



HAUTE ÉCOLE DE  
NAMUR-LIÈGE-LUXEMBOURG

**Catégorie technique orientation TI-IR-RT**

**Année académique : 2023-2024**

---

# Ligne de commande sous Linux Debian III

---

### Introduction

L'interface en ligne de commande (CLI - Command Line Interface) reste un outil fondamental pour les professionnels de l'informatique. Elle offre une puissance et une flexibilité inégalées, permettant des interactions précises avec le système d'exploitation (OS) et les logiciels. Même sous Windows, des outils comme PowerShell et le Windows Subsystem for Linux (WSL) ont réaffirmé l'importance de la CLI.

### Spécificités des Systèmes

Tout système d'exploitation a ses particularités. Dans le cadre de Linux et Windows, ces différences sont notables, notamment dans la gestion des fichiers et des permissions. Une compréhension approfondie de ces spécificités permet une utilisation efficace et sécurisée du système. Une des différences notables quand on compare les deux systèmes, c'est la notation des répertoires et périphériques.

#### Sous Windows :

```
C:\Users\user>
```

Nous avons 3 choses remarquables :

- on utilise le « \ » pour séparer les dossiers et sous-dossiers
- on utilise une lettre (ici « C : » pour le disque dur) pour indiquer le périphérique actif
- on a pas de distinction visuelle dans le prompt pour différencier user/administrateur

#### Sous Linux :

```
user@deb-11-xfce-base:~$
```

Nous avons 5 choses à signaler :

- on utilise le « / » pour séparer les dossiers et sous-dossiers

```
user@deb-11-xfce-base:/etc/apt/sources.list.d$
```

- Le périphérique actif dans la ligne de commande n'est pas différencié du reste
- il y a une colorisation du prompt du user
- il y a une distinction entre user/administrateur

```
root@deb-11-xfce-base:~#
```

- Il y a un raccourci (« ~ ») pour le répertoire « home » du user (idem pour le root). C'est le même raccourci pour tous, mais ils pointent tous sur un dossier différent.

## Organisation des Fichiers et Répertoires

L'organisation des fichiers en répertoires est cruciale. Elle permet non seulement une gestion ordonnée des données, mais aussi une hiérarchisation des droits. Cette hiérarchisation est fondamentale pour contrôler l'accès aux fichiers et sous-dossiers, garantissant ainsi la sécurité des données.

## Distributions Debian : Une Solution Distinguée

Debian, une des distributions Linux les plus populaires, adopte une structure de fichiers et de répertoires bien organisée, conformément à la norme Filesystem Hierarchy Standard (FHS). Cette structuration permet une gestion simplifiée des packages et des dépendances.

## Fichiers et Dossiers Système

Les fichiers et dossiers système sont essentiels au fonctionnement de l'OS. Ils contiennent des configurations, des scripts de démarrage, et des binaires cruciaux. Leur modification ou suppression incorrecte peut entraîner des dysfonctionnements du système.

### Buts et Fonctions

- **Binaires:** Exécutables nécessaires au fonctionnement du système.
- **Configuration:** Fichiers de configuration pour personnaliser le système.
- **Logs:** Traces des événements du système pour le débogage.

## Architecture de Dossiers sous Debian

Debian suit la FHS avec des dossiers clés comme `/bin`, `/etc`, `/usr`, et `/var`. Par exemple :

- `/bin` : Binaires essentiels.
- `/etc` : Fichiers de configuration.
- `/home` : Répertoires personnels des utilisateurs.
- `/lib` : Bibliothèques essentielles.
- `/opt` : Logiciels additionnels.
- `/tmp` : Fichiers temporaires.
- `/usr` : Binaires, bibliothèques, documentation, etc., non essentiels.
- `/var` : Fichiers variables comme les logs.

Une représentation "graphique" des dossiers donnera ceci :

```
jeadev@jeadev-HP-ProBook-450:~$ tree / -L 1
/
├── bin -> usr/bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib32 -> usr/lib32
├── lib64 -> usr/lib64
├── libx32 -> usr/libx32
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

## Fonctionnalités Spécifiques des Dossiers sous Debian

Chaque dossier sous Debian a un but précis, facilitant la localisation des fichiers et la réparation des problèmes.

### La Notion de Lien

Un lien est une référence à un fichier ou un répertoire. Sous Linux, il existe deux types de liens : les liens symboliques (ou soft links) et les liens durs (ou hard links).

```
└── sbin -> usr/sbin
```

## Chemin Absolu vs Chemin Relatif

Cette notion est valable sous Windows, comme sous Linux. Une différence existe cependant entre les deux systèmes au niveau des caractères utilisés.

- **Chemin Absolu** : Indique la localisation précise d'un fichier ou répertoire depuis la racine (/).

```
/etc/ghostscript  
/etc/hosts  
/etc/hosts.allow  
/etc/hosts.deny
```

- **Chemin Relatif** : Indique la localisation d'un fichier ou répertoire par rapport au répertoire courant.

```
jeadev@jeadev-HP-ProBook-450:/etc$ du emacs/  
16      emacs/site-start.d  
20      emacs/
```

## Répertoires . et ..

Cette notion aussi, est commune aux systèmes Windows et Linux.

- **.** : Référence au répertoire courant.
- **..** : Référence au répertoire parent.

Ces notions fondamentales de la CLI et de l'organisation des fichiers sous Debian fournissent la base nécessaire pour explorer et interagir efficacement avec le système d'exploitation Linux. Le laboratoire suivant vous permettra de mettre en pratique ces concepts essentiels.

```
4,0K oct 10 23:35 .  
4,0K avr 18 2022 ..  
541M nov 2 2022 all.tar.gz  
4,0K jun 10 01:46 .AMC.d
```

---

## Gestion des fichiers.

---

Nous en arrivons à l'un des objectifs de ce laboratoire : la gestion des fichiers.

Il est très important de pouvoir gérer les fichiers et dossiers en ligne de commande, car cela permet d'inscrire des commandes dans un fichier de script qui permet d'automatiser certaines opérations. De plus, les commandes dans un terminal permettent une multitude de variations que l'interface graphique ne permet pas.

### Gestion des répertoires

Les fichiers seront mis dans un répertoire. Le premier répertoire du système, c'est la racine (/). C'est une très mauvaise pratique que d'utiliser la racine pour stocker des fichiers (ou des dossiers).

La bonne pratique, sous Linux comme sous Windows, consiste à organiser des répertoires de façon réfléchie.

Une des premières choses à faire pour le faire quelque part, c'est d'y aller ! Nous allons donc naviguer dans les dossiers.

Pour changer de répertoire, il y a la commande « cd » (pour change directory).

<b><u>Utilisation</u></b>	<b><u>action</u></b>
cd	déplacement vers le répertoire « home » de l'utilisateur courant
cd ~	idem
cd /	déplacement vers la racine du système de fichiers
cd -	déplacement vers le dossier précédent (pas le parent)
cd xxxx	déplacement vers le dossier xxxx qui est un sous-répertoire du répertoire courant (erreur s'il n'existe pas). « xxxx » est un chemin relatif.
cd /xxxx	déplacement vers le dossier xxxx qui est à la racine du système de fichier (erreur s'il n'existe pas). « /xxxx » est un chemin absolu.
cd ../xxxx	déplacement vers le dossier xxxx qui est au même niveau que le répertoire courant

Une fois que l'on sait où on est, on a souvent quelque-chose à y faire. Nous allons maintenant créer des dossiers, les supprimer et les déplacer.

<b><u>Utilisation</u></b>	<b><u>action</u></b>
mkdir	erreur (il faut donner le nom du dossier à créer)

`mkdir rep1` création du dossier « rep1 » dans le répertoire courant

`mkdir rep1 rep2` création des dossier « rep1 » et « rep2 » dans le répertoire courant

`mkdir rep3/rep4` erreur si le répertoire « rep3 » n'existe pas

`mkdir -p rep3/rep4` là, ça passe

`mkdir /rep1` création du dossier « rep1 » à la racine du système de fichier  
... si vous êtes en « root » : **question : pourquoi ?**

Où suis-je ? C'est une question que l'on se pose parfois quand on navigue dans la ligne de commande. Outre le fait que, souvent, le répertoire courant est affiché dans le prompt de la ligne de commande, on a parfois besoin de le savoir dans un script (fichier qui rassemble une liste de commandes).

`pwd`

Une action souvent utilisée aussi, c'est lister le contenu d'un fichier.

## **Utilisation**

`ls`

```
mint@mint:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
```

`ls -l`

Cette option permet d'afficher plus de détails sur les fichiers. On voit ici, le propriétaire du fichier, le groupe auquel est attaché le fichier, sa date de création et les droits appliqués à ces fichiers.

```
mint@mint:~$ ls -l
total 0
drwxr-xr-x 2 mint mint 60 Oct 12 07:20 Desktop
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Documents
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Downloads
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Music
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Pictures
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Public
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Templates
drwxr-xr-x 2 mint mint 40 Oct 12 07:20 Videos
```

`ls -la`

```

mint@mint:~$ ls -la
total 32
drwxr-x--- 14 mint mint 420 Oct 12 07:20 .
drwxr-xr-x  1 root root  60 Oct 12 07:20 ..
-rw-----  1 mint mint  49 Oct 12 07:20 .Xauthority
-rw-r--r--  1 mint mint 220 Oct 12 07:20 .bash_logout
-rw-r--r--  1 mint mint 3771 Oct 12 07:20 .bashrc
drwx-----  9 mint mint 180 Oct 12 07:20 .cache
drwxr-xr-x 12 mint mint 320 Oct 12 07:20 .config
-rw-r--r--  1 mint mint  22 Oct 12 07:20 .gtkrc-2.0
-rw-r--r--  1 mint mint 516 Oct 12 07:20 .gtkrc-xfce
drwxrwxr-x  3 mint mint  60 Oct 12 07:20 .linuxmint
drwx-----  3 mint mint  60 Oct 12 07:20 .local
-rw-r--r--  1 mint mint 807 Oct 12 07:20 .profile
-rw-----  1 mint mint 4766 Oct 12 07:21 .xsession-errors
drwxr-xr-x  2 mint mint  60 Oct 12 07:20 Desktop
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Documents
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Downloads
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Music
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Pictures
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Public
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Templates
drwxr-xr-x  2 mint mint  40 Oct 12 07:20 Videos

```

Cette option permet de lister les fichiers cachés. Pour cacher un fichier ou un répertoire, sous Linux, on fait commencer son nom avec un ".". Ces fichiers n'apparaissent pas dans les commandes sans l'option qui va bien. De plus, le "." fait partie du nom du fichier. Il faut donc le mettre quand on veut réaliser une commande sur ce fichier.

Exemple : `cat .bashrc`

D'autres options très intéressantes sont encore disponibles avec la commande `ls`. Utilisez l'aide de la commande pour vous familiariser avec les principales.

## Gestion des fichiers

Les fichiers sont d'autres objets accessibles dans un système de fichier. Nous venons de voir les répertoires, voici les fichiers. Sur ces *objets*, il est possible d'exécuter des commandes. Certaines peuvent s'exécuter sur les deux. (copy, move ...)

Copier un fichier, on peut le faire avec des options diverses.



## Utilisation      action

### Versions simples

cp fichier1 fichier2      **copie** le fichier1 dans un nouveau fichier de nom fichier2

cp fichier1 rep1 **copie** le fichier fichier1 dans le répertoire rep1 (s'il existe sinon ...)

cp fichier1 rep1/. **copie** le fichier fichier1 dans le répertoire rep1 (s'il existe sinon ...)

cp fichier1 /home/<user>/rep1

**copie** le fichier fichier1 dans le répertoire rep1 (s'il existe sinon ...)

cp fichier1 /home/<user>/rep1/.

**copie** le fichier fichier1 dans le répertoire rep1 (s'il existe sinon ...)

Nous avons ici des solutions avec chemin **absolus** et **relatifs**.

### Versions multiples

cp fichier1 fichier2 fichier3 rep1

**copie** les fichiers fichier1, fichier2 et fichier3 dans le répertoire rep1

cp -t rep1 fichier1 fichier2 fichier3

idem

mv fichier1 fichier2      **déplace** le fichier fichier1 vers le fichier2 en écrasant un éventuel fichier2 existant, donc, le renomme

mv fichier1 rep1 **déplace** le fichier fichier1 vers le répertoire rep1

rm fichier1 fichier2      **efface** les fichiers fichier1 et fichier2

rm -r rep1      **efface** le dossier rep1

Beaucoup d'autres options sont possibles. Explorez-les pour vous faire une idée.

## Les jokers

Pour les commandes que nous venons de voir, ainsi que pour les autres, nous pouvons utiliser des

Jokers qui permettent de traiter plusieurs fichiers sans forcément les nommer tous.

Les premiers sont « \* » et « ? ».

« ? » remplace un caractère

« \* » remplace 1 ou plusieurs caractères

« {} » énumère des caractères

Exemples :

Si on a la situation suivante :

```
user@mint:~/Documents/tests$ ls
fic00  fic10  fic20  fic30  fic40  fic50  fic60  fic70  fic80  fic90  lev2
fic01  fic11  fic21  fic31  fic41  fic51  fic61  fic71  fic81  fic91
fic02  fic12  fic22  fic32  fic42  fic52  fic62  fic72  fic82  fic92
fic03  fic13  fic23  fic33  fic43  fic53  fic63  fic73  fic83  fic93
fic04  fic14  fic24  fic34  fic44  fic54  fic64  fic74  fic84  fic94
fic05  fic15  fic25  fic35  fic45  fic55  fic65  fic75  fic85  fic95
fic06  fic16  fic26  fic36  fic46  fic56  fic66  fic76  fic86  fic96
fic07  fic17  fic27  fic37  fic47  fic57  fic67  fic77  fic87  fic97
fic08  fic18  fic28  fic38  fic48  fic58  fic68  fic78  fic88  fic98
fic09  fic19  fic29  fic39  fic49  fic59  fic69  fic79  fic89  fic99
user@mint:~/Documents/tests$
```

Si on veut déplacer tous les fichiers contenant un « 4 » comme quatrième caractère :

```
user@mint:~/Documents/tests$ mv *4? lev2/
```

Ce qui donnera :

```
user@mint:~/Documents/tests$ ls lev2/
fic40  fic41  fic42  fic43  fic44  fic45  fic46  fic47  fic48  fic49
```

Si nous avions fait :

```
user@mint:~/Documents/tests$ mv *4* lev2/
```

Ceci aurait donné :

```
user@mint:~/Documents/tests$ ls lev2/
fic04  fic24  fic40  fic42  fic44  fic46  fic48  fic54  fic74  fic94
fic14  fic34  fic41  fic43  fic45  fic47  fic49  fic64  fic84
```

« \* » remplace 0,1,ou plusieurs caractères. Donc, avec « \*4\* », le « 4 » est pris n'importe où dans le nom du fichier.

Nous avons maintenant la situation suivante :

```
user@mint:~/Documents/tests$ ls -R
.:
fic00 fic09 fic18 fic27 fic36 fic55 fic63 fic72 fic81 fic90 fic99
fic01 fic10 fic19 fic28 fic37 fic56 fic65 fic73 fic82 fic91 lev2
fic02 fic11 fic20 fic29 fic38 fic57 fic66 fic75 fic83 fic92
fic03 fic12 fic21 fic30 fic39 fic58 fic67 fic76 fic85 fic93
fic05 fic13 fic22 fic31 fic50 fic59 fic68 fic77 fic86 fic95
fic06 fic15 fic23 fic32 fic51 fic60 fic69 fic78 fic87 fic96
fic07 fic16 fic25 fic33 fic52 fic61 fic70 fic79 fic88 fic97
fic08 fic17 fic26 fic35 fic53 fic62 fic71 fic80 fic89 fic98

./lev2:
fic04 fic24 fic40 fic42 fic44 fic46 fic48 fic54 fic74 fic94
fic14 fic34 fic41 fic43 fic45 fic47 fic49 fic64 fic84
user@mint:~/Documents/tests$
```

Pour remplacer les fichiers du dossier « lev2 » au dossier « test », on peut utiliser la commande :

```
user@mint:~/Documents/tests$ mv lev2/* .
user@mint:~/Documents/tests$ ls -R
.:
fic00 fic10 fic20 fic30 fic40 fic50 fic60 fic70 fic80 fic90 lev2
fic01 fic11 fic21 fic31 fic41 fic51 fic61 fic71 fic81 fic91
fic02 fic12 fic22 fic32 fic42 fic52 fic62 fic72 fic82 fic92
fic03 fic13 fic23 fic33 fic43 fic53 fic63 fic73 fic83 fic93
fic04 fic14 fic24 fic34 fic44 fic54 fic64 fic74 fic84 fic94
fic05 fic15 fic25 fic35 fic45 fic55 fic65 fic75 fic85 fic95
fic06 fic16 fic26 fic36 fic46 fic56 fic66 fic76 fic86 fic96
fic07 fic17 fic27 fic37 fic47 fic57 fic67 fic77 fic87 fic97
fic08 fic18 fic28 fic38 fic48 fic58 fic68 fic78 fic88 fic98
fic09 fic19 fic29 fic39 fic49 fic59 fic69 fic79 fic89 fic99

./lev2:
user@mint:~/Documents/tests$
```

« ls -R » est là pour montrer le résultat.

Si, en une seule commande, je veux créer les fichiers « fica », « ficb », « ficc », « ficd », « fice », « ficf », « fic00 », « fic01 », « fic02 », « fic03 », « fic04 », « fic05 », « fic06 », « fic07 », « fic08 », « fic09 », « fic10 », « fic11 », « fic12 », « fic13 », « fic14 », « fic15 », je peux le faire avec la commande :

```
user@mint:~/Documents/tests$ touch fic{{a..f},{00..15}}
user@mint:~/Documents/tests$ ls
fic00 fic02 fic04 fic06 fic08 fic10 fic12 fic14 fica ficc fice lev2
fic01 fic03 fic05 fic07 fic09 fic11 fic13 fic15 ficb ficd ficf
user@mint:~/Documents/tests$
```

Notez qu'il existe encore beaucoup d'options dans ce genre (caractères numérique, ponctuation, ...).

## Utilisation du résultat d'une commande

Certaines commandes donnent des informations que l'on peut réutiliser dans d'autres commandes. On peut le faire en passant par des variables, ou en intégrant une instruction dans une autre.

Une commande souvent utilisée est la commande « echo » qui permet d'afficher une information.

```
user@mint:~/Documents/tests$ echo -e "\"liste\":\nNom\tPrénom\nMarcel\tDurant\nRoger\tDuflant"
"liste":
Nom      Prénom
Marcel   Durant
Roger    Duflant
user@mint:~/Documents/tests$
```

... et bienvenue, au passage, dans le monde merveilleux des expressions régulières.

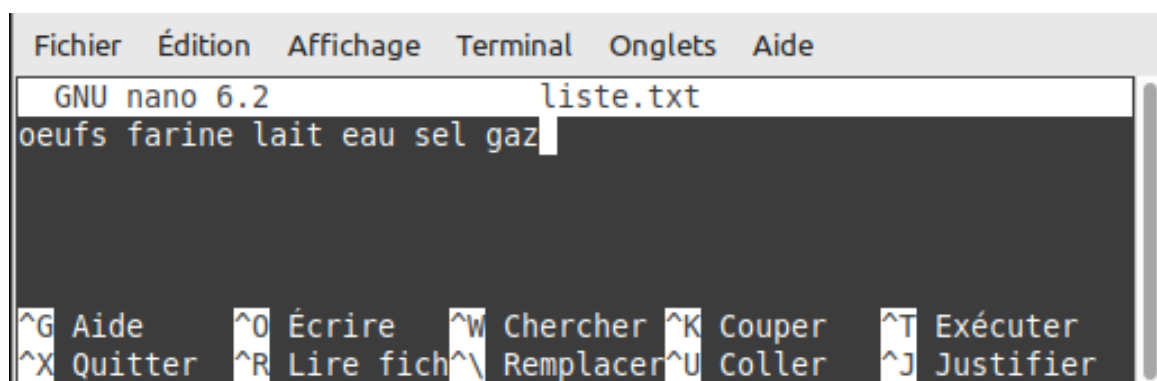
Le caractère « \ » est un caractère dit *d'échappement*, c'est-à-dire, que l'on va donner une fonction au caractère.

Allez voir sur les ressources internet tous ceux que vous pouvez trouver.

Dans cette commande, vous pouvez utiliser le résultat d'une autre commande. Exemple :

```
user@mint:~/Documents/tests$ echo "Il est $(date +%R) ... et tout va bien"
Il est 11:04 ... et tout va bien
user@mint:~/Documents/tests$
```

Si, dans nano, on édite le fichier liste.txt comme suit :



```
GNU nano 6.2 liste.txt
oeufs farine lait eau sel gaz
```

... et qu'on lance la commande :

```
user@mint:~/Documents/tests$ touch $(cat liste.txt)
user@mint:~/Documents/tests$ ls
eau farine gaz lait lev2 liste.txt oeufs sel
user@mint:~/Documents/tests$
```

... on obtient le résultat donné par la commande ls.

## Exercices :

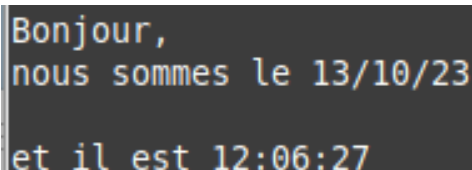
Le travail sur lequel le laboratoire sera évalué est un fichier qui rassemble les commandes que l'on vous demande d'exécuter. Chaque commande sera sur une ligne séparée. Il n'y a pas besoin de mettre de texte en commentaire dans le fichier.

Ce fichier s'appelle « **monPremierScript.sh** ».

Il vous est demandé, en utilisant l'éditeur « nano » du terminal Linux, de recopier dans l'ordre les commandes qui vont donner les actions suivantes :

(N.B. toutes les actions qui vous sont demandées sont à exécuter en une seule commande)

1. (/1) Créer, en une commande un ensemble de dossiers de « fic00 » à « fic99 » dans lesquels il y a les dossier « subfolder00 » à « subfolder99 »
2. (/1) Effacer tous les dossiers « subfolder55 »



```
Bonjour,  
nous sommes le 13/10/23  
et il est 12:06:27
```

3. (/1) Dans le dossier « fic55/subfolder56 », créer les fichier « fichiera » à « fichierz »
4. (/1) Déplacer le dossier « fic11 » dans le dossier « fic12 »
5. (/3) Afficher un message affichant la date courant et l'heure courante au format suivant :

(au format local)

6. (/3) Dans chaque dossier « ficxx » contenant un « 0 », créez un fichier « nv\_date » où « date » est la date au format <année sur 2 digits>\_<mois sur 2 digits>\_<jour sur 2 digits>