

Infrastructure technologique et virtualisation

## Module 4 : Administration système Linux

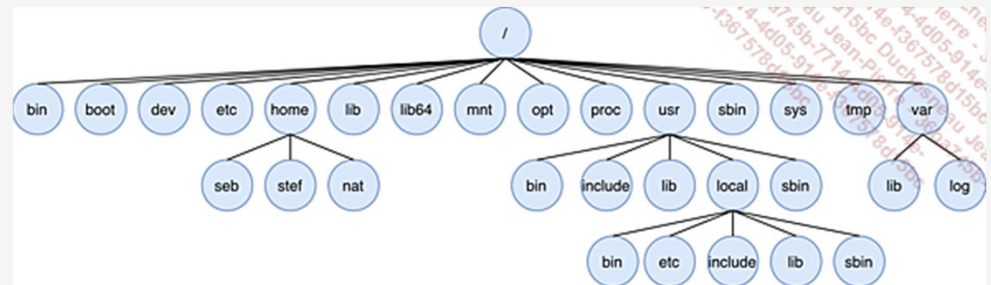
Gestion des fichiers et les droits d'accès

Source : ROHAUT, Sébastien (2020), Linux Maîtrisez l'administration du système (6<sup>e</sup> édition) . Édition  
Eni ISBN : 9782409025716

# Gestion des fichiers

Nous avons déjà vu qu'un système de fichiers, appelé communément File System ou FS, définit l'organisation des données sur un support de stockage, donc comment sont gérés et organisés les fichiers par le système d'exploitation.

**Linux** est, comme tout Unix, **un système d'exploitation entièrement orienté fichier**. Tout (ou presque) est représenté par un fichier, tant les données (fichiers de données de tout type comme une image ou un programme), que les périphériques (terminaux, souris, clavier, carte son, etc.) ou encore les moyens de communication (sockets, tubes nommés, etc.). On peut dire que le système de fichiers est le cœur de tout système Unix.



# Les divers types de fichiers

On distingue trois types de fichiers : ordinaires, catalogue, spéciaux.

## Les fichiers ordinaires ou réguliers

Les fichiers ordinaires sont aussi appelés fichiers réguliers, ordinary files ou regular files. Ce sont des fichiers tout à fait classiques qui contiennent des données. Par données, comprenez n'importe quel contenu :

- texte
- image
- audio
- programme binaire compilé
- script
- base de données
- bibliothèque de programmation
- etc.

Par défaut, rien ne permet de différencier les uns des autres, sauf à utiliser quelques options de certaines commandes (ls -F par exemple) ou la commande **file**.

```
jpduches@VM-DevOpsJPD:~$ file helloWorld.java
helloWorld.java: C++ source, ASCII text
jpduches@VM-DevOpsJPD:~$ file helloWorld.class
helloWorld.class: compiled Java class data, version 55.0
jpduches@VM-DevOpsJPD:~$
```

```
jpduches@srvdevops2jpd: ~
-rwxrwxr-x 1 jpduches jpduches 149 May 22 12:18 espace.sh
jpduches@srvdevops2jpd:~$ file espace.sh
espace.sh: Bourne-Again shell script, ASCII text executable
jpduches@srvdevops2jpd:~$ file /bin/bash
/bin/bash: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamic
ally linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=a6cb40
078351e05121d46daa768e271846d5cc54, for GNU/Linux 3.2.0, stripped
jpduches@srvdevops2jpd:~$
jpduches@srvdevops2jpd:~$
```

# Les catalogues

---

Les fichiers catalogues sont les répertoires, dossiers ou directory.

Les répertoires permettent d'organiser le disque dur en créant une hiérarchie. Un répertoire peut contenir des fichiers normaux, des fichiers spéciaux et d'autres répertoires, de manière récursive.

Un répertoire n'est rien d'autre qu'un fichier particulier contenant la liste des fichiers eux-mêmes présents dans ce répertoire, comme un index. Cette notion est très utile lorsque de la gestion des droits.

```
jpduches@VM-DevOpsJPD:~$ ls -al
total 183312
drwxr-xr-x 35 jpduches jpduches      4096 jun 23 10:24 .
drwxr-xr-x  9 root      root         4096 jun 21 11:07 ..
drwxrwxr-x  3 jpduches jpduches      4096 jun  6 12:28 .aspnet
-rw-r--r--  1 jpduches jpduches    11188 jun 21 11:20 .bash_history
-rw-r--r--  1 jpduches jpduches      220 mai 30 10:35 .bash_logout
-rw-r--r--  1 jpduches jpduches     3771 mai 30 10:35 .bashrc
drwxr-xr-x  4 jpduches jpduches      4096 jun 15 14:31 Bureau
drwx----- 16 jpduches jpduches      4096 jun  3 10:54 .cache
drwxr-xr-x 18 jpduches jpduches      4096 jun 11 09:00 .config
-rw-rw-r--  1 jpduches jpduches         0 jun 16 13:22 Devoirs
drwxr-xr-x  3 jpduches jpduches      4096 jun 11 10:28 Documents
drwxrwxr-x  6 jpduches jpduches      4096 jun  6 12:22 .dotnet
drwxrwxr-x  7 jpduches jpduches      4096 jun  3 10:54 eclipse
```

# Les fichiers spéciaux (info supplémentaire)

---

Le troisième type de fichier est le fichier spécial. Il existe plusieurs genres de fichiers spéciaux. Ils se trouvent principalement dans le répertoire `/dev` s'ils représentent des périphériques, mais peuvent être créés partout ailleurs, avec la commande **mknod**.

Ce sont principalement, mais pas seulement, des fichiers servant d'interface pour les divers périphériques. Ils peuvent s'utiliser, suivant le cas, comme des fichiers normaux. Un accès en lecture ou écriture sur ces fichiers est directement redirigé vers le périphérique (en passant par le pilote associé s'il existe). Par exemple si vous redirigez un fichier d'onde sonore (wave) vers le fichier représentant la sortie de la carte son, il y a de fortes chances que ce son soit audible par vos haut-parleurs.

## Voici une liste des types de fichiers spéciaux :

- Les fichiers en mode **bloc** permettent d'accéder à des périphériques fonctionnant par blocs de données, comme un disque, de manière aléatoire : on peut accéder directement à n'importe quel bloc.
- Les fichiers en mode **caractère** permettent d'accéder à un périphérique octet par octet, séquentiellement, en partant du début : le bloc `n` ne peut être accédé sans avoir accédé au bloc `n-1`. C'est le cas pour un lecteur de bandes.
- Les fichiers en mode **FIFO**, tubes ou pipes permettent à deux processus de s'échanger des données.
- Les fichiers en mode **socket** permettent aussi à des processus d'échanger des données, mais via des protocoles réseau classiques.
- Le **lien symbolique** peut parfois être considéré comme un fichier spécial, même s'il n'ouvre aucune interface particulière : il référence uniquement un autre fichier.

# Nomenclature des fichiers

---

- 255 caractères (y compris le suffixe)
- Sensible à la casse des caractères : Toto, TOTO, ToTo sont des noms différents.
- La plupart des caractères (les chiffres, les lettres, les majuscules, les minuscules, certains signes, les caractères accentués) sont acceptés, y compris l'espace.
- Cependant quelques caractères sont à éviter car ils ont une signification particulière au sein du shell : `&` ; `(` `)` `~` `<espace>` `\` `|` ``` `?` `-` (en début de nom).

Les noms suivants sont valides :

- Fichier1
- Paie.txt
- 123traitement.sh
- Paie\_juin\_2002.xls
- 8

Ces noms, bien que valides, peuvent poser des problèmes :

- Fichier\*
- Paie(decembre)
- Ben&Nuts
- Paie juin 2002.xls
- -f

# Les chemins

---

- Les chemins permettent de définir un emplacement au sein du système de fichiers. Un nom de fichier est ainsi généralement complété par son chemin d'accès. C'est ce qui fait que le fichier toto du répertoire rep1 est différent du fichier toto du répertoire rep2.
- Le / situé tout en haut s'appelle la racine ou root directory (à ne pas confondre avec le répertoire de l'administrateur root).
- **un chemin absolu** : [/home/toto/Docs/Backup/fic.bak](#)

Un chemin absolu ou complet :

- démarre de la racine, donc commence par un /.
- décrit tous les répertoires à traverser pour accéder à l'endroit voulu.
- ne contient pas de `.` ni de `..`.

# Chemin relatif

---

Un nom de chemin peut aussi être relatif à sa position courante dans le répertoire. Le système (ou le shell) mémorise la position actuelle d'un utilisateur dans le système de fichiers, le répertoire actif. Vous pouvez accéder à un autre répertoire de l'arborescence depuis l'emplacement actuel sans taper le chemin complet uniquement en précisant le chemin le plus court relativement à votre position actuelle au sein de l'arborescence.

- **Le point `.` représente le répertoire courant, actif. Il est généralement implicite.**
- **Les doubles points `..` représentent le répertoire de niveau inférieur.**

## Un chemin relatif :

- décrit un chemin relatif à une position donnée dans l'arborescence, généralement (mais pas toujours) depuis la position courante.
- décrit en principe le plus court chemin pour aller d'un point à un autre.
- peut contenir des points ou des doubles points.



## Exemples :

---

- `/usr/local/bin` est un chemin complet ou absolu.
- `Documents/Photos` est un chemin relatif : le répertoire Documents est considéré comme existant dans le répertoire courant.
- `./Documents/Photos` est un chemin relatif parfaitement identique au précédent, sauf que le répertoire actif (courant) est explicitement indiqué par le point. « `./Documents` » indique explicitement le répertoire Documents dans le répertoire actif.
- `/usr/local/../bin` est un chemin relatif : les `..` sont relatifs à `/usr/local` et descendent d'un niveau vers `/usr`. Le chemin final est donc `/usr/bin`.
- `../lib` est un chemin relatif : les `..` sont relatifs au répertoire courant puis descendent d'un niveau pour remonter dans `lib`. Si vous êtes dans `/usr/bin`, ce chemin représente `/usr/lib`.

# Le tilde

---

Le bash interprète le caractère tilde ~ comme un alias du répertoire personnel.

Les chemins peuvent être relatifs au tilde, mais le tilde ne doit être précédé d'aucun caractère.

Pour vous déplacer dans le répertoire tmp de votre dossier personnel d'où que vous soyez :

```
$ cd ~/tmp
```

Si vous entrez ceci, vous obtenez une erreur :

```
cd /~
```

# La commande CD

---

Pour vous déplacer dans les répertoires, vous utilisez la commande **cd** (change directory). La commande **pwd** (print working directory) que vous avez déjà rencontrée affiche le chemin complet du répertoire courant.

Si vous saisissez `cd .`, vous ne bougez pas. Le point sera très utile lorsque vous devrez spécifier des chemins explicites à des commandes situées dans le répertoire où vous êtes positionné.

Le `cd ..` remonte d'un niveau. Si vous étiez dans `/home/seb`, vous vous retrouvez dans `home`.

La commande **cd** sans argument permet de retourner directement dans son répertoire utilisateur.

```
[seb@client ~]$ pwd
/home/seb
[seb@client ~]$ cd ../public
[seb@client public]$ cd /usr/local/bin
[seb@client bin]$ cd ../../lib
[seb@client lib]$ cd
[seb@client ~]$ pwd
/home/seb
```

# Lister les fichiers et les répertoires

La commande **ls** permet de lister le contenu d'un répertoire (catalogue) en lignes ou colonnes. Elle supporte plusieurs paramètres dont voici les plus pertinents.

Paramètre	Signification
-l	Pour chaque fichier ou dossier, fournit des informations détaillées.
-a	Les fichiers cachés sont affichés (ils commencent par un point).
-d	Sur un répertoire, précise le répertoire lui-même et non son contenu.
-F	Rajoute un caractère à la fin du nom pour spécifier le type : / pour un répertoire, * pour un exécutable, @ pour un lien symbolique, etc.
-R	Si la commande rencontre des répertoires, elle rentre dans les sous-répertoires, sous-sous-répertoires, etc., de manière récursive.
-t	La sortie est triée par date de modification du plus récent au plus ancien. Cette date est affichée.
-c	Affiche / tri (avec -t) par date de changement d'état du fichier.
-u	Affiche / tri (avec -t) par date d'accès du fichier.
-r	L'ordre de sortie est inversé.
-i	Affiche l'inode du fichier.
-C	L'affichage est sur plusieurs colonnes (par défaut).
-1	L'affichage est sur une seule colonne.
-Z	Affichage des attributs du fichier lié au type de système de fichier ou au contexte de sécurité (selinux par exemple).

Le paramètre qui vous fournit le plus d'informations est le `-l` : il donne un certain nombre de détails sur les fichiers.

```
$ ls -l
total 15368
-rw-r--r-- 1 seb seb 8690 19 janv. 21:35 1I_1_controle_2_2004_2005.sxw
-rw-r--r-- 1 seb seb 8843 19 janv. 21:35 1I_2_controle_2_2004_2005.sxw
-rw-r--r-- 1 seb seb 54272 19 janv. 21:35 1i_classe1_controle1.doc
-rw-r--r-- 1 seb seb 52736 19 janv. 21:35 1ip_controle_2004_1.doc
-rw-r--r-- 1 seb seb 7452 19 janv. 21:35 1IP_controle_3_2005.sxw
-rw-rw-r-- 1 seb seb 21497 19 janv. 21:35 projet_4SRS_2013.odt
...
```

La ligne total indique la taille totale en blocs de 1024 octets (ou 512 octets si une variable appelée `POSIXLY_CORRECT` est positionnée) du contenu du répertoire. Cette taille est celle de l'ensemble des fichiers ordinaires du répertoire et ne prend pas en compte les éventuels sous-répertoires et leur contenu (pour ceci, il faudra utiliser la commande `du`).

Vient ensuite la liste détaillée de tout le contenu.

-rw-r--r--	1	seb	users	69120	sep 3 2006	3i_rattrapage_2006.doc
1	2	3	4	5	6	7

- 1 : Le premier caractère représente le type de fichier (- : ordinaire, d : répertoire, l : lien symbolique...) ; les autres, par blocs de trois, les droits pour l'utilisateur (rw-), le groupe (r-) et tous (r-). Les droits sont expliqués au chapitre Les disques et le système de fichiers.
- 2 : Un compteur de liens (chapitre Les disques et le système de fichiers).
- 3 : Le propriétaire du fichier, généralement celui qui l'a créé.
- 4 : Le groupe auquel appartient le fichier.
- 5 : La taille du fichier en octets.
- 6 : La date de dernière modification (parfois avec l'heure), suivant le paramètre (t, c, u).
- 7 : Le nom du fichier.

# Les liens symboliques

---

Vous pouvez créer des liens, qui sont un peu comme des raccourcis. Un lien est un fichier spécial contenant comme information un chemin vers un autre fichier. C'est une sortie d'alias. Il existe deux types de liens : le **lien dur** (hard link) que vous verrez plus loin, lors de l'étude des systèmes de fichiers, et le lien symbolique (soft link) qui correspond à la définition donnée.

Il est possible de créer des liens symboliques vers n'importe quel type de fichier, quel qu'il soit et où qu'il soit. La commande de création des liens symboliques ne vérifie pas si le fichier pointé existe. Il est même possible de créer des liens sur des fichiers qui n'existent pas avec le paramètre `-f`. **ln -s fichier lien**

```
jpduches@VM-DevOpsJPD:~$ ln -s Documents/Projects/laboratoire1/ Web
drwxrwxr-x  3 jpduches jpduches  4096 jun  3 10:56 .vscode
lrwxrwxrwx  1 jpduches jpduches   32 jun 23 11:18 Web -> Documents/Projects
/laboratoire1/
```

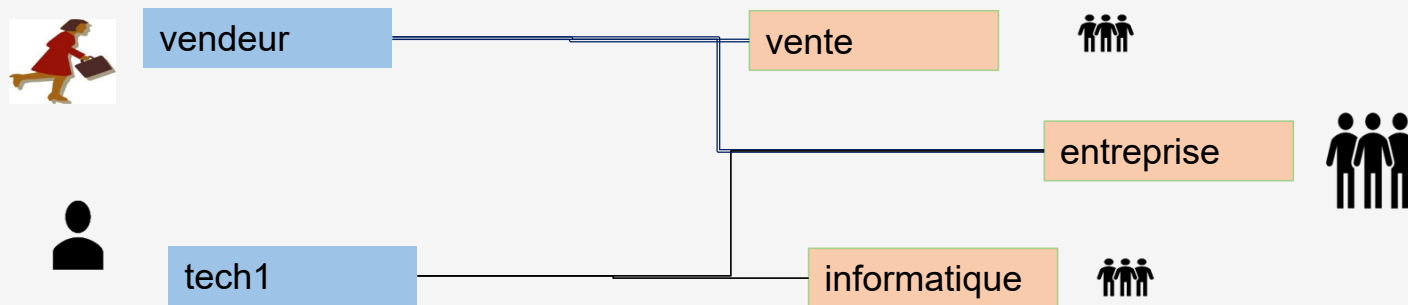
Le cas échéant le lien se comportera à l'identique du fichier pointé avec les mêmes permissions et les mêmes propriétés :

- si le fichier pointé est un programme, lancer le lien lance le programme.
- si le fichier pointé est un répertoire, un `cd` sur le lien rentre dans ce répertoire.
- si le fichier pointé est un fichier spécial (périphérique), le lien est vu comme périphérique.
- etc.

```
jpduches@VM-DevOpsJPD:~$ cd Web
jpduches@VM-DevOpsJPD:~/Web$ pwd
/home/jpduches/Web
jpduches@VM-DevOpsJPD:~/Web$
```

# La gestion des usagers et des groupes

---

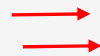


## ■ Les comptes usagers peuvent/doivent être associés à des groupes

- Exemple : 2 comptes à créer: **tech1** et **vendeur**.
- L'utilisateur **vendeur** appartient au groupe **vente**
- L'utilisateur **tech1** appartient au groupe **informatique**
- les deux utilisateurs **vendeur** et **tech1** appartiennent au groupe **entreprise**
- L'administrateur **ubuntuadmin** appartient au groupe **adm; sudo; entreprise**

# Création de comptes

Forme console **\$sudo adduser** <nom>



```
ubuntuadmin@u7654321:~$ sudo adduser tech1
[sudo] Mot de passe de ubuntuadmin :
Ajout de l'utilisateur « tech1 » ...
Ajout du nouveau groupe « tech1 » (1001) ...
Ajout du nouvel utilisateur « tech1 » (1001) avec le groupe « tech1 » ...
Création du répertoire personnel « /home/tech1 »...

Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur tech1
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
  Nom complet []: Technicien informatique
  N° de bureau []: 1234
  Téléphone professionnel []: 111-222-3333
  Téléphone personnel []:
  Autre []:
Ces informations sont-elles correctes ? [O/n] █
```

## Attention

Le pipe (|) permet de chaîner des processus de sorte que la sortie du processus alimente l'entrée du suivant.

2 Vérifications : ↓

**cat** /etc/passwd | **grep tech**

**cat** /etc/group | **grep tech**

```
ubuntuadmin@u7654321:~$ cat /etc/passwd | grep tech
tech1:x:1001:1001:Technicien informatique,1234,111-222-3333,:/home/tech1:/bin/bash
ubuntuadmin@u7654321:~$
```



# Les utilisateurs :

```
ubuntuadmin@u7654321:~$ cat /etc/passwd | grep tech
tech1:x:1001:1001:Technicien informatique,1234,111-222-3333,:/home/tech1:/bin/bash
ubuntuadmin@u7654321:~$
```

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

**UID** : User ID.

**GID** : Group ID.

Les UID et les GID sont en principe uniques. Le login (nom d'utilisateur) est unique.

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

## L'utilisateur dispose des attributs de base suivants :

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

# Les UID et les login

---

Les **UID** d'une valeur inférieure à 100 sont en principe associés à des comptes spéciaux avec des droits étendus. Ainsi l'UID de root, l'administrateur, est 0. Selon les distributions, à partir de 100, 500 ou 1000, et ce jusqu'au maximum 65535 ( $2^{16}-1$ ), ce sont les UID des utilisateurs sans pouvoirs particuliers.

La plupart des Unix, dont Linux, peuvent utiliser des UID sur 32 bits signés, la limite étant donc supérieure à quatre milliards (les compagnies disposant de plus de 65535 employés ne rencontrent donc plus de problèmes). Ces paramètres peuvent être modifiés par le contenu du fichier **/etc/login.defs**.

---

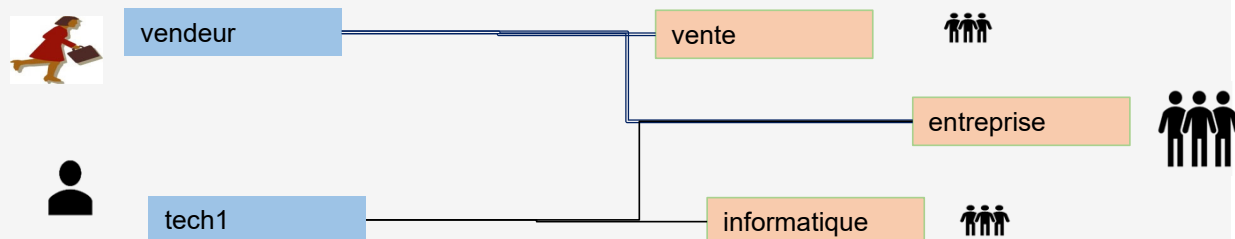
Un **login** a en principe une taille de 8 caractères. Linux et d'autres systèmes acceptent une taille plus grande, mais avec la plupart des commandes l'affichage ou la gestion des logins sont limités à 8 caractères. La taille maximale peut être obtenue ainsi :

```
$ getconf LOGIN_NAME_MAX  
256
```

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

# Création de groupes

\$ **sudo addgroup** <nom>



```
adminubuntu@u7654321 :~$ sudo addgroup entreprise
adminubuntu@u7654321 :~$ sudo addgroup informatique
adminubuntu@u7654321 :~$ sudo addgroup vente
adminubuntu@u7654321 :~$
```


Vérification: \$ **cat /etc/group**

```
ubuntuadmin@u7654321:~/Documents/Travail1$ cat /etc/group | tail --lines=3
entreprise:x:1001:
informatique:x:1002:
vente:x:1003:
ubuntuadmin@u7654321:~/Documents/Travail1$
ubuntuadmin@u7654321:~/Documents/Travail1$
```

# Les groupes

---

```
ubuntuadmin@u7654321:~/Documents/Travail1$ cat /etc/group | tail --lines=3
entreprise:x:1001:
informatique:x:1002:
vente:x:1003:
ubuntuadmin@u7654321:~/Documents/Travail1$
ubuntuadmin@u7654321:~/Documents/Travail1$
```

- Chaque utilisateur fait partie d'au moins un groupe.
- Un groupe regroupe des utilisateurs, il peut en contenir un ou plusieurs. 
- Comme pour les logins, le GID du groupe accompagne toujours l'utilisateur pour le contrôle de ses droits.
- Les groupes sont aussi des numéros (GID). Il existe des groupes spécifiques pour la gestion de certaines propriétés du système et notamment l'accès à certains périphériques

```
jpduches@VM-SEJPD:~$ cat /etc/group |grep adm
adm:x:4:syslog,jpduches
lpadmin:x:120:jpduches
jpduches@VM-SEJPD:~$
```

# Résumé des commandes

---

Ajouter un compte **adduser** *<compte>*

Ajouter un groupe **addgroup** *<groupe>*

Ajouter un compte à un groupe **usermod -a -G** *<nom\_groupe>* *<nom\_compte>*

\*\*\*\*Retirer un compte **deluser** *<compte>*

\*\*\*\*Retirer un compte du groupe **deluser** *<compte>* *<groupe>*

# Les droits de base

---

- Chaque fichier ou répertoire se voit attribuer des droits qui lui sont propres, des autorisations d'accès individuelles. Lors d'un accès le système vérifie si celui-ci est permis.
- À sa création par l'administrateur, un utilisateur se voit affecter un UID
- Chaque groupe possédant un identifiant unique, le GID
- La commande `id` permet d'obtenir ces informations

```
jpduches@VM-SEJPD:~$ id
uid=1000(jpduches) gid=1000(jpduches) groupes=1000(jpduches),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
jpduches@VM-SEJPD:~$
```

# Gestion de la sécurité des fichiers

- Premier symbole

d → directory

- → fichier

l → lien symbolique

```
jpduches@VM-SEJPD:~/Exercices$ ls -al
total 20
drwxrwxr-x 5 jpduches jpduches 4096 nov 11 11:51 .
drwxr-xr-x 18 jpduches jpduches 4096 nov 11 11:47 ..
drwxrwxr-x 4 jpduches jpduches 4096 nov 11 11:48 Exercice1
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice2
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice3
-rw-rw-r-- 1 jpduches jpduches  0 nov 11 11:50 fichier1.txt
-rw-rw-r-- 1 jpduches jpduches  0 nov 11 11:51 fichier2.txt
lrwxrwxrwx 1 jpduches jpduches 12 nov 11 11:51 fichier3.txt -> fichier1.txt
jpduches@VM-SEJPD:~/Exercices$
```

# gestion de la sécurité

permissions

Droits	Fichier	Répertoire
<b>r (Read)</b>	Lire le contenu	Lister ( <b>cd</b> , <b>ls</b> , <b>cat</b> , . . .)
<b>w (write)</b>	Modifier le contenu	Modifier ( <b>mkdir</b> , <b>rmd</b> , <b>nano</b> , . . . )x
<b>x (eXécute)</b>	Exécuter le fichier	( <b>cd</b> ) Peut changer de repertoire seulement si le repertoire parent a le droit x

```
tech1@7654321:~$ ls -l doc1.txt
-rwxrw-r-- 2 tech1 informatique 4996 Nov 12 10:23 doc1.txt
```

Loupe

-	rwx	rw-	r--	2	tech1	informatique	. . .	doc1.txt
								
	proprio				groupe			

Donc: tech1 est propriétaire: a les droits Lire/Écrire/Exécuter doc1.txt  
toutes les personnes du groupe informatique peuvent Lire/Écrire doc1.txt  
toutes les autres personnes enregistrés dans Linux peuvent Lire doc1.txt



## Modification des droits

---

Lors de sa création, un fichier ou un répertoire dispose de droits par défaut. Utilisez la commande **chmod** (change mode) pour modifier les droits sur un fichier ou un répertoire. Il existe deux méthodes pour modifier ces droits : ***par la forme symbolique et par la base 8***. Seul le propriétaire d'un fichier peut en modifier les droits (plus l'administrateur système). Le paramètre -R change les droits de manière récursive.

Forme symbolique →

Base 8 →

```
jpduches@VM-SEJPD:~/Exercices$ ls -l fichier2.txt
-rw-rw-r-- 1 jpduches tech1 0 nov 11 11:51 fichier2.txt
jpduches@VM-SEJPD:~/Exercices$ chmod g+x fichier2.txt
jpduches@VM-SEJPD:~/Exercices$ ls -l fichier2.txt
-rw-rwxr-- 1 jpduches tech1 0 nov 11 11:51 fichier2.txt
jpduches@VM-SEJPD:~/Exercices$ chmod 664 fichier2.txt
jpduches@VM-SEJPD:~/Exercices$ ls -l fichier2.txt
-rw-rw-r-- 1 jpduches tech1 0 nov 11 11:51 fichier2.txt
jpduches@VM-SEJPD:~/Exercices$
```

## Par symbole (Formes symbolique)

La syntaxe est la suivante :

`chmod modification Fichier1 [Fichier2]`

- S'il faut modifier des droits de l'utilisateur, utiliser le caractère **u**.
- Pour les droits du groupe le caractère **g**.
- Pour les droits des autres utiliser le caractère **o**.
- Pour ajouter des droits utiliser le caractère **+**
- Pour retirer des droits utiliser le caractère **-**
- Enfin le droit d'accès **r, w, x**.

```
jpduches@VM-SE3PD:~/Exercices$ ls -l
total 12
drwxrwxr-x 4 jpduches jpduches 4096 nov 11 11:48 Exercice1
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice2
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice3
-rw-rw-r-- 1 tech1 jpduches 0 nov 11 11:50 fichier1.txt
-rw-rw-r-- 1 jpduches tech1 0 nov 11 11:51 fichier2.txt
lrwxrwxrwx 1 jpduches jpduches 12 nov 11 11:51 fichier3.txt -> fichier1.txt
jpduches@VM-SE3PD:~/Exercices$ chmod g+x fichier1.txt
chmod: modification des droits de 'fichier1.txt': Opération non permise
jpduches@VM-SE3PD:~/Exercices$ chmod g+x fichier2.txt
jpduches@VM-SE3PD:~/Exercices$ ls -l
total 12
drwxrwxr-x 4 jpduches jpduches 4096 nov 11 11:48 Exercice1
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice2
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice3
-rw-rw-r-- 1 tech1 jpduches 0 nov 11 11:50 fichier1.txt
-rw-rw-r-- 1 jpduches tech1 0 nov 11 11:51 fichier2.txt
lrwxrwxrwx 1 jpduches jpduches 12 nov 11 11:51 fichier3.txt -> fichier1.txt
```

# Changer de propriétaire et de groupe

Il est possible de changer le propriétaire et le groupe d'un fichier à l'aide des commandes **chown** (change owner) et **chgrp** (change group). Le paramètre -R change la propriété de manière récursive.

Seul root a le droit de changer le propriétaire d'un fichier. Mais un utilisateur peut changer le groupe d'un fichier s'il fait partie du nouveau groupe.

```
jpduches@VM-SEJPD:~/Exercices$ chown tech1 fichier1.txt
chown: modification du propriétaire de 'fichier1.txt': Opération non permise
jpduches@VM-SEJPD:~/Exercices$ sudo chown tech1 fichier1.txt
jpduches@VM-SEJPD:~/Exercices$ ls -l
total 12
drwxrwxr-x 4 jpduches jpduches 4096 nov 11 11:48 Exercice1
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice2
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice3
-rw-rw-r-- 1 tech1    jpduches    0 nov 11 11:50 fichier1.txt
-rw-rw-r-- 1 jpduches jpduches    0 nov 11 11:51 fichier2.txt
lrwxrwxrwx 1 jpduches jpduches   12 nov 11 11:51 fichier3.txt -> fichier1.txt
jpduches@VM-SEJPD:~/Exercices$ sudo chgrp tech1 fichier2.txt
jpduches@VM-SEJPD:~/Exercices$ ls -l
total 12
drwxrwxr-x 4 jpduches jpduches 4096 nov 11 11:48 Exercice1
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice2
drwxrwxr-x 2 jpduches jpduches 4096 nov 11 11:46 Exercice3
-rw-rw-r-- 1 tech1    jpduches    0 nov 11 11:50 fichier1.txt
-rw-rw-r-- 1 jpduches tech1      0 nov 11 11:51 fichier2.txt
lrwxrwxrwx 1 jpduches jpduches   12 nov 11 11:51 fichier3.txt -> fichier1.txt
jpduches@VM-SEJPD:~/Exercices$
```

# Changer de propriétaire et de groupe

---

Il est possible de changer le propriétaire et le groupe d'un fichier à l'aide des commandes **chown** (change owner) et **chgrp** (change group). Le paramètre -R change la propriété de manière réursive.

Seul root a le droit de changer le propriétaire d'un fichier. Mais un utilisateur peut changer le groupe d'un fichier s'il fait partie du nouveau groupe.

```
chown utilisateur fic1 [Fic2...]  
chgrp groupe fic1 [Fic2...]
```

```
$ chgrp video fic1  
$ ls -l fic1  
-rwxr-xr-x 1 seb video 0 mar 21 22:03 fic1
```

# Droits d'accès étendus

---

## SUID et SGID

Il est possible d'établir des **droits d'accès étendus** à l'exécution d'une commande. Ces droits d'accès étendus appliqués à une commande permettent à cette commande de s'exécuter avec les droits du propriétaire ou du groupe d'appartenance de la commande, et non plus avec les droits de l'utilisateur l'ayant lancée.

L'exemple le plus simple est le programme **passwd** permettant de changer son mot de passe. Si la commande était exécutée avec les droits d'un utilisateur classique, **passwd** ne pourrait pas ouvrir et modifier les fichiers **/etc/passwd** et **/etc/shadow** :

```
$ ls -l /etc/passwd  
-rw-r--r-- 1 root root 1440 fév 24 10:35 /etc/passwd
```

Vous constatez que ce fichier appartient à root, et que seul root peut y écrire. Un utilisateur simple ne peut lire que son contenu sans interagir. La commande **passwd** ne devrait donc pas pouvoir modifier les fichiers.

Voyez les droits de la commande **passwd** (/bin/passwd ou /usr/bin/passwd) :

```
ls -l /usr/bin/passwd  
-rwsr-xr-x 1 root shadow 78208 sep 21 23:06 /usr/bin/passwd
```

Un nouveau droit est apparu : le droit **s** pour les droits de l'utilisateur root. Ce nouvel attribut permet l'exécution de la commande avec des droits d'accès étendus. Le temps du traitement, le programme est exécuté avec les droits du propriétaire du fichier ou de son groupe d'appartenance. Dans le cas de **passwd**, il est lancé avec les droits de root le temps de son traitement.

# Droits d'accès étendus

---

Le droit `s` sur l'utilisateur est appelé le **SUID-Bit** (Set User ID Bit), et sur le groupe le **GUID-Bit** (Set Group ID Bit).

Seul le propriétaire ou l'administrateur peut positionner ce droit.

Positionner le SUID-bit ou le SGID-Bit n'a de sens que si les droits d'exécution ont préalablement été établis (attribut `x` sur le propriétaire ou le groupe).

Si ceux-ci ne sont pas présents ; le `s` est remplacé par un **S**.

La commande **chmod** permet de placer les SUID-Bit et GUID-Bit.

`chmod u+s` commande

`chmod g+s` commande

Les valeurs octales sont 4000 pour le SUID-Bit et 2000 pour le GUID-Bit.

`chmod 4755` commande

`chmod 2755` commande

## Droits d'accès étendus Real / effectif

---

Dans les données d'identification du processus vous avez vu la présence de **UID et de GID réels et effectifs**. Quand une commande est lancée avec un SUID-Bit ou un SGID-Bit positionné, les droits se trouvent modifiés. Le système conserve les UID et GID d'origine de l'utilisateur ayant lancé la commande (UID et GID réels) transmis par le père, les numéros UID et GID effectifs étant ceux du propriétaire ou du groupe d'appartenance du programme.

Ex : toto (UID=100, GID=100) lance passwd, appartenant à root (UID=1, GID=1) avec le SUID-Bit positionné.

UID réel : 100  
GID réel : 100  
UID effectif : 1  
GID effectif : 100

Si le SGID-Bit était positionné seul :

UID réel : 100  
GID réel : 100  
UID effectif : 100  
GID effectif : 1

Il est à noter que les SUID-Bit et SGID-bit ne sont pas transmis aux enfants d'un processus. Dans ce cas les enfants seront exécutés avec les droits de l'utilisateur ayant lancé la commande de base.

# Sticky bit

---

Le **sticky bit** (bit collant) permet d'affecter une protection contre l'effacement du contenu d'un répertoire. Imaginez un répertoire /tmp où tous les utilisateurs ont le droit de lire et d'écrire des fichiers.

```
$ ls -ld /tmp
$drwxrwxrwx 6 root system 16384 Aug 14 13:22 tmp
```

Dans ce répertoire tout le monde peut supprimer des fichiers y compris ceux qui ne lui appartiennent pas (droit w présent partout et pour tous). Si l'utilisateur toto crée un fichier, l'utilisateur titi peut le supprimer même s'il ne lui appartient pas.

Le sticky bit appliqué à un répertoire, ici /tmp, empêche cette manipulation. Si le fichier peut encore être visualisé et modifié, seul son propriétaire (et l'administrateur) pourra le supprimer.

```
$ chmod u+t /tmp
$ls -ld /tmp
$drwxrwxrwt 35 root root 77824 mar 21 22:30 /tmp
En octal, on utilisera la valeur 1000 (chmod 1777 /tmp).
```

Bien qu'appliqué à l'utilisateur, le sticky bit, représenté par un **t** apparaît au niveau de « others ».