

# Systèmes d'exploitation 420-W12-SF

La ligne de commandes  
Linux Bash

Gestion des usagers et droits d'accès

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

Jean-Pierre Duchesneau, Automne 2021

# Les cours

---

1. Intro au infrastructure informatique et les composantes interne du PC
2. Les composantes interne du PC et réseau
3. Système d'Exploitation
4. Virtualisation de clients et de serveurs
5. Disque dur, partition et système de fichier
6. **La ligne de commandes (Shell) et les scripts**
  1. CMD
  2. Linux
  3. Bash
  4. PowerShell
7. Git, le contrôle de version
8. WAMP

# Le shell bash (Révision)

L'interpréteur de commandes, ou interprète, permet d'exécuter des instructions que vous saisissez au clavier ou au sein d'un script et vous en retourne les résultats.

Cet interpréteur est un programme appelé shell.

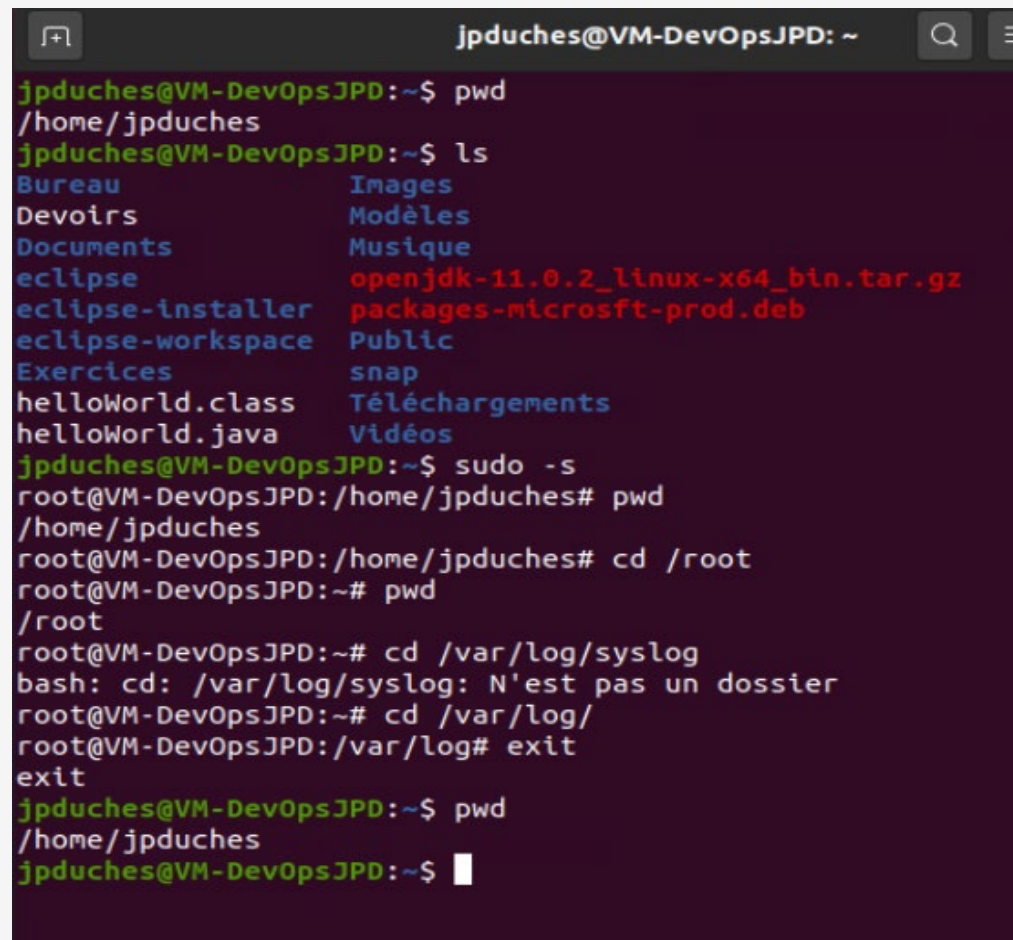
C'est à rapprocher du mot **kernel** : le kernel, signifiant noyau. Shell signifiant coquille, c'est donc ce qui « entoure » le **noyau Linux** : le moyen de l'utiliser à l'aide de commandes. C'est donc une interface fonctionnant en mode texte entre le noyau Linux et les utilisateurs (avancés), voire les applications.

Il existe plusieurs shells, chacun disposant de spécificités propres. Le Bourne Shell (sh) est le shell le plus connu et le plus courant sur les Unix.

Le shell de référence sous Linux se nomme le Bourne Again Shell (bash) et dérivé du sh.

Le shell fonctionne au sein d'un terminal. Il attend des saisies au clavier dans la console ou la fenêtre, et affiche ses résultats au même endroit.

L'invite (prompt) fournit des informations sur le terminal et votre position dans le système de fichiers.



```
jpduches@VM-DevOpsJPD: ~  
jpduches@VM-DevOpsJPD:~$ pwd  
/home/jpduches  
jpduches@VM-DevOpsJPD:~$ ls  
Bureau          Images  
Devoirs         Modèles  
Documents       Musique  
eclipse         openjdk-11.0.2_linux-x64_bin.tar.gz  
eclipse-installer packages-microsoft-prod.deb  
eclipse-workspace Public  
Exercices       snap  
helloWorld.class Téléchargements  
helloWorld.java Vidéos  
jpduches@VM-DevOpsJPD:~$ sudo -s  
root@VM-DevOpsJPD:/home/jpduches# pwd  
/home/jpduches  
root@VM-DevOpsJPD:/home/jpduches# cd /root  
root@VM-DevOpsJPD:~# pwd  
/root  
root@VM-DevOpsJPD:~# cd /var/log/syslog  
bash: cd: /var/log/syslog: N'est pas un dossier  
root@VM-DevOpsJPD:~# cd /var/log/  
root@VM-DevOpsJPD:/var/log# exit  
exit  
jpduches@VM-DevOpsJPD:~$ pwd  
/home/jpduches  
jpduches@VM-DevOpsJPD:~$
```

# Syntaxe générale des commandes (Révision)

Les commandes ont très souvent une syntaxe reprenant la même structure :

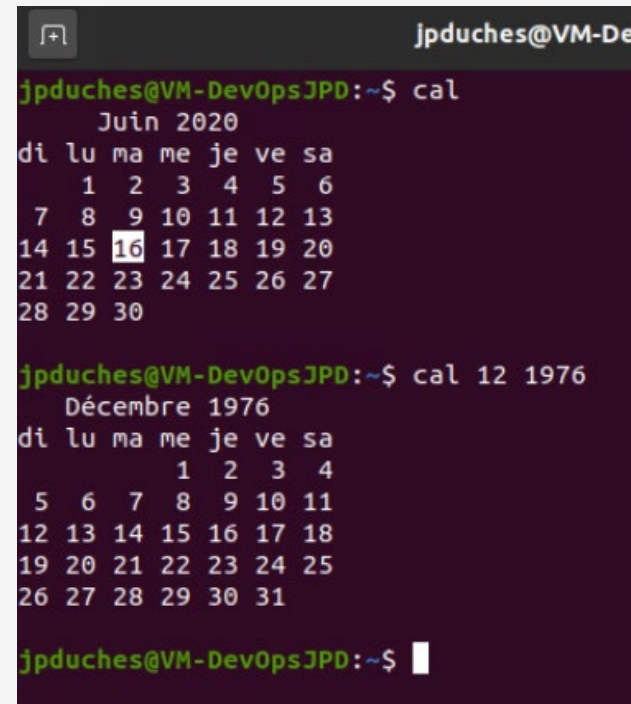
## Commande **[paramètre][arguments]**

Une commande peut avoir ni paramètres, ni arguments. Dans ce cas elle exécute l'action par défaut pour laquelle elle est programmée, ou affiche un message d'erreur si ceux-ci sont nécessaires.

Un **paramètre** est une option de la commande. Les deux mots sont ici synonymes. C'est souvent une simple lettre ou un simple chiffre précédé d'un tiret : -l, -p, -s, etc.

Les **arguments** sont les entités sur lesquelles la commande doit exécuter son action. Leur type dépend de la commande. Ce peut être un fichier, du texte, des nombres, etc.

La commande **cal** admet deux arguments optionnels (les seuls dans la norme POSIX). Si un seul est précisé, il s'agit de l'année, et l'intégralité du calendrier de cette année est affichée. Si deux arguments sont précisés, le premier est le mois, le second l'année.



```
jpduches@VM-De
jpduches@VM-DevOpsJPD:~$ cal
    Juin 2020
di lu ma me je ve sa
   1  2  3  4  5  6
  7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

jpduches@VM-DevOpsJPD:~$ cal 12 1976
    Décembre 1976
di lu ma me je ve sa
           1  2  3  4
  5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

jpduches@VM-DevOpsJPD:~$
```

# Le Shell et ses principales commande (Révision)

---

- **ls** : liste le contenu d'un répertoire
- **pwd** : afficher le répertoire courant
- **cd** : change de répertoire
- **cd ..** : répertoire parent
- **cd ~** : allez au répertoire de l'utilisateur
- **mkdir** : créer un répertoire
- **rmdir** : supprime un répertoire
- **cp** : copie de fichier
- **mv** : déplacement de fichier.
- **rm** : supprime le fichier (rm -rf /home/toto/temp)
- **passwd** : change le mot de passe de l'utilisateur
- **cat**: affiche le contenu du fichier
- **date** : date et l'heure actuelles.

# Rappel de l'historique (Révision)

La flèche du haut remonte dans l'historique. La flèche du bas navigue dans l'autre sens, jusqu'à l'invite d'origine. Si vous appuyez sur la touche [Entrée] vous lancez de nouveau la commande.

Plus vous tapez des commandes, plus l'historique s'agrandit. Le shell conserve ainsi un grand nombre d'entrées dans l'historique (le nombre de lignes conservées peut être modifié). Cet historique est conservé dans un fichier caché de votre répertoire personnel appelé `.bash_history`. Vous pouvez voir le contenu de l'historique avec la commande **history**.

```
jpduches@VM-DevOpsJPD: ~  
421 ls -al  
422 git status  
423 git log  
424 mkdir JPDPProf  
425 cd JPDPProf/  
426 touch JPDtext.txt  
427 gedit JPDtext.txt  
428 git status  
429 git add .  
430 git commit -m 'Mon premier commit sur un dépôt distant'  
431 git status  
432 git push  
433 pwd  
434 cd  
435 pwd  
436 cat .ssh/id_ed25519.pub  
437 cd Bureau/infraubr4361  
438 ls  
439 git status  
440 git remote -v  
441 git fetch  
442 ls  
443 ls -al  
444 git pull
```

```
532 type pwd  
533 type cat  
534 clear  
535 historey  
536 clear  
537 history  
jpduches@VM-DevOpsJPD:~$
```

# Rappel de l'historique (Révision)

La commande **fc** effectue presque la même chose lorsqu'on utilise le paramètre **-l**. Par défaut elle se limite aux quinze dernières commandes. Aussi vous pouvez lui passer le nombre des dernières commandes, comme

Vous pouvez rappeler une commande avec **fc** et le paramètre **-s** suivi du numéro de la commande. Elle sera alors automatiquement lancée.

D'autres raccourcis-clavier sont pratiques :

- **[Ctrl] a** : aller au début de la ligne.
- **[Ctrl] e** : aller en fin de ligne.
- **[Ctrl] l** : effacer le contenu du terminal, et afficher l'invite en haut de celui-ci.
- **[Ctrl] u** : effacer la ligne jusqu'au début.
- **[Ctrl] k** : effacer la ligne jusqu'à la fin.

```
528 clear
529 cal
530 cal 12 1976
531 type date
532 type pwd
533 type cat
534 clear
535 historey
536 clear
537 history
jpduches@VM-Dev0psJPD:~$ fc -s 530
cal 12 1976
    Décembre 1976
di lu ma me je ve sa
                1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

jpduches@VM-Dev0psJPD:~$
```



# Les commandes man et help

**man** : est une commande disponible sur les systèmes d'exploitation de type Unix. Elle permet de visionner les contenus d'une documentation formatée pour être exploitable par man;

**help** : Afficher une liste des commandes shell. Utilisez help directement avec une commande shell pour afficher une courte description de la commande correspondante

```
jpduches@Bilbo: ~  
jpduches@Bilbo:~$ ls --help  
Usage: ls [OPTION]... [FILE]...  
List information about the FILES (the current directory by default).  
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.  
  
Mandatory arguments to long options are mandatory for short options too.  
-a, --all do not ignore entries starting with .  
-A, --almost-all do not list implied . and ..  
--author with -l, print the author of each file  
-b, --escape print C-style escapes for nongraphic characters  
--block-size=SIZE with -l, scale sizes by SIZE when printing them;  
e.g., '--block-size=M'; see SIZE format below  
-B, --ignore-backups do not list implied entries ending with ~  
-c with -lt: sort by, and show, ctime (time of last  
modification of file status information);  
with -l: show ctime and sort by name;  
otherwise: sort by ctime, newest first  
-C list entries by columns  
--color[=WHEN] colorize the output; WHEN can be 'always' (default  
if omitted), 'auto', or 'never'; more info below  
-d, --directory list directories themselves, not their contents  
-D, --dired generate output designed for Emacs' dired mode  
-f do not sort, enable -aU, disable -ls --color  
-F, --classify append indicator (one of */=>@|) to entries  
--file-type likewise, except do not append '*'  
--format=WORD across -x, commas -m, horizontal -x, long -l,  
single-column -1, verbose -l, vertical -C  
--full-time like -l --time-style=full-iso  
-g like -l, but do not list owner  
--group-directories-first
```

```
jpduches@Bilbo: ~  
LS(1) User Commands LS(1)  
NAME  
ls - list directory contents  
SYNOPSIS  
ls [OPTION]... [FILE]...  
DESCRIPTION  
List information about the FILES (the current directory by default). Sort entries alphabetically if none of  
-cftuvSUX nor --sort is specified.  
  
Mandatory arguments to long options are mandatory for short options too.  
  
-a, --all do not ignore entries starting with .  
  
-A, --almost-all do not list implied . and ..  
  
--author with -l, print the author of each file  
  
-b, --escape print C-style escapes for nongraphic characters  
  
--block-size=SIZE with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below  
  
Manual page ls(1) line 1 (press h for help or q to quit)
```



# Commande de base

Ls Lister fichier/sous-répertoire d'un répertoire

- Option `-l` (long) lister détails : sécurité, propriétaire, groupe
- Option `-a` (caché) lister les fichier cachés

**Attention** : Les commandes sont sensible à la casse.  
Donc les commandes `LS`, `Ls` et `IS` n'existent pas.

```
jpduches@Bilbo: ~  
jpduches@Bilbo:~$ ls -l  
total 196  
drwxr-xr-x 4 jpduches jpduches 4096 Aug  5 10:22 docker  
drwxr-xr-x 4 jpduches jpduches 4096 Aug 12 15:43 exercice12  
drwxr-xr-x 2 jpduches jpduches 4096 Aug 17 12:57 html  
-rw-r--r-- 1 jpduches jpduches 89416 Aug 18 10:04 k8s.json  
-rw-r--r-- 1 jpduches jpduches 89416 Aug 18 10:04 k8s.yaml  
drwxr-xr-x 4 jpduches jpduches 4096 Jun  4 17:14 scripts  
drwxr-xr-x 9 jpduches jpduches 4096 Aug 17 13:18 webapp  
jpduches@Bilbo:~$ ls -al  
total 296  
drwxr-xr-x 17 jpduches jpduches 4096 Aug 18 10:04 .  
drwxr-xr-x  3 root      root      4096 Apr 22  2021 ..  
drwxr-xr-x  3 jpduches jpduches 4096 Aug 17 13:18 .aspnet  
lrwxrwxrwx  1 jpduches jpduches   26 Aug  3 11:04 .aws -> /mnt/c/Users/jpduches/.aws  
lrwxrwxrwx  1 jpduches jpduches   28 Aug  3 11:04 .azure -> /mnt/c/Users/jpduches/.azure  
-rw----- 1 jpduches jpduches 21348 Nov  4 10:41 .bash_history  
-rw-r--r-- 1 jpduches jpduches  220 Apr 22  2021 .bash_logout  
-rw-r--r-- 1 jpduches jpduches 3771 Apr 22  2021 .bashrc  
drwxr-xr-x  3 jpduches jpduches 4096 May 17 16:33 .cache  
drwx----- 4 jpduches jpduches 4096 Jun  9 15:06 .config  
drwxr-xr-x  4 jpduches jpduches 4096 Aug  4 12:01 .docker  
drwxr-xr-x  6 jpduches jpduches 4096 Aug 17 13:18 .dotnet  
-rw-r--r--  1 jpduches jpduches   69 May 18 16:15 .gitconfig  
drwxr-xr-x  3 jpduches jpduches 4096 Aug 18 09:48 .kube  
drwxr-xr-x  2 jpduches jpduches 4096 Apr 22  2021 .landscape  
drwxr-xr-x  3 jpduches jpduches 4096 May 17 16:33 .local  
-rw-r--r--  1 jpduches jpduches    0 Nov 11 15:31 .motd_shown  
drwxr-xr-x  3 jpduches jpduches 4096 Aug  5 11:01 .nuget  
-rw-r--r--  1 jpduches jpduches  807 Apr 22  2021 .profile
```

# 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:~$
```

# 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 l'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`.

# 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
```



# 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 /~
```

# 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).

# 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 sorte 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/
```

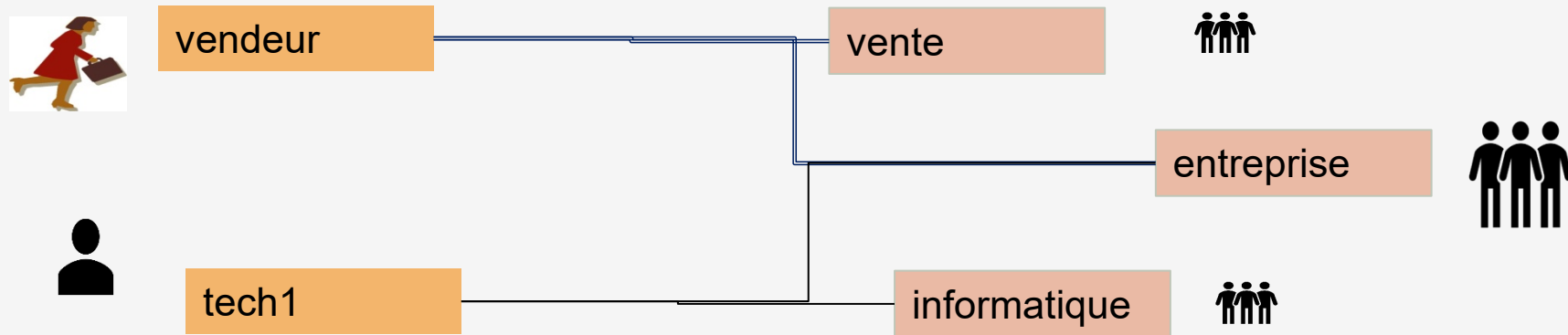
Le cas échéant le lien se comportera à l'identique du fichier pointé avec les mêmes permissions et les mêmes propriétés :

- 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 **ubuntusudo** 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 ? [0/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>



vendeur

tech1

vente



entreprise



informatique



```
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 d'accès.
- 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

Loupe

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

```
-   rwx   rw-   r--   2   tech1   informatique   . . .   doc1.txt
  {      {      {      {      {
  {      {      {      {      {
```

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-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$ chmod g+x fichier1.txt
chmod: modification des droits de 'fichier1.txt': Opération non permise
jpduches@VM-SEJPD:~/Exercices$ chmod g+x 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-rwxr-- 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
```

# Travail à faire

---

## Exercices

- Dernière période pour l'installation de Linux
- Exercice 9 Visite guidée de Ubuntu (**attention au nouveau compte Prof**)
- Exercice 10 Commandes de bases du Shell Linux
- Exercice 11 Gestion des usagers Linux