Table des matières

[Vue générale 3](#_Toc74642988)

[Caractéristiques. 3](#_Toc74642989)

[Tâches. 3](#_Toc74642990)

[Monoprocesseur. 3](#_Toc74642991)

[Multiprocesseur 3](#_Toc74642992)

[Multi-plateformes. 4](#_Toc74642993)

[Time-sharing. 4](#_Toc74642994)

[Intégration entrées – sorties système fichier. 4](#_Toc74642995)

[Réentrance. 4](#_Toc74642996)

[Système de développement. 4](#_Toc74642997)

[SGF hiérarchique (Système de Gestion de Fichier). 4](#_Toc74642998)

[Constitution du système. 5](#_Toc74642999)

[Différence entre devices et drivers. 6](#_Toc74643000)

[Technologie Udev. 7](#_Toc74643001)

[Sécurité multi-utilisateurs 8](#_Toc74643002)

[3 Volets 8](#_Toc74643003)

[Utilisateurs : 8](#_Toc74643004)

[Droits : 8](#_Toc74643005)

[Contrôle : 8](#_Toc74643006)

[Gestion des utilisateurs. 8](#_Toc74643007)

[Gestion des droits. 9](#_Toc74643008)

[Pratique 11](#_Toc74643009)

[Désactiver SeLinux 11](#_Toc74643010)

[Vérifier si les cartes réseau sont sur ON au démarrage des machines : 11](#_Toc74643011)

[Listes des services au démarrage : 11](#_Toc74643012)

[Propriété des services au démarrage : 11](#_Toc74643013)

[Propriété des différents disques dur : 11](#_Toc74643014)

[Lost+found : 11](#_Toc74643015)

[Etc/Fstab : 11](#_Toc74643016)

[Swap : 11](#_Toc74643017)

[Afficher les différents filesystems montés et les décrire : 11](#_Toc74643018)

[Modification de l’UID minimum : 11](#_Toc74643019)

[Affiché les différents UUID / Label des partitions 11](#_Toc74643020)

[FUSE 12](#_Toc74643021)

[SSHFS 12](#_Toc74643022)

[RAMDISK TMPFS 12](#_Toc74643023)

[RAMDISK SIMPLE 12](#_Toc74643024)

[Rappel CMD 13](#_Toc74643025)

# Vue générale

## Caractéristiques.

Un système d’exploitation est un ensemble de programmes coordonnant le fonctionnement d’un système informatique. Pour Linux, voici ces caractéristiques.

Multi-utilisateurs : Plusieurs utilisateurs peuvent accéder en même temps au système et travailler en apparente simultanéité. Ces utilisateurs peuvent y accéder en local (tty) et en réseau (pts)

Multi-tâches : Le système est capable de gérer plusieurs tâches en apparente simultanéité. Le système est préemptif, il libère les ressources nécessaires à l’application. En réalité, les tâches ne s’effectuent pas en même temps. Dans le kernel on trouve un « ordonnanceur » ou encore « Schedule » qui est chargé de gérer le temps alloué à chaque tâche. Principe du tourniquet, chacun son tour. Chaque process a un time slice qui lui est propre.

## Tâches.

Un programme = suite d’instructions pour la réalisation d’un traitement. Statique. Un processus = l’image du programme qui s’exécute. Dynamique. 3 types de processus : processus lourd, sous-processus lourds ou threads.

Lourd → Le programme est exécuté de A à Z dans un seul µP. Si un deuxième µP existait, il ne servirait à rien. 1 seul fil d’exécution.

Sous-Lourd → Notion d’inter-processus et d’IPC (communication). L’IPC découpe de gros process en de plus petits et permet au sous-process de communiquer entre eux en passant par une couche IPC. Celle-ci est gérée par l’OS. Si 2 µP, chacun se charge d’un process. 2 fils d’exécution.

Thread → La communication se faire directement, sans passer par une autre couche. Mais cette communication est à charge du développeur. En threading 1 tache = 1 fil d’exécution.

Commande lscpu pour connaitre l’architecture cpu du système. Fichier /proc/cpinfo.

## Monoprocesseur.

Linux peut ordonnancer aussi bien 1 µP que plusieurs. Dans le cas d’un monoprocesseur, le Schedule donne à chaque tâche sa tranche de temps (time slice) selon sa priorité. Le Schedule gère aussi l’état dynamique des processus. Plus une tâche est prioritaire, plus du temps lui est alloué. La commande nice permet de lancer un process avec un niveau de priorité différent. La commande ps -el pour connaitre le niveau de priorité PRI des process et le modificateur nice NI. De base PRI = 80, NI = 0. Si je lance nice -20 « commande », le PRI sera de 60, le NI de -20. PRI peut osciller entre 60 et 100. 60 = prioritaire, 100 = dernier de la file.

Multiprocesseur.

Autant de tâches simultanées qu’il y a de µP. Architecture SMP : Symetric Multi Processors. Tous les processeurs peuvent accéder à toutes les ressources du système. Semblable à un entonnoir. Mais s’il est trop rempli, overflow et des données peuvent être perdues. SMP marche donc bien lorsque peu de µP. cat /proc/cpuinfo ou lscpu. Si besoin est de travailler avec ~ 8 µP, mieux vaut utiliser NUMA.

NUMA : Non Uniform Memory Access. Dans cette architecture, les zones mémoires sont séparées et situées en différents endroits. Division des µP en plusieurs nœuds possédant leurs propres bus de communication avec la RAM. (Physiquement, 1 seul RAM). Existence de route de secours pour que les µP d’un nœud peuvent écrire dans la RAM de l’autre en cas de problème. Numactl --hardware.

## Multi-plateformes.

Une fois le noyau compilé, Linux est supporté par beaucoup d’architectures CPU. Différences entre 32 bits et 64 bits. Une petite zone mémoire de quelques bytes, un RI Registre d’Instruction et un RA Registre d’Adresses. La longueur du RA et RI sont pareils et valent 32 bits pour des processeurs 32 bits et 64 bits pour un P de 64 bits.

32 bits : Un set d’instruction est présent, écrit en langage assembleur et détermine les fonctions, applications et instruction disponibles. La suite des instructions assembleur est compilé, ça donne un programme binaire de la même taille. Quand le µP exécute le programme, il le fait du haut vers le bas. Dans un premier temps, la première instruction en binaire va être saisie et rangée dans le registre d’instruction avec la même longueur de bits. Dans un second temps, exécution de l’instruction. Dans un troisième temps, il va chercher l’instruction suivante dans le binaire et le range dans son RI. Quatrième temps, il l’exécute et ainsi de suite. Donc un µP de 32 bits possède un RI de 32 bits et ne peut exécuter des instructions plus longues que 32 bits. Quant au RA, il contient des adresses mémoires (RAM). Il s’agit des adresses réelles des éléments telle que des variables par hasard. Pareil que RI, une adresse ne peut-être codée que sur 32 bits. Ainsi, dans un µP 32 bits, on a 232 – 1 cellule de mémoire, environ 4 Go de mémoire utilisable. Si plus, ne sert à rien. Capacité d’adressage = 4 Go.

64 bits : Même logique, mais en 64 bits. Une adresse peut être codé sur 64 bits, donc 264 – 1 cellule de RAM adressable, environ 1,8 x 1010 Go de RAM. En 64 bits, on parle d’architecture x86\_64. Dans cette architecture, un RI de 64 bits ofc mais surtout, un set d’instruction étendu. On retrouve de base toutes les instructions assembleurs 32 bits de l’architecture x86. À ça, on rajoute des instructions pouvant aller jusque 64 bits. Et en plus, des instructions regroupant certaines instructions de x86. Pour un même programme, moins d’instructions, car re groupable. En x86, un programme de 7 instructions peut devenir un programme de seulement 3 par exemple. Exécution dans le même sens, de haut en bas. On profite ainsi du plus grand RI, et un programme écrit en 64 bits sera ainsi plus rapide à exécuter (moins de « lignes », moins d’instructions). Jamais plus lent. Possible égalité, si pas de regroupement d’instruction. Mais souvent plus rapide. CentOS 8 que pour du 64 bits. Si on veut le faire tourner sur un µP 32 bits, c’est la merde car le RI pourraient contenir des instructions trop longues.

## Time-sharing.

Les ressources sont réparties équitablement entre les user et les process. Si même niveau de priorité, même time slice et même part de RAM. Prioritaire, plus de RAM et de temps.

## Intégration entrées – sorties système fichier.

Un périphérique = un fichier spécial. Le terminal de base = /dev/tty1. Je peux envoyer un coucou sur le 2ème terminal à l’aide de « echo coucou > /dev/tty2 » une redirection comme s’il s’agissait d’un fichier.

## Réentrance.

Si un programme est utilisé par plusieurs processus, le programme ne se démultiplie pas en RAM. Imaginons que plusieurs utilisateurs compilent un programme C (gcc). Au lieu d’avoir le binaire de gcc présent n-fois, tous les user se partagent le binaire de gcc pour effectuer leurs processus. Autre exemple, le bash. Si plusieurs user, un seul binaire bash en RAM.

## Système de développement.

Beaucoup d’outils pour développer. Éditeur de base : vi, vim ou encore mcedit, nano. Des compilateurs comme gcc, g++ pour du C, C++. Et aussi des IDE plus complet comme Code Blocks, Eclipse, etc…. gcc -o *progname* *sourcename.c* compilation classique d’un programme C. gcc -S -o *progname* *sourcename.c* compilation d’un programme C en langage assembleur.

## SGF hiérarchique (Système de Gestion de Fichier).

En anglais, un file system. C’est lui qui gère l’organisation logique des données dans un DD pour les localiser et les utiliser. Il assure la correspondance entre l’organisation logique et physique d’un fichier. Il est capable de dire où commence et où s’arrête des fichiers composés de blocs non-fragmentés (queue-leu-leu) ainsi que de donner la succession d’adresses de blocs dans le cas où le fichier est fragmenté. Taille de 1 bloc par défaut = 4ko. C’est plus rapide pour un DD classique mécanique une suite de blocs non-fragmentés.

Métadonnées d’un filesystem : boot block, super block, table des I-Nodes. Un I-Node = 128 bytes, possiblement des milliers. Dans le super block, les métadonnées du file system en lui-même. Les I-Nodes sont les métadonnées d’un fichier dans le FS. Le file system se trouve tout au début de la partition extended 4, avant les data proprement dits. Mount permet de voir les file system disponible. Les informations issues d’un ls -l proviennent des métadonnées du fichier, dans le I-Node associé (type de fichier, autorisations, propriétaire, taille…). ls -il permet de voir le numéro d’I-Node associé, celui dans lequel on retrouve les métadonnées. stat pour avoir plus de détails encore. Pour les file system de type extended on peut voir les métadonnées du FS lui-même. dumpe2fs pour visualiser le super block. Si je crée un nouveau fichier/modification → MAJ de la table des I-Nodes et MAJ du superblock grâce à un module (un process) en RAM qui mettra à jour les infos. lsmod pour visualiser la liste des modules en RAM.

Chaque type de filesystem possède son propre module nécessaire à son bon fonctionnement. Linux accepte un grand nombre de modules et donc de file system différent. Il existe différents type de filesystem, des non-journalisés, journalisés ou à snapshots. Les non-journalisés, problème possible lors d’une coupure de courant/arrêt du système pas propre. Exemple, extended 2, fat, hfs… A l’inverse, les journalisés sont plus protégés contre les coupures de courant. Exemples ext3, ext4, ntfs. Snapshot permet de prendre des instantanées, des « sauvegardes » du file system à une date précédant l’hypothétique corruption.

Chaque SGF doit fournir les objectifs suivants : intégrité, sécurité, permanence et datation. Intégrité : Ce qu’on lit correspond à ce qui a été écrit, pas de modifications inopinées. Possible chiffrement des partitions. Sécurité : Contrôle d’accès et des différentes autorisations et permissions des fichiers. Permanence : Conserver les infos mêmes en cas de problèmes (journalisé ou pas). Datation : savoir différencier et dire quand un fichier a été lu, quand il a été modifié et quand les métadonnées ont étés modifiées.

Chaque file system possède sa technique et un fichier qui regroupe tout ce qui est nécessaire.

Il y a une couche « virtuelle » qui gère les différents types de filesystem. Lorsque le noyau doit faire un appel system, le VFS aiguille correctement vers le bon filesystem capable de gérer la requête. Possible perte d’infos lors de transfert vers des file system non Unix. Possible utilisation des Access Control List et des quotas.

Arbre Standard. Des dossiers et une arborescence retrouvée sur beaucoup de machine Linux à base Red Hat. Le « / » représente le dossier racine, ce à quoi tout est rattaché. Le « /boot » est le dossier dans lequel on retrouve des dossiers et sous-dossiers ainsi que des fichiers, tout ce qui est nécessaire pour le démarrage de la machine. Faire attention à ce qu’on chipote là-dedans (vmlinuz = le noyau). Le « /home » les home directories des utilisateurs. Le « /root » la home directory de l’administrateur système.

Le « /usr » un dossier avec des sous-dossiers « bin », « lib », « lib64 », « sbin » et d’autres. Dans « /bin » on retrouve les binaires des commandes externes, utilisable par tous. Dans « /sbin » les binaires des commandes d’administrations, utilisable par le root. Dans « /lib » les modules de FS et des librairies 32 bits. Nécessaire au développement. Dans « /lib64 » pareil mais en 64 bits.

Le « /dev » avec ce qui est nécessaire à la gestion des périphériques. Dans « /var » tout ce qui est variable. Dans « /etc » les fichiers de config du système et des services. Dans « /tmp » des fichiers et dossiers temporaires. Dans « /mnt » des points de montage temporaire de FS. Dans « /media » le montage de médias. Dans « /proc », pas dans le DD mais en mémoire RAM.

## Constitution du système.

Architecture classique à 3 espace : utilisateur, noyau, matériel. Utilisateur là où tout le monde peut travailler tous les jours, là où les applications se jouent. Depuis l’espace U, on ne peut pas travailler directement sur le matériel, on est cloisonné. Si besoin est de modifier le matériel (enregistrer un fichier, modifier, etc…), ça passe par les appels systèmes de l’espace N. C’est le noyau et les appels systèmes qui font le lien entre l’espace U et M. Espace N sert aussi au bon fonctionnement global de la machine, il s’occupe des FS, de l’ordonnanceur, les piles et pilotes réseau, etc… 2 sous-couches dans N, une sous-couche ‘drivers/pilote logiciel’ et une sous-couche ‘contrôleur de disque/driver matériel’.

Noyau : Cœur du système. Contient les fonctions de base utilisées tout le temps et essentiel pour le bon fonctionnement. Noyau configurable, modulable et re-compilable en Linux. Il se trouve dans /boot sous le nom ‘vmlinuz’. Commande uname-r pour connaître la version.

Shell : Se divise en 2, une partie interpréteur de commande (programme écrite en assembleur) et une partie commandes internes (des commandes codés en binaire dans le Shell même. L’interpréteur de commande fournit un prompt qui attend qu’on rentre qqch. Après Enter, l’interpréteur regarde si cela correspond à quelque chose. Si oui, il charge ce qu’il a besoin en RAM. Erreur du binaire lorsque option pas bonne (ls -W), erreur de l’interpréteur lorsque mauvaise commande (LS -l). Commandes externes : interpréteur va chercher dans les binaires si ça correspond et la charge en RAM (entrée/sortie). Par contre, commandes internes, déjà en RAM, plus rapide car binaire déjà en RAM, pas d’entrée/sortie. Pour le manuel : man pour une commande externe, help pour une commande interne. Dans le fichier ~/.bash\_history donne l’historique des commandes exécutés par l’user. Commande history fait la même chose.

Shell configurable par plusieurs fichiers en plusieurs mode :

* Shell de connexion : celui obtenu après la saisie du login/mdp. Se retrouve dans le fichier /etc/passwd. Avant même la connexion effective, le Shell veut lancer les commandes suivantes : /etc/profile , ~/.bash\_profile, , (~/.bash\_login), (~/.profile). En se déconnectant , ~/.bash\_logout sera exécuté.
* Shell simple : Exécution du bash dans une fenêtre à part (en mode graphique, ouvrir un terminal) ou bien en tapant la commande bash, on obtient un bash simple. Il tentera d’exécuter ~/.bashrc s’il existe. Souvent, ~/.bash\_profile exécute ~/.bashrc aussi.
* Shell non-interactif : Lorsqu’on exécute un script, bash *scriptname*. Le bash s’instancie juste pour exécuter le script et ne fait rien d’autre, n’appelle pas de fichier supplémentaire. Une fois le script fini, ce bash se supprime.

Synoptique : Explication processus parents et fils.

Applications disponibles sur Linux. Une commande peut être considérés comme une application finalement. Beaucoup de logiciels graphiques dispos également, pas que de la ligne de commande (vive le clic-clic).

Messagerie : Pour écrire des messages aux autres utilisateurs de mon système (machine ou serveur). Commande write *username* pour envoyer un message, mesg *y/n* pour autoriser ou non l’envoi de message sur ma machine et wall pour écrire à ceux connectés. Commande mail (configurer le serveur SMTP d’abord) pour envoyer des mails en mode texte.

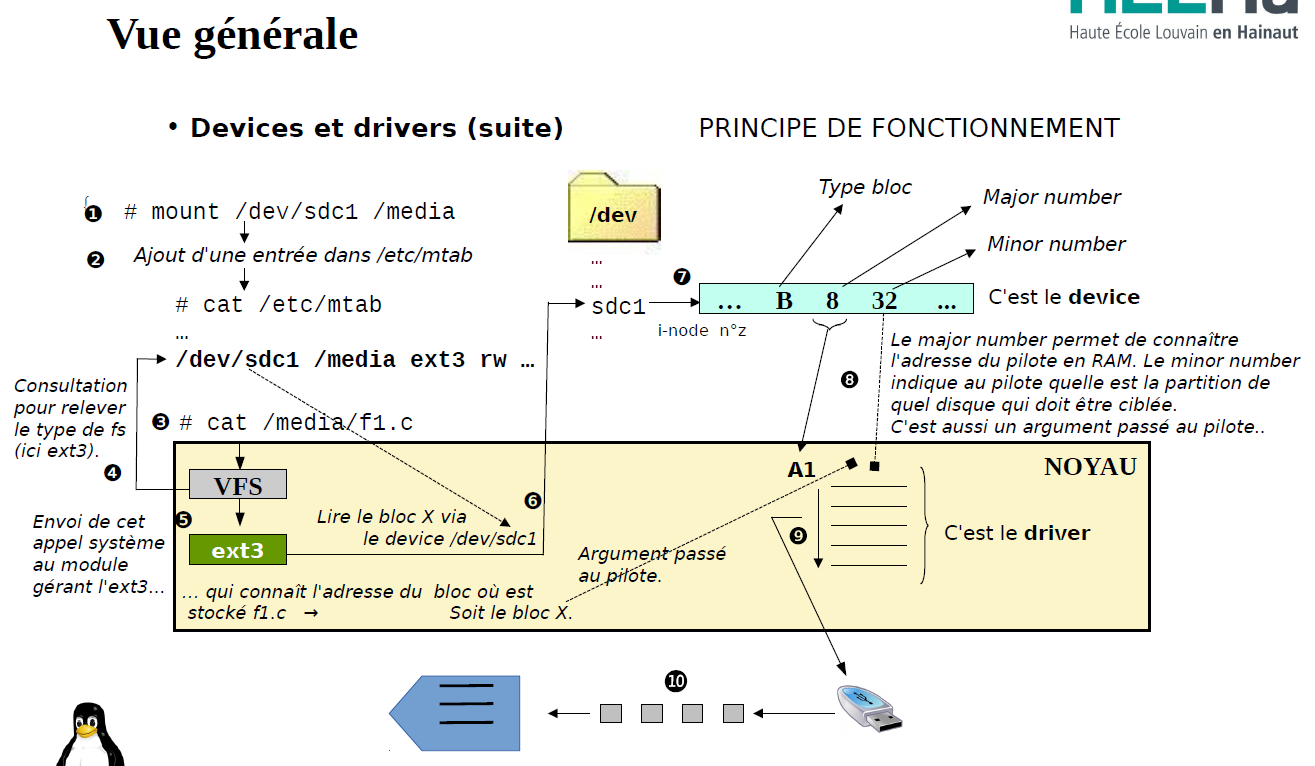
## Différence entre devices et drivers.

Dans tout Linux, dossier /dev avec la liste de tous les devices. Mais pas une taille, mais 2 nombres, un major number et un minor number. Dans /dev/sda, la liste des disques dures. Les devices ce sont des i-nodes reliés à aucun fichier.

En RAM, à partir d’une adresse données, il y a des routines implantés, des programmes et donc des drivers ou pilotes de périphérique en français. Chaque routine est spécialisée dans un type d’entrée/sortie. Major Number = périphérique dans son ensemble, le minor number = la sous-partie concernée. Le driver se charge d’envoyer les infos au contrôleur du disque et pilote les opérations.

2 types de devices : type bloc, type caractère. Par exemple, les tty (terminaux) sont des types caractère. Donc les drivers savent s’ils doivent travailler bloc par bloc ou bien caractère par caractère.

Une nomenclature à respecter pour les devices. /dev/sda pour le premier device. /dev/sda2 pour la deuxième partition de ce device. Major Number en lien avec le type de device. Et le Minor number qui part de 0 jusqu’au nombre total de partitions.



/etc/mtab existe dans toutes les distris Linux et s’update tout seul, assez dynamique.

## Technologie Udev.

Dans les noyaux Linux récents, technologie Udev présente de base. Si on liste les process actifs, il y a un process systemd-udevd. Lorsque machine éteinte, sur le DD, dans /dev il n’y a plus rien. En démarrant, chargement en RAM de systemd-udevd. Ce process va regarder tous les périphériques présents et les lister. Après repérage, il va créer les devices sda, sdb, etc. Création et destruction automatique. Remarque : udevd gère le Plug-and-play, on insère une clé USB et c’est reconnu automatiquement. le process udevd regarde en permanence s’il y a un nouveau périphérique branché et l’ajoute dans /dev.

# Sécurité multi-utilisateurs

## 3 Volets

Utilisateurs : L’admin système peut toujours ajouter autant de users qu’il veut. Par défaut, 1 seul user : root. Un utilisateur doit TOUJOURS faire partie d’au moins 1 ‘groupe’. Pour chaque user, un UID unique l’identifie, par exemple root a toujours un UID de 0. Un même utilisateur peut appartenir à plusieurs groupes. Le UID ne change pas selon le nom. Un groupe a un nom et un GID unique pour l’identifier. Pas de limite de taille pour les groupes. Un utilisateur peut représenter une application tournante (exemple : serveur web apache) sur le système.

Droits : Les droits portent sur des dossiers, sous-dossiers et fichiers présent dans un système. Dans un ls -l, en regard du nom, on retrouve les différents droits et permissions pour chaque entrée.

Contrôle : Le système sait qui peut faire quoi sur quel fichier ? Cela implique que le système est en permanence au courant de qui essaie d’accéder au fichier, il doit donc être absolument connecté. Pas possible d’accéder à un fichier sans être connecté au préalable. Le système possède un ‘annuaire local’ de tous les utilisateurs existants sur le système. Cet annuaire = 3 fichiers → /etc/passwd , /etc/group et /etc/shadow.

Gestion des utilisateurs → L’admin a à disposition plusieurs commandes pour gérer comme il le veut. Exemple : groupadd, groupdel, groupmod, useradd, userdel, usermod, passwd. Seul l’admin pour utiliser ces commandes. Quand on touche aux groupes, cela modifie le fichier /etc/group. Le reste des commandes modifie /etc/passwd et /etc/shadow. La commande passwd est particulière car un utilisateur peut aussi la lancer pour modifier son propre mdp. Il existe dans tout Linux un dossier /etc/skel. Dans ce dossier, on peut y placer ce qu’on veut. Et lorsqu’on créera un nouvel utilisateur, ce nouvel user aura une copie conforme de ce qui se trouve dedans dans sa home directory et en sera le propriétaire . La commande id permet à un utilisateur de connaitre sa situation, quel est son uid, son gid et la liste des groupes auxquels il appartient. Pour un user qui est dans plusieurs groupes, il peut changer de groupe actif grâce à la commande newgrp. Ne marche que si configurer dans plusieurs groupes. Enfin, fichier /etc/nsswitch.conf qui configure le service d’authentification, si on ne laisse que ‘files’ en face de passwd shadow et group, le service ne se basera que sur les fichiers locaux de la machine, il n’ira pas chercher autre part. Si on rajoute ‘nis’, si la connexion locale échoue, le système ira voir pour une connexion NIS (dans un annuaire distant, serveur NIS). Si on rajoute ‘db’, si les 2 connexions précédentes foirent, le service d’authentification ira voir dans une database s’il ne retrouve pas une correspondance.

Gestion des droits → Pour jouer sur la zone de ‘protection bits’, root peut utiliser plusieurs commande pour gérer les droits et les perms. Par exemple, chgrp, chown, chmod, umask. Mais un utilisateur peut aussi utiliser chgrp pour modifier le groupe auquel est affilié un fichier à conditions de faire partie des groupes nécessaires. Un utilisateur peut aussi utiliser chmod (droit lecture/écriture) ainsi que umask. Dans cette partie, on joue avec les droits permissions et propriétaires d’un fichier.

Contrôle. Le système test à chaque fois les permissions pour savoir si un utilisateur privilégié ou non pour accéder à un fichier. De deux manières, la table des I-Nodes contient tous les droits collant un fichier. Mais aussi, un vecteur état du processus qui permet de savoir qui lance un processus, qui en est le propriétaire. Comparaison entre les 2, si la personne qui a lancé un processus fait partie des personnes autorisés dans la table des I-Nodes → Ok, il y accède.

## Gestion des utilisateurs.

Dans /etc/passwd, différents champs. Chaque ligne = 1 compte utilisateur, humain ou non. Champs séparés par des « : », des champs peuvent être vide. Le premier champ = username. Le deuxième champ = avant, le mdp du user mais plus maintenant car tout le monde peut avoir accès à ce fichier. Troisième champ = UID unique. Quatrième champ : GID du groupe de connexion, celui dans lequel il tombe de base. Cinquième champ : des commentaires, optionnel. Sixième champ : la home directory de l’utilisateur, modifiable. Septième champ : La commande qui s’exécutera à la connexion, en règle générale : un shell.

Le contenu de /etc/passwd peut être modifié « en dur », à la main, mais aussi par les différentes commandes de gestions des user, ça aura un impact sur /etc/passwd.

Dans /etc/shadow, même chose que dans passwd, des champs séparés par des « : ». Il y a autant de lignes aussi, 1 par utilisateur humain ou non. Ce fichier ne peut être lu que par le root. Premier champ : nom du user. Deuxième champ : Entre $, un code, qui correspond à l’algo de hachage utilisé, suivi de « sel » jusqu’au prochain $, et enfin le mot de passe chiffré. → $code$sel$mdpchiffré. Si pas encore de mdp, on trouve « !! ». Les derniers champs : politique de changement du mdp. Possible d’imposer un changement de mdp 1 fois par mois par exemple.

Enregistrement d’un MDP : 1) le système génère un « sel » aléatoire. 2)Il la concatène au mdp en clair. 3) Hachage et génération de l’empreinte. 4) Enregistrement dans /etc/shadow des infos $code$sel$mdpchiffré. Dans /etc/login.defs, on peut changer l’algo de hachage dans ENCRYPT\_METHOD.

Vérif par le système : 1) Il cherche la ligne correspondante au login rentré. Si pas bon login, erreur. 2) Si login correct, il regarde l’algo de chiffrement nécessaire. 3) Il ajoute le sel au mdp rentré et concatène. 4) Il hache ce qu’il a obtenu, et génère l’empreinte. 5) Il vérifie et compare cette empreinte avec celle dans /etc/shadow. Si c’est bon → connexion.

Pourquoi sel ? Pour éviter que 2 mot de passes identiques soient chiffrés de la même manière. Même 2 mdp identique en clair ne seront pas chiffrés de la même manière grâce au sel aléatoire et différent.

Conseil pour mdp d’un utilisateur : Au moins 8 ou 12 caractère avec chiffres, lettres et caractères spéciaux. Pas utiliser des mdp trop personnel. Des mot de passes différents pour les différentes machines. Pas écrire ou divulguer le mdp. Pas réutiliser un mdp périmé. Pas taper un mdp si quelqu’un est à portée de vue. Pas laisser des users sans mdp, dangereux. Pas rentrer un mdp sans être sûr de la console sur laquelle on est. Exemple du cheval de Troie.

Dans /etc/group, toujours la même histoire, des lignes, des champs. Une ligne correspond à 1 groupe. Premier champ : Nom du groupe, nom unique. Deuxième champ : Toujours un x, possible de mettre un mot de passe pour un groupe. Troisième champ : GID du groupe. Quatrième champ ne représente PAS les membres du groupe, mais plutôt ceux pour qui c’est un groupe secondaire auxquels ils ont aussi accès.

Distinction dans la commande id : gid = le groupe dans lequel le user est MAINTENANT et groupes = tous les groupes auxquels elle a accès.

Dans un ls -l, on retrouve le propriétaire du fichier suivi du nom du groupe dans lequel il se trouve.

Quand on change de groupe, avec la commande newgrp, on rajoute une surcouche. En faisant en exit, on descend d’une couche.

## Gestion des droits.

Droits d’accès : pour chaque fichier, une zone de droits d’accès. Pour les voir, ls -l. Dans la liste, le premier caractère indique le type d’entrée, d pour un dossier et – pour un fichier ordinaire. Juste après, la ‘protection bits’ , un nombre de lien d’accès, le propriétaire, un nom de groupe, la taille, date de dernière modifie et ensuite nom. Toutes ces infos sont stockées dans l’I-Node du fichier/dossier sauf le nom.

Protection bits : 3 parties, les accès de l’utilisateurs, les accès du groupe, les accès de tous les autres. r = read, lire ; w = write, écrire ; x = exécutable. Suite de neufs caractères, 3 par parties. Dans l’I-Node, selon si c’est présent ou non, 1 ou 0 pour chaque charactère. On obtient donc 9 bits qui donnent les accès. Possible représentation octal pour chaque partie.

Read : Lire un fichier ; afficher le contenu d’un dossier,

Write : Écrire dans un fichier ; ajouter, effacer, modifier dans le répertoire.

Exécute : Exécuter un fichier (script ou autre) ;permission d’accéder aux entrées du répertoire.

Contrôle des accès : Linux regarde dans les I-Nodes. Lorsqu’un utilisateur rentre une commande, cette commande est toujours accompagnée par un vecteur état. Dans ce vecteur état, des informations sur ce process, le UID (qui lance la commande) et le GID notamment. Ainsi, Linux compare les informations des différents droit d’accès. Dans la séquence de 10 caractères à gauche d'un ls -l, vous avez le premier caractère qui est compté à part, et qui indique le type d'entrée (fichier/dossier). Ensuite la zone de 'protection bits'. C'est cette zone que je sépare en 3 champs de 3 bits chacun. Zone 1 : utilisateur en lui-même (u); Zone 2 : groupe concerné (g); Zone 3 : les autres (o). Est-ce que le UID et le GID correspondent aux premiers champs ? Si oui, ok, si non, il regarde le 3ème champ, s’il a la permission, il peut. Il vérifie d’abord son appartenance (UUID, GID ou autres) et puis les permissions accordées à ces gens-là. Cette vérif se fait pour toute l’arborescence. Si un user est dans plusieurs groupes, lors de la vérif, Linux envisage tous les groupes auxquels il appartient.

Manipulations des droits d’accès : chmod, chown, chgrp.

**chmod** : root + propriétaire.

Remplacer toutes les permissions : chmod 444 *fichier*→ utilisation d’un masque octal (100 100 100 / r--r--r--). chmod a=r fichier → tous les champs sont changés par r. chmod

Ajouter des permissions : chmod *zone*+*perm* fichier (ug+w pour les champs user et groupe on rajoute la permission write).

Retirer des permissions : chmod *zone*-*perm* (g-x pour les groupe, plus exécutable).

**chown** : Que par le root. Changer le propriétaire d’un fichier/dossier. chown *newuser* *fichier*.

**chgrp** : Root + ceux dans le groupe en question. chgrp *newgrp* *fichier*.

Option -R pour les 3 commandes, ce qui la rend récursive et applique les changements à toute l’arborescence.

**umask** : Par root et tous les users concernés. Permet de fixer des droits par défaut des dossiers et fichiers. Le masque par défaut 002, retirer le droit d’écriture aux autres. En modifiant le umask, tout ce qui sera créer par la suite permet de changer les permissions de ce qui est créé selon ce que le user veut. La commande umask en retire mais n’en rajoute pas. Max des dossiers = 7 7 7. Max des fichiers 6 6 6. Règle générale = X . NOT Y. Pour que ce soit le cas à chaque connexion, il devra modifier .bash/profile et spécifier dans l’user environnement le umask qu’il veut.

# Pratique

## Désactiver SeLinux

Modifier le fichier /etc/selinux/config

- SELINUX=enforcing 🡪 SELINUX=disable

## Vérifier si les cartes réseau sont sur ON au démarrage des machines :

- cd /etc/sysconfig/network-scripts

Vérifier ensuite avec un éditeur si l’option « ONBOOT » est sur « yes »

Ensuite redémarrer le réseau grâce à la commande :

- systemctl restart NetworkManager

## Listes des services au démarrage :

- systemctl list-unit-files

## Propriété des services au démarrage :

- systemctl disable/stop/enable/start/restart … *« Nom du service »*

## Propriété des différents disques dur :

- fdisk /dev/ … *« alias du disque »*

## Lost+found :

Uniquement dispo sur les machines linux sous ext… Permet de garder en mémoire quelque donnée qui aurai été perdue/corrompue. (Dans le cadre d’une panne de courant, une extinction a chaud …)

Une vérification est possible via la cmd : fsck

## Etc/Fstab :

Liste le différent disque ou point de montage que la machine possède, et leur attribue des « paramètres » ce qui permettrait par exemple de ne pas monter un disque automatiquement au démarrage ou de changer ses propriétés.

## Swap :

Système de fichier de secoure qui prendra le relaiclears sur la ram quand elle serra saturer.

- swapoff/on /dev/sda Permet de désactiver ou réactiver le swapping.

## Afficher les différents filesystems montés et les décrire :

- df -Th

## Modification de l’UID minimum :

Modifier la ligne « UID\_MIN xxxx » dans le fichier /etc/login.defs

## Affiché les différents UUID / Label des partitions

- blkid Afficher les UUID de chaque partitions.

- ls -l /dev/disk/by-uuid Afficher les UUID de chaque partitions.

- ls -l /dev/disk/by-label Affiche les LABEL de chaque partitions

## FUSE

Fuse est un paquet qui permet de monter virtuellement un file système potentiellement distant (Permet de monter une image d’un dossier distant sur la machine local. Exemple avoir accès a un dossier sur un serveur distant sans devoir se connecter via ssh/ftp …

Fuse fonctionne via d’autre paquet dépendant de lui tels que sshfs ou autre (Voir doc) il faut installer le paquet fuse ET les différentes dépendances pour que tout fonctionne.

## SSHFS

Installer le dépôt epel via : dnf install epel-release.

Est-ce que le packet fuse est installé ? : rpm -qa | grep QQC.

/etc/yum/repos.d contient la liste des dépôts et leur config.

Pour en activer un, modifier la ligne ENABLE. Une fois le dépôt PowerTools activé, dnf install fuse-sshfs -y

Pour SSH on installera sshfs et pour l’exemple on se connectera sur [Linux41@nat.bacisat.be](mailto:Linux41@nat.bacisat.be). La commande pour monter le point d’accès au file système est : - sshfs -p « coordonnée *du serveur »* (19207 [linux41@nat.bacisat.be](mailto:linux41@nat.bacisat.be)) « *Ou se trouve-il »* (:/home/linux41) « *Ou le monter »* (~/studix)

Pour démonter un fuse il faut entrer la commande :

- fusermount -u « *Le point de Montage »* (/root/studix)

## RAMDISK TMPFS

La création d’un ramdisk est importante dans le cadre de linux car les logs et les fichiers temporaire corresponde à énormément d’écriture sur un disque dur, et si celui-ci est un ssd, il est très mauvais de répéter des opérations tels que celle-ci.

- mkdir /mtn/« emplacement de montage »

- mount -t tmpfs -o size= « taille maximal » tmpfs /mtn/« emplacement de montage »

Pour qu’il soit présent au démarrage il suffit d’ajouter une ligne dans le /etc/fstab :

tmpfs /pointmontage tmpfs defaults, size=256M 0 0

## RAMDISK SIMPLE

Création du dossier ou le ramdisk serra monter a la fin de la manip :

- mkdir /mtn/ramdisk

Attribution d’un espace disque dans la ram, en plaçant des 0 logique a l’intérieur, dans l’emplacement « /dev/ram0 » avec pour espace 50Mo et des secteurs de 1024octet.

- dd if=/dev/zero of=/dev/ram0 count=50000 bs=1024

Formatage de cette espace en ext3.

- mkfs.ext3 /dev/ram0

Montage du volume ram dans l’emplacement disque.

- mount /dev/ram0 /mtn/ramdisk

Pour qu’il soit présent au démarrage il suffit d’ajouter une ligne dans le /etc/rc.d/rc.local :

- dd if=/dev/zero of=/dev/ram0 count=50000 bs=1024

- mkfs.ext3 /dev/ram0

- mount /dev/ram0 /mtn/ramdisk

# Rappel CMD

|  |  |  |
| --- | --- | --- |
| CMD | Option | Description |
|  |  |  |
| Fdisk |  | Rentré dans l’utilitaire fdisk |
|  | -l | Affiche les disques et leur table de partition |
|  | -n | Crée une nouvelle partition |
|  | -w | Ecrire la table de partition (MBR) |
|  | -d | Supprimé une partition |
|  | -t | Modifier le type de partition |
|  | /dev/\* | Permet d’entrer dans l’utilitaire de partition sur le disque séléctionné |
|  | | |
| Systemctl | List-unit-files | Affiche les services |
|  | Disable/Enable | Désactive/Active un service |
|  | Stop/Start | Arrête/Démarre un service |
|  |  |  |
| Mount |  | Affiche tous ce qui est monté actuellement dans l’os |
|  | -a | Monte tout ce qui est possible de monter automatiquement (Point de références : fstab) |
|  | « Chemin d’un disque »  🡪 « Chemin de montage » | Monte le disque sélectionné dans l’emplacement choisi |
|  | -t | Permet de choisir le type de filesystem a utilisé |
|  | -o | Attribuer des options supplémentaires tels que : « ro » pour Read Only par ex. |
|  | -L | Liste les partitions du disque via son Label |
|  | -U | Liste les partitions du disque via son UUID |
|  |  |  |
| Umount | -a | Démonte toutes les partitions à l’exception de « / » |
|  | « Chemin d’un disque » | Démonte le disque spécifier |
|  |  |  |
| swapon | « Chemin de la swap » | Active la zone de swap |
|  |  |  |
| swapoff | « Chemin de la swap » | Désactive la zone de swap |
|  |  |  |
| Mkfs.\* | « Chemin d’un disque » | Formate un disque via FS choisi |
|  | -c | Formate lent d’un disque (vFat) |
|  | -n | Ajoute un Label au disque (vFat) |
|  | -L | Ajoute un Label au disque (NTFS) |
|  | -s | Défini le nombre de secteur par cluster |
|  |  |  |
| e2label | QQC /dev/\* | Donne le label QQC au disque sélectionné |
|  | /dev/\* | Affiche le label du disque sélectionné |
|  |  |  |
| df | -T | Affiche le type de partition |
|  | -h | Affiche la place disponible lisible pour un humain |
|  |  |  |
| useradd |  | Ajoute un utilisateur |
|  | -d | Défini ma home directory de l’utilisateur |
|  | -u \* | Attribue l’uuid défini à l’utilisateur |
|  | -g \* | Attribue un groupe primaire a l’utilisateur |
|  | -G \* | Attribue un groupe secondaire a l’utilisateur |
|  | -s \* | Attribue un shell spécifique a l’utilisateur |
|  | -m | Permet de crée la home directory de l’utilisateur si celle-ci n’existe pas |
|  | -c \* | Ajoute un commentaire sur l’utilisateur |
|  |  |  |
| userdel |  | Supprime un utilisateur l’utilisateur |
|  | -r | Supprime le groupe par defaut qui a été créé avec l’utilisateur |
|  |  |  |
| passwd |  | Attribue un mot de passe à un utilisateur |
|  | -S \* | Vérifie l’état de compte de l’utilisateur sélectionné |
|  | -l \* | Verrouille l’utilisateur sélectionné |
|  | -u \* | Déverrouille l’utilisateur sélectionné |
|  | -d | Ne pas attribuer de mdp a cette utilisateur |
|  |  |  |
| usermod | -a | Sert a ajouté quelque chose, utile pour ne pas réécrire tout un ensemble de groupe … |
|  | -g | Affecte le groupe principal attribuer à l’utilisateur |
|  | -G | Affecte les groupes secondaires attribuer à l’utilisateur |
|  | -u \* | Modifie l’uuid de l’utilisateur sélectionnée |
|  |  |  |
| Groupadd |  | Crée un groupe |
|  | -g | Attribue un GID |
|  |  |  |
| groupdel |  | Supprime un groupe existant |
|  |  |  |
| chmod |  | Joue sur les permissions d’un fichier |
|  | U – g – o - a | User – Groupe – Ôtre - All |
|  | + r – w - x | Read – Write – Xecute |
|  |  |  |
| chgrp |  | Joue sur les groupes des utilisateurs |
|  | -R | Récursif sur une arborescence |
|  |  |  |
| chown |  | Joue sur le propriétaire d’un fichier |
|  | -R | Récursif sur une arborescence |
|  |  |  |
| newgrp |  | Permet de connecter un utilisateur a un nouveau groupe |
|  |  |  |
| Chsh |  | Permet de changer d’interpréteur de commande |
|  |  |  |
| umask |  | Permet de changé les permission par défaut sur des dossiers/fichiers |