Systèmes d'exploitation

Partie 2 : Lignes de commandes et système de fichiers

Dernière mise à jour le : 09/04/2024

Introduction

Les premiers systèmes d'exploitation étaient dépourvus d'interface graphique : concrètement, il n'y avait pas de fenêtres pilotables à la souris et toutes les interactions avec l'OS devaient se faire en **lignes de commandes**, c'est-à-dire des suites de caractères formant des instructions.

De nos jours, les interfaces de graphiques modernes permettent d'effectuer la plupart des opérations souhaitées. Il est cependant nécessaire de connaître quelques-unes des lignes de commande de base car :

- c'est important pour un informaticien
- cela permet bien souvent d'aller plus vite dans certaines tâches
- cela permet d'exécuter des tâches que l'on ne pourrait pas (ou difficilement) effectuer avec l'interface graphique
- cela permet d'exécuter des instructions sur un ordinateur distant (pour lequel on n'a pas accès à son écran, qui n'existe pas toujours d'ailleurs), par exemple sur un serveur

Pour écrire et exécuter des lignes de commande on utilise pour cela une application appelée **console *ou* terminal** ou **invite de commandes** (pour simplifier on admet que ces termes désignent la même chose).

Nous allons illustrer tout cela sous Linux (mais il existe des commandes équivalentes sous Windows).

Sous GNU/Linux, il suffit d'ouvrir un Terminal pour écrire des lignes de commandes.

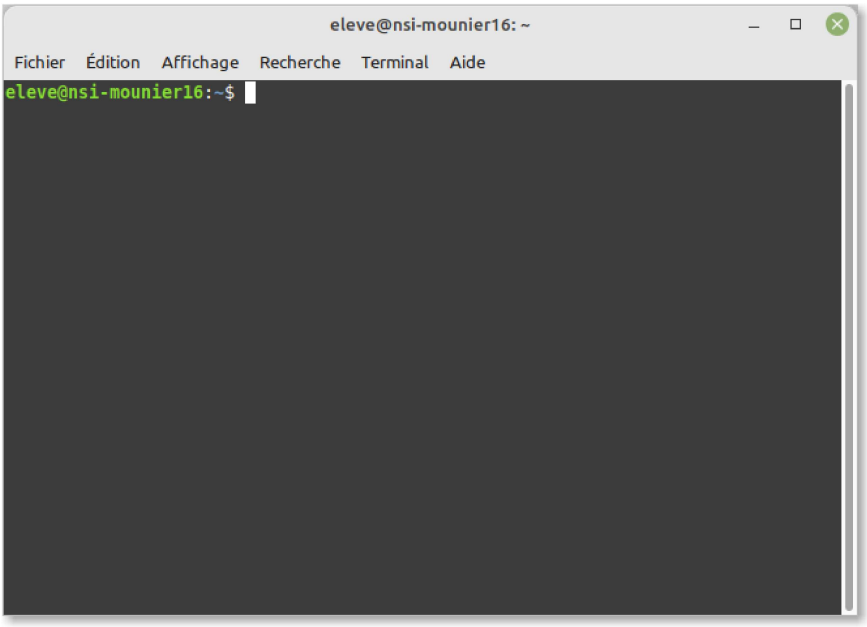


Fig. 1 - Un terminal sous Linux..

L'invite de commande se compose du nom d'utilisateur, de `@` et du nom de l'ordinateur (et aussi éventuellement de son groupe), suivi de `:`, du répertoire courant (`~` pour le répertoire utilisateur) et termine par `$`. Dans l'exemple ci-dessus,

```
eleve@nsi-mounier16:~$
```

indique qu'il s'agit de la console de l'utilisateur `eleve` qui utilise l'ordinateur appelé `nsi-mounier16` et qu'il est situé dans le répertoire utilisateur `~` (nous y reviendrons !).

Avant d'en dire plus sur les lignes de commandes il est nécessaire de présenter le système de fichiers de Linux.

Système de fichiers

Arborescence

Le système de fichiers de Linux (valable aussi par macOS) est représenté sous forme d'une **arborescence** que l'on peut représenter par ce qu'on appelle un arbre. En voici une version incomplète :

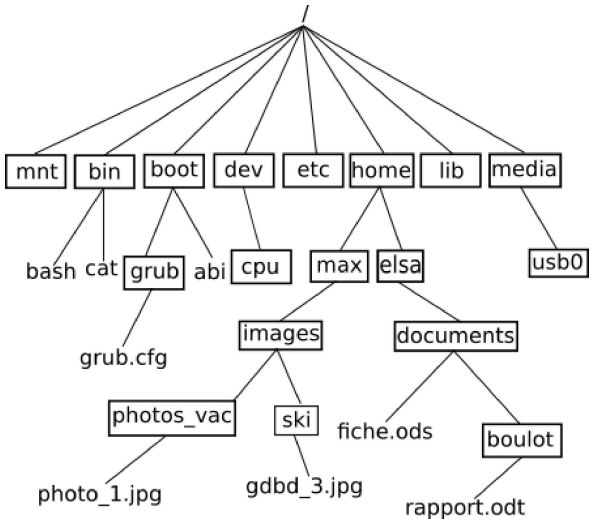


Fig. 2 - Une arborescence de fichiers.

Sur cette image, on trouve notamment :

- le **répertoire racine** `/` tout en haut , qui est le point d'entrée du système de fichiers, puis des répertoires (encadrés) contenant eux-mêmes des fichiers (non encadrés) et/ou des répertoires, ainsi de suite.
- le **répertoire des utilisateurs** `/home` qui contient les dossiers et fichiers personnels des utilisateurs : un il y a **un sous-répertoire** par utilisateur. Par exemple le répertoire personnel de l'utilisateur `eleve` est `/home/eleve`, aussi noté `~`.

Chemin absolu et chemin relatif

Pour indiquer la position d'un répertoire ou d'un fichier dans l'arborescence on utilise soit son **chemin absolu** soit son **chemin relatif**.

Chemin absolu

Le **chemin absolu** doit indiquer le chemin à partir de la racine `/`. Par exemple, le chemin absolu du fichier `fiche.ods` est :

```
/home/elsa/documents/fiche.ods
```

Chemin relatif

Le **chemin relatif** est le chemin d'un élément à partir d'un répertoire quelconque de l'arborescence (autre que la racine). Par exemple, le fichier `photo_1.jpg` a pour chemin relatif **depuis le répertoire** `max` :

```
images/photo_vac/photo_1.jpg
```

Vous noterez l'absence de `/` en tête de chemin, sinon on repartirait de la racine.

Comment faire s'il faut remonter dans l'aborescence ?

Si on veut exprimer le chemin relatif du fichier `photo_1.png` depuis le répertoire `ski`, il est nécessaire de remonter d'abord vers le répertoire `images`. Pour remonter d'un cran dans l'aborescence on utilise deux points : `..`. Ainsi, le chemin relatif évoqué est :

```
../photos_vac/photo_1.jpg
```

> Il est possible de remonter de plusieurs crans : `../..` depuis le répertoire `ski` permet de remonter vers le répertoire `max`.

Exercice 1

Question 1 : En vous basant sur l'arborescence précédente, déterminez le chemin **absolu** permettant d'accéder aux fichiers :

- `cat`
- `rapport.odt`

Question 2 : Toujours en vous basant sur cette arborescence, déterminez le chemin relatif permettant d'accéder au fichier :

- `rapport.odt` depuis le répertoire `elsa`
- `fiche.ods` depuis le répertoire `boulot`

Les commandes pour manipuler les fichiers et les répertoires

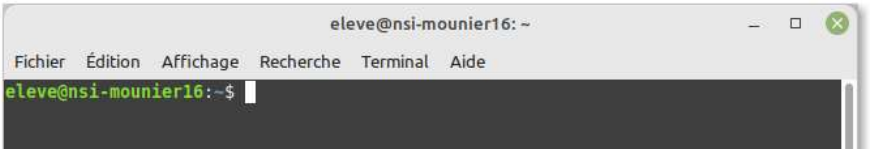
Nous allons pouvoir écrire nos premières lignes de commande pour manipuler fichiers et répertoires sous Linux.

> Pour travailler avec un terminal dans un environnement Linux chez vous, vous pouvez utiliser un des émulateurs en ligne suivants :
>
> - jslinux de Fabrice Bellard
> - cahier-nsi.fr/jslinux/ de l'éditeur Bordas
>
> Notez que pour le premier vous êtes connecté en tant que l'utilisateur `root` (= administrateur) donc votre répertoire personnel est `/root` et que l'invite de commande est légèrement différente (`#` au lieu de `$` par exemple).
>
> Pour le second, le login est `Angie` et le mot de passe est `NSI` (il ne s'affiche pas à l'écran quand on l'écrit) ; il y a déjà quelques répertoires et fichiers dans le répertoire personnel mais il suffit de ne pas en tenir compte.

Le répertoire courant

Lorsque l'on ouvre un Terminal, on se trouve à un endroit de l'aborescence, cet endroit s'appelle le **répertoire courant** et est indiqué par son chemin juste après le `:` (et avant le symbole `$`).

Par défaut, lorsque l'on ouvre un Terminal, on se trouve dans le **répertoire personnel de l'utilisateur**, noté `~`, qui est un raccourci pour le chemin `/home/eleve` (si `eleve` est l'utilisateur courant).



Si on se trouve dans le dossier `Documents` de l'utilisateur courant, alors le répertoire courant est alors `~/Documents` qui est équivalent à `/home/eleve/Documents` :

La commande `pwd` (pour **path working directory**) permet de connaitre le *chemin du répertoire courant* :

Pour exécuter une commande il suffit d'appuyer sur la touche `Entrée`.

Se déplacer dans l'arborescence : commande `cd`

La commande `cd`, pour **change directory** (soit *changer de répertoire*), permet de se déplacer dans l'arborescence, donc de changer le répertoire courant. Il suffit de faire suivre cette commande du chemin, absolu ou relatif, où on veut se rendre.

Exemples :

1. Si on se trouve dans le répertoire `elsa` et que l'on veut se rendre dans le répertoire `documents`, il faut exécuter la commande :

```
cd documents
```

ou

```
cd /home/elsa/documents
```

2. Si le répertoire courant est `photos_vac` et que l'on veut se rendre dans le répertoire `ski`, il faut exécuter la commande :

```
cd ../ski
```

ou

```
cd /home/max/images/ski
```

3. Si le répertoire courant est le répertoire `boulot` et que l'on veut se rendre dans le répertoire `documents`, il faudra exécuter la commande :

```
cd ..
```

ou

```
cd /home/elsa/documents
```

4. Pour revenir à son répertoire personnel, quel que soit l'endroit où on se trouve, il suffit d'exécuter la commande

```
cd
```

ou

```
cd ~
```

Exercice 2

Question : En utilisant toujours cette même arborescence, quelles sont les commandes à exécuter si le répertoire courant est `home` et que vous souhaitez vous rendre dans le répertoire `boulot` ? *Vous donnerez les chemins relatif et absolu.*

Lister le contenu d'un répertoire : commande `ls`

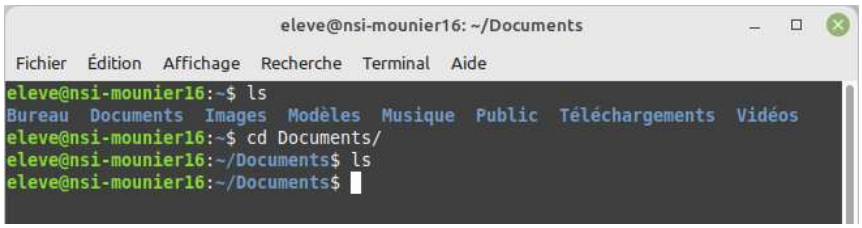
Pour lister le contenu (répertoire et fichiers) du répertoire courant, il suffit d'exécuter la commande `ls` (pour *list*, soit *lister*).

Exemples :

1. Si l'utilisateur `elsa` se trouve dans son répertoire `documents`, alors l'exécution de la commande `ls` va lister le contenu de ce répertoire

```
elsa@ordi:~/documents$ ls
boulot fiche.ods
```

2. Les commandes et les résultats ci-dessous indiquent que le répertoire personnel de l'utilisateur `eleve` contient 8 répertoires (`Bureau`, `Documents`, ...) et que le répertoire `Documents` est vide puisque que la commande `ls` ne renvoie rien lorsqu'on se trouve dans ce répertoire.



En particulier, exécuter `ls` à partir d'un répertoire vide ne produit aucun résultat (ce qui est logique).

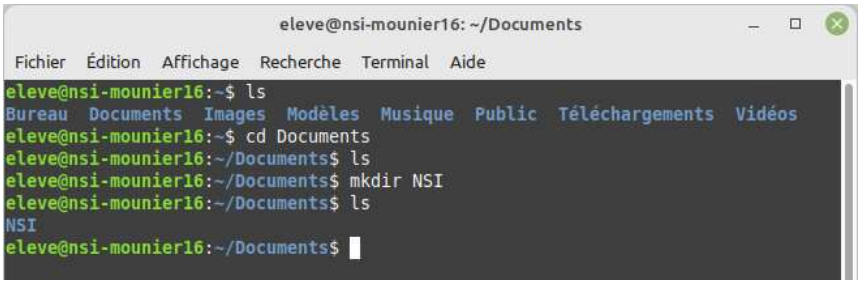
> Essayez d'utiliser la commande `ls` dans différents répertoires et comparez avec le navigateur de fichiers.

Créer un répertoire : commande `mkdir`

La commande `mkdir`, pour **make directory** (soit *créer un répertoire*), permet de créer un répertoire dans le répertoire courant. Il suffit de faire suivre cette commande du nom du répertoire à créer :

```
mkdir nom_repertoire
```

Exemple : On crée un répertoire `NSI` dans le répertoire `Documents` avec ce qui suit (on a aussi listé les éléments pour vérifier avec `ls` et utilisé `cd` pour se déplacer dans l'arborescence)



Exercice 3

Dans cet exercice, vous allez utiliser le Terminal pour effectuer les différentes questions. Vous écrirez sur votre feuille les lignes de commandes correspondant aux différentes questions dès que vous aurez testé sur votre machine.

💻 **Question 1** : Placez-vous dans votre répertoire personnel (commande `cd`) puis listez tous les éléments avec `ls`.

💻 **Question 2** : Créez une répertoire appelé `OS` et vérifiez qu'il a bien été créé avec `ls`.

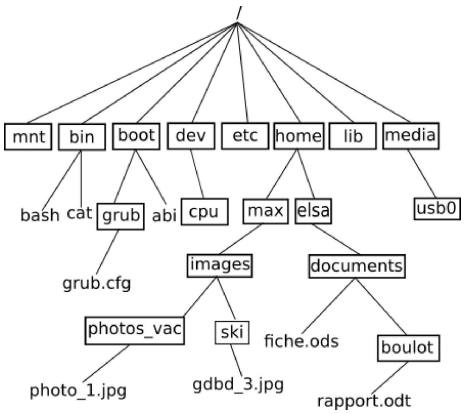
💻 **Question 3** : Déplacez-vous dans le répertoire `OS` avec `cd` puis créez un sous-répertoire appelé `test`. Vérifiez qu'il a bien été créé..

💻 **Question 4** : Déplacez-vous dans le répertoire `test` avec `cd` puis créez un sous-répertoire appelé `commandes`. Vérifiez qu'il a bien été créé..

💻 **Question 5** : Revenez dans le répertoire `OS` puis créez un répertoire `test2` et vérifiez qu'il a bien été créé.

Exercice 4

On reprend l'arborescence de notre cours.



✍ **Question** : Dessinez l'arborescence, à partir du répertoire `elsa`, après l'exécution des commandes suivantes :

```
$ pwd
/home/elsa
$ cd documents/boulot
$ mkdir premiere
$ cd premiere
$ mkdir nsi
$ cd ../..
$ mkdir projet
```

Supprimer des éléments : commande `rm`

La commande `rm`, pour **remove** (soit *supprimer*), permet de supprimer un fichier ou un répertoire. Il suffit de faire suivre cette commande du nom du répertoire ou du fichier à supprimer:

```
rm nom_repertoire_ou_nom_fichier
```

La plupart des commandes peuvent être utilisées avec des options. Si on veut supprimer un répertoire non vide, il faut utiliser la commande `rm` avec l'option `-r` (pour *récursif*)

```
rm -r nom_repertoire
```

mais il faut être vigilant car cela supprime le répertoire et tout son contenu (sous-répertoires et fichiers).

Créer un fichier : commande `touch`

La commande `touch` permet de créer un fichier vide. Il suffit de faire suivre cette commande du nom du fichier à créer :

```
touch nom_fichier.extension
```

Attention, il faut indiquer l'extension du fichier : par exemple `touch main.py` va créer un fichier Python vide appelé "main".

Copier un fichier : commande `cp`

La commande `cp`, pour **copy** (soit *copier*), permet de copier un fichier. Il suffit de faire suivre cette commande du chemin (absolu ou relatif) du fichier à copier puis du chemin de destination de la copie :

```
cp /repertoire_source/nom_fichier_a_copier /repertoire_destination/nom_fichier
```

Remarques :

- Vous remarquerez l'espace entre le chemin du fichier de départ et le chemin du fichier destination.
- Le nom du fichier "destination" n'est pas forcément le même que celui du fichier "source" : on peut avoir `cp main.py ../essai.py`.

Déplacer un fichier : commande `mv`

La commande `mv`, pour **move** (soit *déplacer*), permet de déplacer un fichier. Il suffit de faire suivre cette commande du chemin (absolu ou relatif) du fichier source (celui à copier) puis du répertoire cible. Si la cible est un répertoire alors la source est copiée dedans, sinon elle est renommée.

Exemples :

1. Pour déplacer le fichier `bar.txt` dans le répertoire `baz` :

```
mv foo/bar.txt baz/
```

2. Pour renommer le fichier `foo_bar.txt` en `foo_baz.txt` :

```
mv foo_bar.txt foo_baz.txt
```

Exercice 5

Cet exercice est la suite de l'exercice 3. En particulier, vous devez partir avec l'arborescence suivante dans votre répertoire personnel :

```
/home/premiere
|__ /OS
    |__ /test
    |   |__ /commandes
    |__ /test2
```

Vous allez utiliser le Terminal pour effectuer les différentes questions. Vous écrirez sur votre feuille les lignes de commandes correspondant aux différentes questions dès que vous aurez testé sur votre machine.

💻 **Question 1** : Placez-vous dans votre répertoire personnel (commande `cd`) puis déplacez-vous dans le répertoire `/commandes`.

💻 **Question 2** : Créez des fichiers `commandes_1.txt`, `commandes_2.txt` et `essai.py` dans le répertoire `commandes` puis vérifiez avec `ls` qu'ils ont bien été créés.

💻 **Question 3** : Supprimez le fichier `essai.py` puis vérifiez qu'il a bien été supprimé.

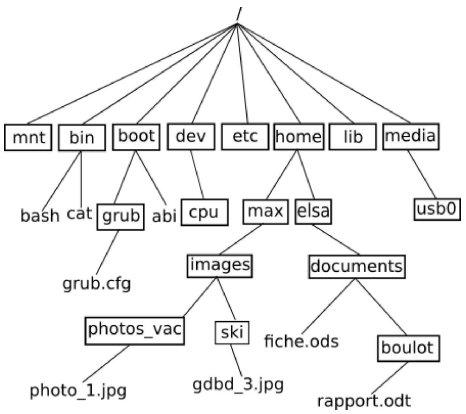
💻 **Question 4** : Copiez le fichier `commandes_1.txt` dans le répertoire `test2` et vérifiez qu'il a bien été copié.

💻 **Question 5** : Déplacez le fichier `commandes_2.txt` dans le répertoire `test2` et vérifiez qu'il a bien été déplacé (il ne doit plus se trouver dans le répertoire `test`).

💻 **Question 6** : Déplacez-vous dans le répertoire `OS` puis supprimez le répertoire `test` et tout son contenu. Vérifiez qu'il a bien été supprimé.

Exercice 6

On reprend l'arborescence de notre cours et **on suppose que l'utilisateur `max` est connecté**.



✍ **Question** : Dessinez l'arborescence, à partir du répertoire `max`, après l'exécution des commandes suivantes :

```
$ pwd
/home/max
$ cd images
$ mkdir lycee
$ cd lycee
$ touch nsi_1.jpg compte_rendu.odt
$ cp nsi_1.jpg ../photos_vac/nsi.jpg
$ cd ../lycee
$ rm nsi_1.jpg
$ cd ..
$ rm -r ski
```

Droits et permissions

Dans la partie précédente, vous avez par exemple vu comment supprimer des fichiers. Vous n'avez pas la possibilité de supprimer tout et n'importe quoi, et heureusement !

Un système d'exploitation de type "UNIX" est un système **multi-utilisateurs** : plusieurs utilisateurs peuvent se partager un même ordinateur, chacun ayant un environnement de travail qui lui est propre.

Il existe un utilisateur un peu particulier qui a tous les droits (ou presque), on l'appelle le "super utilisateur" ou encore l'**administrateur**, ou **root**. L'administrateur peut définir des **groupes** d'utilisateurs et attribuer plusieurs utilisateurs à un groupe pour ne pas avoir à gérer individuellement les différents utilisateurs.

Le système d'exploitation gère les *droits et permissions* sur les fichiers et les répertoires pour les différents *utilisateurs*.

Nous nous intéressons uniquement ici aux droits relatifs aux fichiers, mais il en existe d'autres liés aux autres éléments du système d'exploitation (imprimante, installation de logiciels, ...).

Utilisateurs et permissions

Pour un fichier, on distingue :

- trois types d'utilisateurs :
  - le *propriétaire* noté `u` (pour *user*)
  - le *groupe principal* noté `g` (pour *group*)
  - les *autres utilisateurs* notés `o` (pour *others*, ceux qui n'appartiennent pas au groupe associé au fichier)
- trois types de permissions :
  - *lecture* (caractère `r` si attribué ou `-` sinon) : pour lire le contenu
  - *écriture* (caractère `w` si attribué ou `-` sinon) : pour modifier le contenu
  - *exécution* (caractère `x` si attribué ou `-` sinon) : pour exécuter un fichier (exécutable)

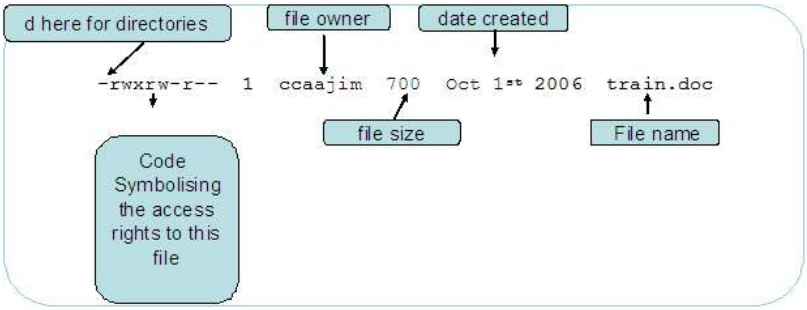


Fig. 3 - Vue des droits et permissions d'un fichier.
Crédits : Jimbotyson, CC BY-SA 3.0, via Wikimedia Commons

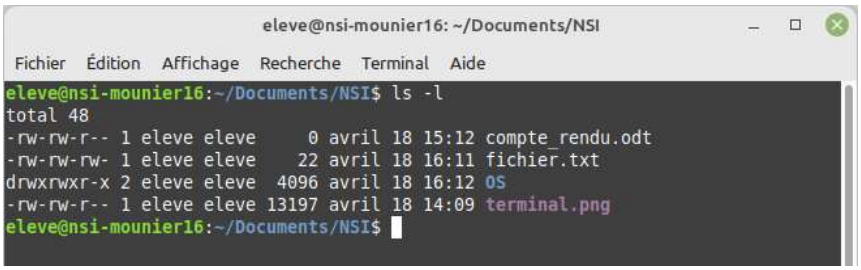
Pour les répertoires, la signification des permissions est précisée dans ce tableau. Attention, si on a le droit d'écriture sur un répertoire on peut supprimer tous les fichiers qu'il contient même ceux dont on n'est pas propriétaire !

	Type de permission	pour un répertoire

`r`	lecture	lister le contenu
`w`	écriture	ajouter, supprimer, renommer des fichiers
`x`	exécution	entrer dedans

Exemples de lectures de permissions

On peut afficher les droits des fichiers d'un répertoire avec la commande `ls -l` :



On trouve la syntaxe suivante :

Droits et permissions (10 premiers caractères)	Nombre de liens	propriétaire	groupe	taille (en octets)	date modification	nom
`-rw-rw-r--`	1	eleve	eleve	13197	avril 18 14:09	`terminal.png`

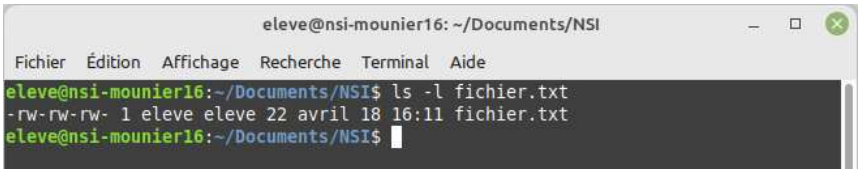
Analyse : Chaque fichier ou répertoire possède un *propriétaire* et un *groupe* (nous ne parlerons pas du nombre de liens dans ce cours). Intéressons-nous surtout aux dix premiers caractères qui définissent l'ensemble des droits et permissions, avec dans l'ordre :

- le premier caractère, ici `-`, indique la nature de l'élément : `-` s'il s'agit d'un fichier, `d` s'il s'agit d'un répertoire (*directory*) et `l` s'il s'agit d'un raccourci vers un autre élément (*link*)
- les trois suivants, ici `rw-`, indiquent les permissions pour le **propriétaire** (u) : ici droit en lecture, en écriture mais pas en exécution (c'est normal car il s'agit d'un fichier image qui ne s'exécute pas)
- les trois suivants, ici `rw-`, indiquent les permissions pour le **groupe** (g) : les mêmes que pour le propriétaire dans notre cas
- les trois derniers, ici `r--`, indiquent les permissions pour les **autres utilisateurs** : ici uniquement le droit en lecture, cela veut dire que n'importe quel autre utilisateur ne pourra pas modifier ce fichier (en particulier pas le supprimer).

En résumé cela donne :

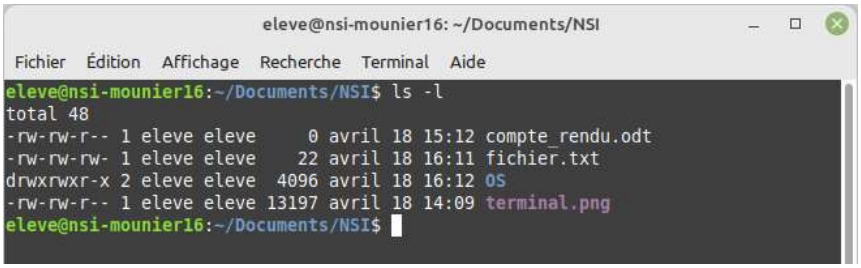
	Lecture	Écriture	Exécution
Propriétaire	oui	oui	non
Groupe	oui	oui	non
Autres	oui	non	non

Si on veut connaître les droits d'un seul fichier ou répertoire il suffit d'indiquer son nom après `ls -l` :



Exercice 7

Question : Analysez les droits et permissions des 3 autres éléments du répertoire `NSI` listés ci-dessous.



Exercice 8

💻 **Question 1** : Placez-vous dans votre répertoire personnel puis déplacez-vous dans le répertoire `/OS` (créez-le au préalable s'il n'existe pas). Créez un fichier `essai.txt` avec la commande `touch`.

💻 **Question 2** : Listez les droits et permissions de tous les éléments du répertoire `OS` avec `ls -l` et analysez les permissions pour le fichier `essai.txt`.

Modifier les permissions d'un fichier avec `chmod`

Seul le propriétaire d'un fichier (ou répertoire) ou l'utilisateur `root` (administrateur) peuvent modifier les permissions d'un fichier (ou répertoire) avec la commande `chmod`.

Cette commande s'utilise avec différentes options facultatives selon la syntaxe suivante :

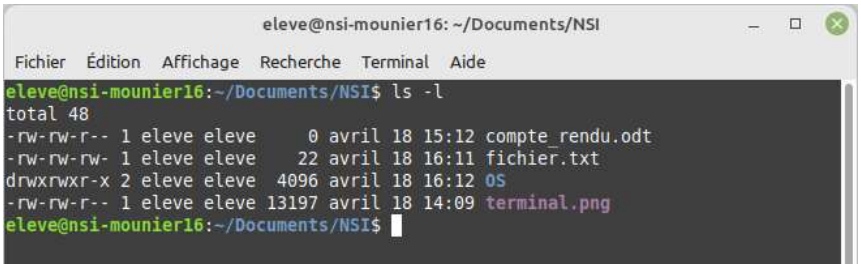
```
chmod [ugoa][+-=][rwx] fichier
```

Les options entre crochets désignent :

- `u` : le propriétaire
- `g` : le groupe
- `o` : les autres utilisateurs
- `a` : tous les utilisateurs (raccourci pour `ugo`)
- `+` : ajouter le(s) droit(s)
- `-` : enlever le(s) droit(s)
- `=` : positionner le(s) droit(s)
- `r` : droit de lecture
- `w` : droit d'écriture
- `x` : droit d'exécution
- `-R` : récursivement (nécessaire pour agir sur un répertoire)

Exemples :

On suppose que l'on part avec les permissions ci-dessous :



1. Pour ajouter le droit d'écriture aux autres utilisateurs sur le fichier `compte_rendu.odt`

   ```
   chmod o+w compte_rendu.odt
   ```

   Dans ce cas, tous les autres utilisateurs pourront modifier le fichier en question.

2. Pour retirer le droit de lecture au groupe sur `terminal.png` :

   ```
   chmod g-r terminal.png
   ```

   Dans ce cas, les utilisateurs du groupe ne pourront plus lire le fichier en question.

3. Pour retirer les droits de lecture et d'écriture au groupe et aux autres utilisateurs sur `fichier.txt` :

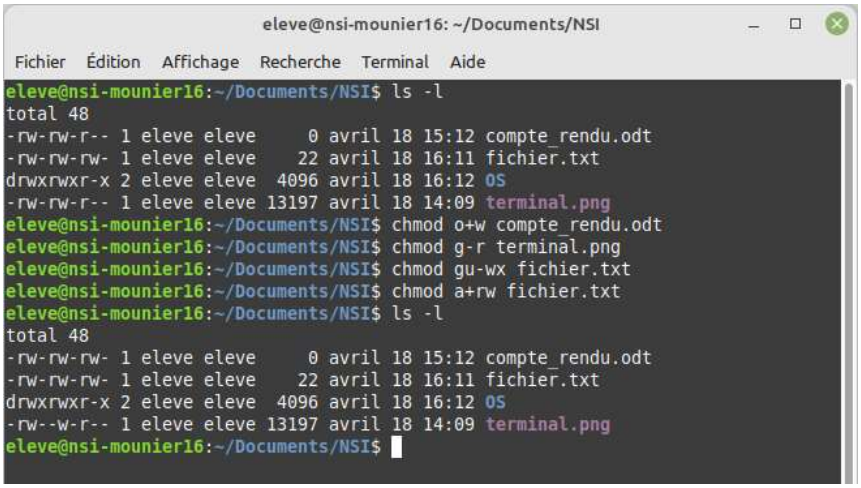
   ```
   chmod go-rw fichier.txt
   ```

   Dans ce cas, ni les membres du groupe, ni les autres utilisateurs ne pourront ni lire ni écrire dans ce fichier.

4. Pour ajouter les droits de lecture et d'écriture à tous les utilisateurs sur `fichier.txt` :

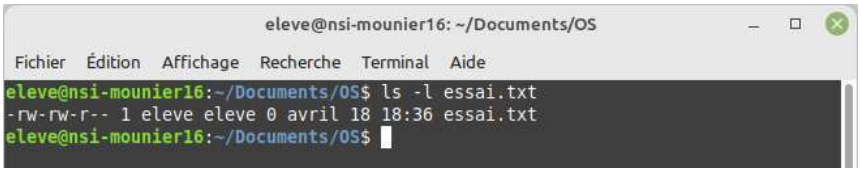
```
chmod a+rw fichier.txt
```

Après ce différentes modifications, on obtient les permissions suivantes :



Exercice 9

On reprend ce qui a été fait à l'exercice 8. En particulier, dans votre répertoire personnel vous devez avoir un répertoire `OS` qui contient un fichier `essai.txt` (vide pour l'instant) qui possède les permissions suivantes :



✍ **Question 1** : Analysez les permissions de ce fichier.

✍💻 **Question 2** : Quelle commande permet d'enlever les droits de lecture aux autres utilisateurs ? Exécutez cette commande et vérifiez que les permissions ont bien changé.

💻 **Question 3** : En étant placé dans le répertoire `OS`, écrivez et exécutez l'instruction suivante, qui permet d'écrire dans le fichier `essai.txt` :

```
echo "Un texte très court." > essai.txt
```

💻 **Question 4** : Ouvrez ensuite ce fichier avec un éditeur de texte (via le navigateur de fichiers) pour vérifier que le texte a bien été écrit dans le fichier. Rajouter une phrase à ce fichier et enregistrez-le.

> ℹ Vous avez pu faire cela car vous possédez le droit d'écriture sur ce fichier.

💻 **Question 5** : Écrivez la commande permettant de retirer le droit d'écriture au propriétaire sur le fichier `essai.txt`. Vérifiez que vous ne pouvez plus modifier son contenu puis fermer l'éditeur de texte.

💻 **Question 6** : Écrivez la commande permettant de retirer le droit de lecture au propriétaire sur le fichier `essai.txt`. Vérifiez que vous ne pouvez même plus l'ouvrir avec un éditeur de texte.

💻 **Question 7** : Écrivez la commande (une seule !) permettant d'ajouter le droit de lecture et d'écriture au propriétaire sur le fichier `essai.txt`. Vérifiez que vous pouvez à nouveau lire et modifier ce fichier.

Exercice 10

💻 **Question 1** : Le répertoire `/usr/bin` contient les exécutables des applications. Déplacez-vous dans ce répertoire et lister tout son contenu. Vous devrez y trouver :

- l'exécutable `chmod` qui correspond au programme exécuté lors que l'on utilise la commande `chmod` dans le terminal
- l'exécutable `python3.8` qui correspond au programme exécutant Python sur votre machine

💻✍ **Question 2** : Affichez les permissions pour ces deux fichiers. Avez-vous le droit d'exécution sur ces deux fichiers ? Expliquez.

💻 **Question 3** : Essayez de supprimer ces fichiers avec `rm`. Expliquez pourquoi ce n'est pas possible.

✍ **Question 4** : Expliquez pourquoi vous pouvez utiliser les programmes `chmod` et `python3.8` (dans un terminal par exemple).

💻 **Question 5** : Déplacez-vous dans le répertoire `~/OS/` et créez un fichier `essai.py`. Ouvrez ce fichier avec un éditeur de texte et écrivez le code

```
t = [1, 2, 3]
for e in t:
    print(e)
```

Exécutez ensuite la commande `python3.8 essai.py` qui permet d'exécuter votre programme Python directement dans le Terminal (vous ne devriez pas être surpris de l'affichage).

En réalité il suffit de taper `python3 essai.py` pour exécuter le fichier Python par l'exécutable `python3` (qui est aussi dans `/usr/bin`) renvoie vers `python3.8`.

De plus, si vous tapez simplement la commande `python3`, vous ouvez un interpréteur Python directement dans le Terminal. Essayez ! (Pour quitter il faut appeler la fonction `exit()`)

Bilan

- Il n'est pas nécessaire d'utiliser (ni d'avoir) une interface graphique pour utiliser un ordinateur : on peut tout faire en lignes de commandes dans un terminal.
- Les systèmes d'exploitation "UNIX" (comme Linux, MacOS) reposent entièrement sur un système de fichiers et de répertoires (tout est fichier ou répertoire). Ce système de fichiers est organisé sous forme d'un arbre.
- Il existe des commandes permettant de se déplacer dans cette arborescence et de manipuler les fichiers (nous en avons vu certaines mais il en existe bien d'autres). **Vous trouverez un mémento des commandes Linux en suivant ce lien : https://juliend.github.io/linux-cheatsheet/**
- Les systèmes multi-utilisateurs permettent à plusieurs utilisateurs d'utiliser un même ordinateur, chacun ayant des droits et permissions différentes et une interface propre. C'est le système d'exploitation qui gère finement tout cela. Il existe un utilisateur spécial, qui a tous les droits sur les fichiers, il s'appelle *superutilisateur* ou *administrateur* ou *root* et possède tous les droits sur les fichiers et répertoires.
- Pour chaque fichier (ou répertoire), on distingue trois types d'utilisateurs : propriétaire (`u`), groupe (`g`) et autres (`o`); et trois types de permissions : droit de lecture (`r`), d'écriture (`r`) et d'exécution (`x`).
- On peut modifier ces droits avec la commande `chmod` à condition d'avoir les droits pour le faire !
- Ce cours n'était qu'une introduction aux commandes bash et aux droits et permissions sur les fichiers. Vous pouvez aller plus loin si vous le souhaitez avec des exercices supplémentaires sur cette page : http://frederic-junier.org/NSI/premiere/chapitre9/TP2/NSI-TP2-systeme-git/.

Références :

- Cours de David Roche sur les Commandes Linux
- La documentation Ubuntu pour l'aborescence de Linux : https://doc.ubuntu-fr.org/arborescence
- Cours de Frédéric Junier : Système d'exploitation et lignes de commande (licence CC BY-NC-SA 4.0)
- Mémento en ligne des commandes Linux de Julien Dubreuil : https://juliend.github.io/linux-cheatsheet/

Voir en ligne : info-mounier.fr/premiere_nsi/archi_os/lignes-commandes-droits