
    -d" " -f 3 | cut -d "/" -f2
}

```

#### 21.4.4. La gestion du réseau

Le fichier print-net.sh :

```

#!/bin/bash

#
# Test du reseau
# Version 1
#
# Arguments :
#
ici=$(dirname "$0")
. $ici/utils.sh

```

```
. $ici/net-utils.sh

echo "L'adresse IP de votre serveur est      : $(hostname -i)"
echo "L'adresse IP de votre gateway est       : $(getGateway)"
echo "L'adresse IP de votre serveur DNS est : $(getDNS)"
echo -n "Votre prefix est : "
getPrefix

echo ""
```

Le fichier check-net.sh :

```
#!/bin/bash

#
# Test du reseau
# Version 1
#
# Arguments :
#
ici=$(dirname "$0")
. $ici/utils.sh
. $ici/net-utils.sh

# Gestion du service fourni en argument
checkIP 127.0.0.1
checkIP $(hostname -i)
checkIP $(getGateway)
checkIP $(getDNS)
checkDNS
```

#### **21.4.5. La gestion des services**

```
#!/bin/bash

#
# Gestion des services
# Version 1
#
# Arguments :
# $1 : le nom du service a relancer
#
. ./utils.sh
```

```
# Test l'état du service
# Si le service est démarré, il propose de le relancer
# Sinon le service est démarré
function startService {
    service=$1
    service $service status 1> /dev/null 2>&1
    status=$?
    if test "$status" = "0"
    then
        # Le service fonctionne déjà
        # Faut-il le relancer ?
        echo "Le service $service fonctionne..."
        read -p "Voulez-vous le relancer ? O/N" rep
        if test "$rep" = "0" -o "$rep" = "o"
        then
            # L'utilisateur a demandé à le relancer
            logger "SUPP -> Relance d'apache"
            msg="Relance du serveur $service"
            service $service restart 1> /dev/null 2>&1
            printOK "$msg" "$?"
        else
            # L'utilisateur ne veut pas le relancer
            msg="Le service ne sera pas relancé"
            printOK "$msg" "0"
        fi
    else
        # Le service ne fonctionne pas
        # Demarrage
        logger "SUPP -> Demarrage d'apache"
        msg="Lancement du serveur $service"
        service $service start 1> /dev/null 2>&1
        printOK "$msg" "$?"
    fi
}

# Gestion du service fourni en argument
service=$1
startService $service
```

#### 21.4.6. L'affichage des utilisateurs

Le fichier affiche-utilisateur.sh :

```
#!/bin/bash

# Extrait du fichier /etc/passwd la liste :
# - des utilisateurs du systeme
# - des utilisateurs standards
# Chaque liste est affiche sur une ligne
#
# Version 1.0
# Date : 24/11/2016

# usersys : la liste des utilisateurs systemes
usersys="Voici la liste des utilisateurs systemes :\n"
# userstd : la liste des utilisateurs standards
userstd="Voici la liste des utilisateurs standard :\n"

# Stocker l'IFS dans une variable
OLDIFS='$IFS'
# Pour que la commande for fonctionne, il faut supprimer l'espace comme
# caractere de separation
IFS=$'\n'
# On boucle sur chaque ligne du fichier /etc/passwd
while read -r ligne
do
    # Isoler l'UID
    uid=$(echo $ligne | cut -d: -f3)
    # Isoler le Nom
    nom=$(echo $ligne | cut -d: -f1)
    # Si uid < 500 => Utilisateur systeme
    if test "$uid" -lt "500"
    then
        # Ajouter le nom a la liste
        usersys="${usersys}${nom}, "
    else
        # Ajouter le nom a la liste
        userstd="${userstd}${nom}, "
    fi
done < /etc/passwd

# Affichage de la liste
echo -e "$usersys"
echo ""
echo -e "$userstd"
```

```
IFS=$OLDIFS
```



---

# Index

## Symboles

&, 143

## A

alias, 36, 59

arp-scan, 231

asynchrone, 142

Attributs des fichiers, 126

audit2allow, 279

audit2why, 279

## B

bg, 144

BIOS, 169, 191

Bloc de boot, 112

bunzip2, 166, 167

bzip, 154

bzip2, 163

## C

CA, 293

cat, 45

cd, 34

cfdisk, 103, 104

chage, 95

chcon, 278

chemin absolu, 33

chemin relatif, 33

chgrp, 90

chkconfig, 184

chmod, 128

chown, 89

clear, 29

cp, 42

cpio, 149, 160

createrepo, 240

cron, 205

crond, 205

crontab, 205, 208

## D

DAC, 269

daemon, 140

date, 30

DHCP, 222

dig, 227

distribution, 17

DNS, 216, 219

## E

echo, 29

Elilo, 170

Enforcing, 275

epel, 240

Étendue, 101

## F

Fail2ban,

faillog, 264

fdisk, 103

fg, 144

FHS, 116

file, 43

find, 50

FQDN, 216

fsck, 115

---

## G

getenforce, 275  
getent, 228  
GID, 24, 79, 81  
gpasswd, 91  
grep, 52  
groupadd, 80  
groupdel, 81  
groupmod, 80  
Grub, 170  
GRUB2, 191  
grub2-mkconfig, 192  
grub2-setpassword, 194  
grub-crypt, 175  
gunzip, 167  
gzip, 153, 163, 166

## H

halt, 189  
head, 46  
history, 28  
home directory, 24  
hostname, 218  
hosts, 219

## I

id, 32, 92  
init, 137, 178, 181  
ip, 217  
IP, 221  
ipcalc, 229  
ip route, 225

## J

job, 144  
jobs, 144

## K

Kerberos, 258  
kernel, 15  
kill, 142

## L

less, 44  
Licence GPL, 19  
Lilo, 170  
Linus Torvalds, 13  
ln, 124  
ls, 35  
lvcreate, 109  
lvdisplay, 110  
LVM, 105

## M

MAC, 269  
man, 26  
matchpathcon, 278  
MBR, 191  
mkdir, 38  
mkfs, 111  
modèle OSI, 216  
more, 44  
mount, 119  
mouvement du Libre, 19  
mv, 41

## N

netstat, 230  
NetworkManager, 222  
newgrp, 92  
nice, 145  
nohup, 143  
nsswitch, 220

---

## O

OTP, 258

## P

PAM, 257  
pam\_cracklib, 263  
pam\_mount, 267  
pam\_nologin, 266  
pam\_tally, 264  
pam\_time, 265  
pam\_unix, 262  
pam\_wheel, 266  
passwd, 93  
Permissive, 275  
PID, 137  
ping, 226  
pipe, 57  
planification, 207  
POST, 169, 191  
PPID, 137  
Primaire, 101  
prompt, 24  
ps, 138  
PTS, 24  
pvcreate, 108  
pvdisplay, 110  
pwd, 34

## R

reboot, 189  
renice, 145  
restorecon, 278  
rm, 40  
rmdir, 39  
root, 83  
RPM, 235

runlevel, 178

## S

SELinux, 269  
semanage, 273  
service, 187  
sestatus, 276  
setenforce, 276  
setsebool, 274  
SGID, 133, 140  
shell, 15, 20  
shutdown, 27, 188  
single, 177  
skel, 85  
sort, 47  
splashimage, 173  
ss, 230  
SSL, 293  
SSO, 258  
stderr, 54  
stdin, 54  
stdout, 54  
Sticky-bit, 132  
Strict, 277  
su, 98, , 248  
sudo, , 249  
SUID, 133, 140  
Super bloc, 112  
synchrone, 142  
systemd, 192

## T

Table des inodes, 113  
tac, 45  
tail, 46  
tar, 149, 152

---

Targeted, 277  
tee, 58  
TLS,  
top, 146  
touch, 39  
TTY, 24  
tubes, 57  
Types de fichiers, 120

## **U**

UEFI, 169  
UID, 24, 79  
umask, 135  
umount, 119  
unalias, 59  
untar, 154  
useradd, 84  
userdel, 87  
usermod, 85

## **V**

vgcreate, 109  
vgdisplay, 110  
visudo, 250

## **W**

wc, 50  
whatis, 26  
wheel, 251  
whereis, 51  
who, 32  
whoami, 32

## **Y**

YUM, 237