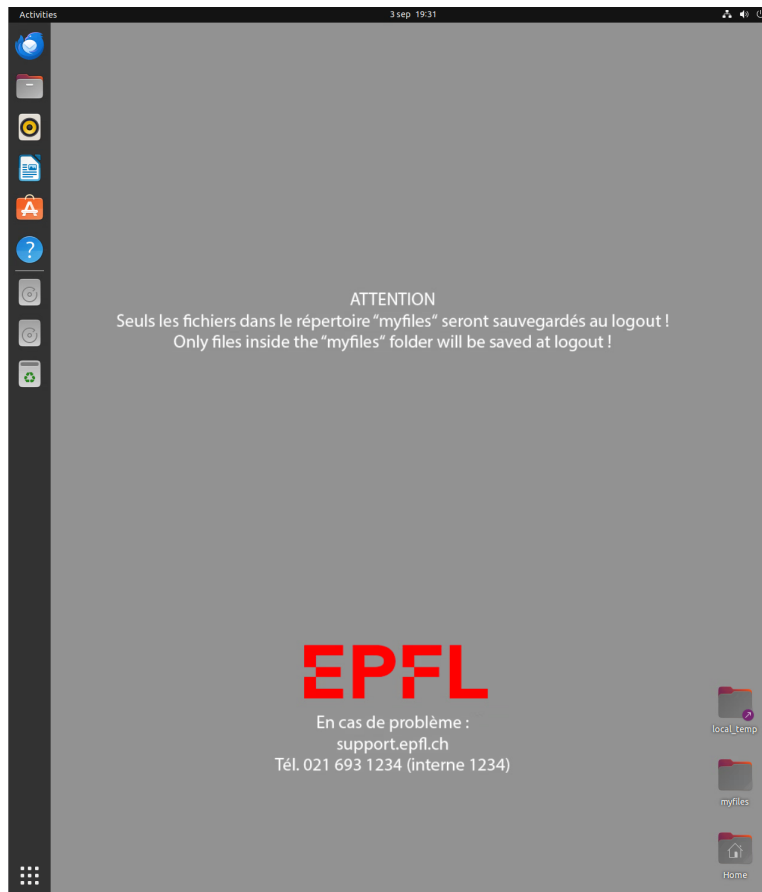

Session d'exercices – Découverte de l'environnement Unix

Le but de cette série d'exercices est de vous offrir une première prise de contact avec l'environnement informatique de l'EPFL et

- permettre de vous familiariser avec votre compte informatique
- faire connaître des principales manipulations de base pour évoluer dans un environnement Unix que vous serez amenés à utiliser
- manipulation et édition de fichiers, commandes Unix, etc.

Exercice 1 : Utilisation du gestionnaire de fichiers

Lorsque vous vous connectez sur votre compte, vous arrivez au bureau qui vous appartient.



Pour ouvrir votre répertoire personnel dans le gestionnaire de fichiers, entrez dans le répertoire nommé **myfiles**. Il est extrêmement important:

En 2011, il a été décidé au niveau de l'École d'installer toutes les données liées aux utilisateurs sur un nouveau serveur central, le service myNAS (<http://mynas.epfl.ch>). C'est un service offert à tous les étudiants et collaborateurs de l'EPFL. Le quota y est de 2 Go pour les étudiants. Ce répertoire est global à toute l'École et suivra l'étudiant tout au long de ses études. Il est le même sous Windows, Mac et Linux dans les différentes salles de l'École.

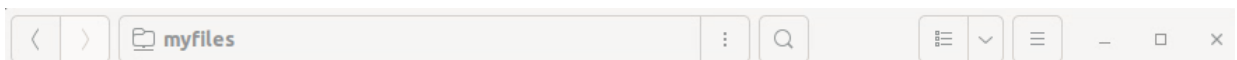
Important :

- C'est le répertoire **myfiles** qui vous donne l'accès à ces données.
- **Toutes les données stockées en dehors de ce répertoire sont détruites à chaque fois que vous vous déconnectez!**
- Il est donc impératif de systématiquement tout stocker dans **myfiles**.

Pour ceux d'entre vous qui sont un peu plus avancés: cela signifie aussi que les fichiers de configuration et de préférence situés directement dans le répertoire principal («Home») sont perdus à chaque déconnexion. Il faut, pour le moment, veiller à les archiver soi-même dans myfiles (et les restaurer au bon endroit).

Dans **myfiles** vous pouvez entre autres :

- créer de nouveaux répertoires (en sélectionnant «New Folder» dans le menu qui apparaît avec le clic droit de la souris) ;
- copier des fichiers ou des répertoires ;
- parcourir simplement la structure des répertoires (par exemple, en double-cliquant sur les dossiers qui sont présents dans la fenêtre et en utilisant les deux flèches de navigation) :



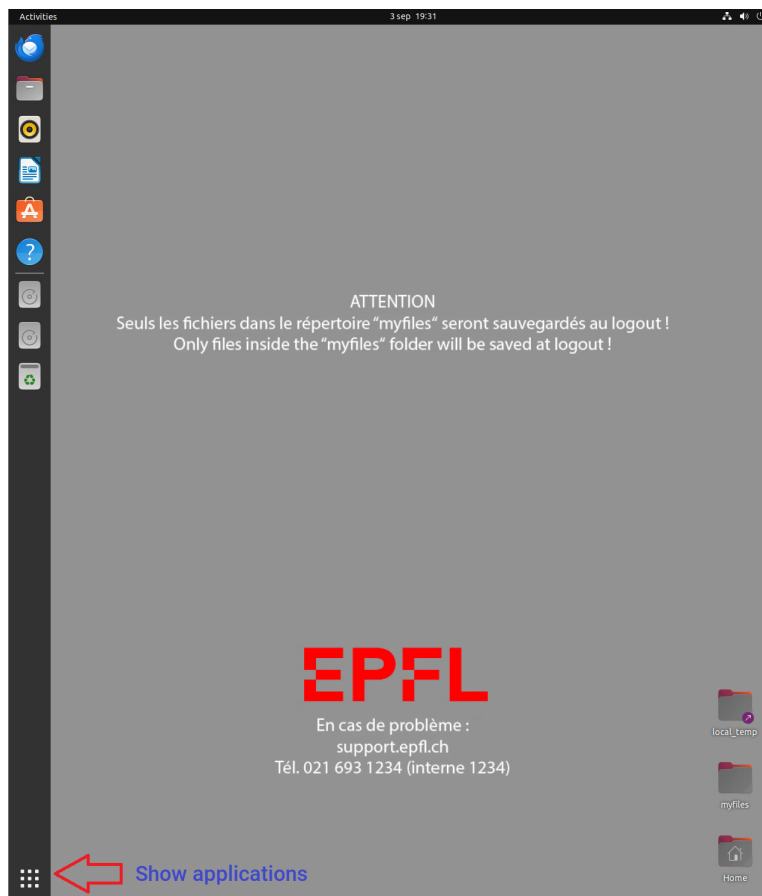
- effacer des fichiers ou des répertoires. (Par exemple, en les sélectionnant avec le bouton gauche de la souris, et en sélectionnant «Move to Trash» dans le menu qui apparaît en cliquant sur la sélection avec le bouton droit. Vous pouvez aussi plus simplement appuyer sur la touche «Del».)

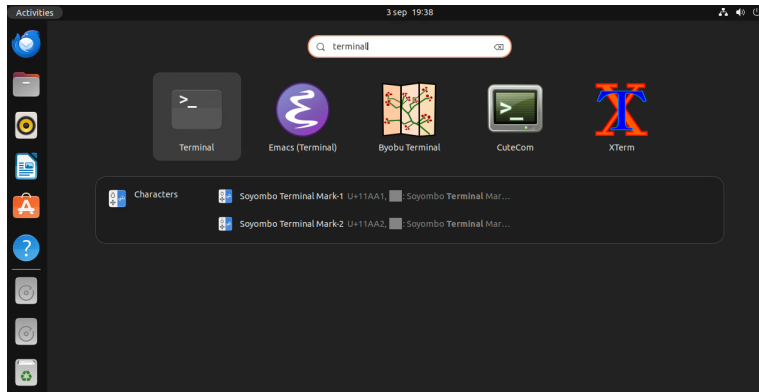
Remarquez que les documents/fichiers qui se trouvent sur votre bureau sont stocké dans le répertoire **Desktop** et donc dehors de **myfiles**! En conséquence, ils sont détruits quand vous vous déconnectez.

Exercice 2 : Ouvrir un terminal

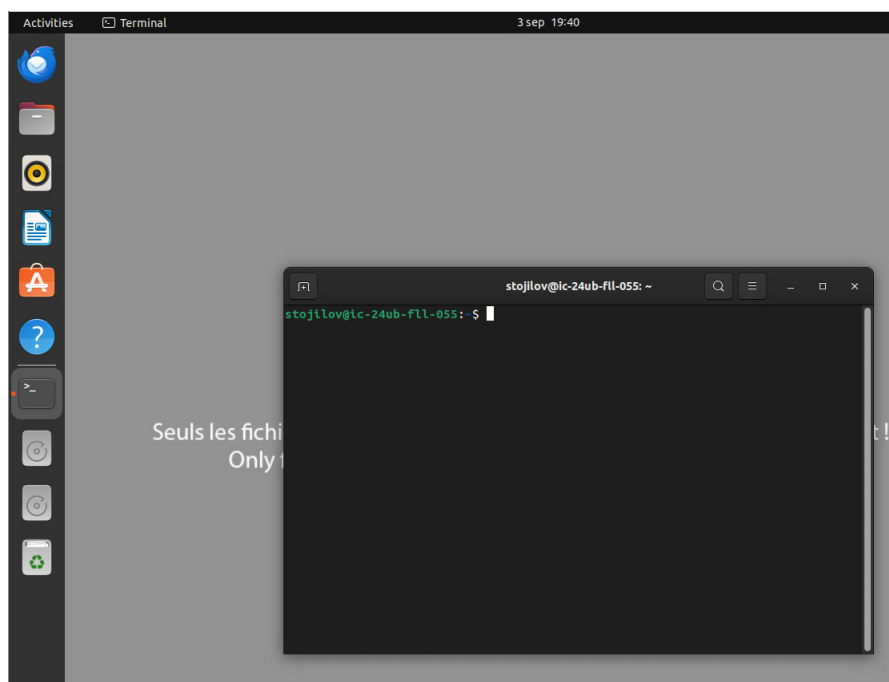
Vous étiez probablement déjà familier avec le gestionnaire de fichiers que nous venons de voir, mais il existe un autre outil qui permet de faire les mêmes choses et bien plus : il s'agit du «terminal». Un terminal (ou ligne de commande, ou console) est une interface qui nous permet d'interagir directement avec le système d'exploitation en tapant des commandes au clavier, sans avoir à cliquer sur des boutons. Les premiers ordinateurs n'avaient que la ligne de commande. Le concept d'une interface utilisateur graphique (GUI) a été développé plus tard, quand les ordinateurs sont devenus plus puissants et capables de gérer les fonctionnalités avancées qui nous paraissent aujourd'hui normales. Malgré les interfaces graphiques avec les souris, la ligne de commande a survécu. Pourquoi ? Parce que, quand on sait s'en servir, on va beaucoup plus vite avec le terminal qu'avec l'interface graphique. En plus, il y a des choses avancées que seul le terminal peut faire.

Commençons par ouvrir un terminal. Pour cela, appuyez sur **CTRL+ALT+T**, ou cliquez en bas à gauche, puis commencez une recherche en tapant «terminal». Cliquez enfin sur l'icône de «Terminal» pour l'activer.



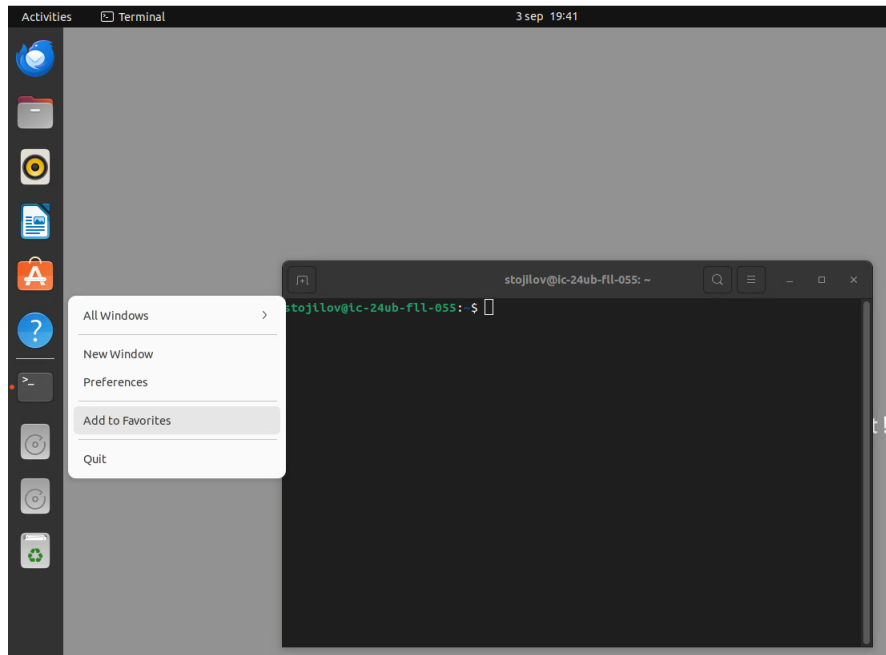


Vous avez créé une fenêtre pour communiquer avec votre système d'exploitation !



Notez qu'on peut avoir plusieurs terminaux ouverts à la fois, de la même manière qu'on peut avoir plusieurs fenêtres d'un gestionnaire de fichiers ouvertes à la fois. On peut voir le terminal comme une fenêtre à travers laquelle on peut interagir avec le système d'exploitation de l'ordinateur : on peut le faire à travers plus d'un terminal.

Si vous souhaitez garder un raccourci vers une application que vous utilisez souvent (et c'est conseillé de le faire pour le terminal), il suffit de faire un clic-droit sur l'icône correspondante à gauche de l'écran et choisir «Add to favorites» :



Exercice 3 : Les commandes Unix

Pour communiquer avec le système d'exploitation à travers le terminal, nous sommes forcés de taper des commandes au clavier. Les systèmes d'exploitation de type UNIX offrent à leurs utilisateurs des centaines de commandes qui font de la console un outil pratique et extrêmement puissant. Vous pouvez vous référer à cette liste des commandes les plus fréquemment incluses dans un système UNIX (et donc aussi Ubuntu Linux) : [Lien sur Wiki](#). Il y a beaucoup de commandes utiles, mais nous n'allons en voir qu'une petite sélection ici.

Toute interaction avec le système d'exploitation à travers le terminal se fait en tapant des *commandes* au clavier, qui prennent la forme suivante :

```
commande <argument1> <argument2> <argument3> ...
```

Le premier mot tapé spécifie quel type de commande effectuer, puis ce qui suit indique comment elle doit être effectuée. Certaines commandes peuvent avoir besoin de plus d'un argument : on verra par exemple une commande qui déplace un objet d'un dossier A vers un dossier B ; la commande aura donc besoin à la fois de A et de B comme arguments.

Lorsque la commande et ses arguments ont été entrés au clavier sur une ligne du terminal, taper «ENTER» pour lancer l'exécution de la commande.

Puisque les arguments d'une commande doivent être séparés par un espace, on peut se demander quoi faire si un argument *contient* un espace : le terminal risquerait de l'interpréter comme deux arguments séparés ! Pas de panique, les programmeurs ont pensé à tout, et il vous suffit de précéder l'espace dans l'argument par un *backslash* ('\'). En règle générale, s'il existe un caractère que le terminal a envie de traiter de manière spéciale (ici l'espace, qui est traité comme un séparateur d'arguments), on peut le forcer à le voir comme n'importe quel autre caractère en le précédant par un backslash. Ici, cela permet de dire au terminal de ne pas voir l'espace comme un séparateur, et de traiter l'ensemble comme un seul argument.

- Essayons la commande `echo`, qui affiche une ligne de texte donnée en paramètre : tapez la commande `echo` suivie par un mot ou une phrase, et appuyez sur la touche «ENTER». Par exemple, tapez :

```
echo Hi there!
```

Notez que le problème du caractère espace présenté plus haut n'apparaît pas ici car `echo` ne prend de toute façon qu'un seul argument, et est donc assez intelligent pour comprendre que l'espace n'est pas un séparateur.

- Essayez maintenant la commande `clear` : elle ne prend aucun argument, mais cache le contenu dans la fenêtre et la fenêtre de votre terminal apparaît vide. Attention : cette commande ne fait que nettoyer l'*affichage* du terminal, elle n'annule aucune commande que vous avez exécutée plus tôt. C'est comme vider sa boîte mail : les mails qui ont été envoyés restent envoyés.
- La commande `man` prend un seul argument : le nom d'une autre commande. Elle affiche des informations sur la commande donnée en argument, telles que le format attendu pour ses arguments, ce qu'elle fait, etc. Par exemple : `man echo` donne tout ce qu'il y a à savoir sur la commande `echo`.

.....

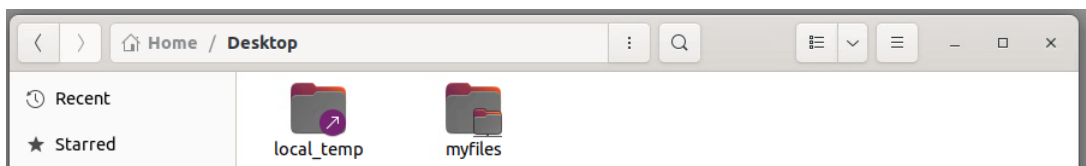
Les prochains exercices présentent d'autres commandes très utiles. Il est donc temps de noter que, pour répéter une des commandes que vous avez déjà entrées (et donc éviter de taper la même chose plusieurs fois) :

1. Avec les flèches ↑ et ↓ (sur le clavier), vous pouvez naviguer à travers l'historique des commandes que vous avez tapées dans ce terminal.
2. Une fois trouvé la commande que vous voulez répéter, utilisez les flèches ← et → pour l'éditer si nécessaire.
3. Finalement, exécutez-la en appuyant sur la touche «ENTER».

Exercice 4 : Naviguer les répertoires

Nous vous avons dit que le terminal peut faire tout ce que le gestionnaire de fichier peut faire, et bien plus. Voyons donc déjà comment faire ce que le gestionnaire de fichiers peut faire !

La structure des fichiers de votre ordinateur vous est certainement connue : vous disposez d'un dossier central dans lequel se trouvent vos dossiers Documents, Téléchargements, etc. Ces dossiers peuvent eux-mêmes contenir d'autres dossiers, et ainsi de suite. Le gestionnaire de fichier vous permet de vous déplacer dans vos dossiers, et il vous affiche toujours dans quel dossier vous vous situez en haut de la fenêtre :

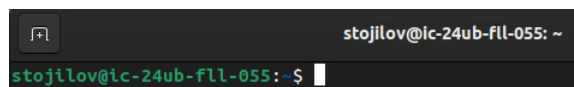


Exemple si l'on se trouve dans le dossier Home/Desktp.

Notez que ce dossier central, dans lequel se trouvent vos Documents, Desktop, etc, s'appelle ici Home : il s'agit du dossier racine de tout ce qui est destiné à l'utilisateur sur l'ordinateur. Notez aussi que l'on utilisera plutôt le terme «répertoire» au lieu de «dossier», ce dernier étant surtout destiné au contexte de l'interface graphique.

Ce que le gestionnaire de fichiers nous affiche en haut de la fenêtre (Home/Desktop dans notre exemple) est en fait le *chemin d'accès* du répertoire courant, c'est à dire les dossiers à parcourir en partant de Home pour arriver dans le dossier courant. Comme vous l'avez vu, on l'écrit en séparant chaque dossier par un slash : [Home/Desktop](#). Et puisque Home est un dossier un peu spécial que l'on verra souvent, les programmeurs ont décidé de lui attribuer un alias plus court : le caractère tilde ('~'). On peut donc écrire [~/Desktop](#) de manière équivalente.

Vous aurez peut-être remarqué que ce symbole spécial se trouve dans le terminal :



Un certain nombre d'informations utiles sont en fait affichées en début de chaque ligne dans le terminal, dans différentes couleurs. Vous remarquerez par exemple votre identifiant gaspar en début de ligne, avant le signe '@'. Ce qui nous intéresse ici est ce qui se situe juste avant le '\$' : notre ami tilde ! Comme le faisait le gestionnaire de fichiers en haut de fenêtre, c'est ici que le terminal vous affiche le chemin d'accès de votre répertoire courant. **Pensez à toujours garder un oeil sur votre répertoire courant : toute commande que vous exécutez dans le terminal s'applique dans ce répertoire.** C'est comme, par exemple, lorsque vous créez un nouveau dossier dans le gestionnaire de fichiers : le nouveau dossier sera situé dans le dossier courant.

Pour naviguer dans les répertoires, c'est à dire pour changer de répertoire courant, on utilise la commande `cd` (pour *change directory*, ou changer de répertoire)

`cd` prend un seul argument, le chemin du répertoire vers lequel on veut se déplacer :

```
cd <chemin du répertoire>
```

où l'on utilise la notation <chemin du répertoire> comme "placeholder", pour indiquer la forme générale de la commande.

Ce "chemin" prendra toujours la forme d'une séquence de noms de répertoires séparés par des slash '/'. Comme nous l'avons vu plus haut, toute commande s'exécute *dans le répertoire courant* : si le chemin a la forme A/B/C, le terminal va donc chercher un répertoire A *dans le répertoire courant*, dans lequel il en cherchera un appelé B, etc.

Quelques exceptions à cette règle existent bien sûr :

- '..' est un nom de répertoire spécial qui correspond au dossier *parent* du dossier courant : taper `cd ..` revient donc à sortir du répertoire courant, et `cd ../Documents` revient à aller dans le dossier `Documents` qui se situe dans le dossier parent.
- '~', que nous avons déjà vu, correspond à votre dossier `Home`. Taper `cd ~` vous ramènera donc toujours dans votre dossier `Home`, et `cd ~/Desktop` vous déplacera dans le dossier `Desktop` se situant dans votre dossier `Home`. D'ailleurs puisque `cd ~` est une commande dont on a souvent besoin, taper `cd` tout court est un raccourci équivalent.
- '/' est la *racine* de l'arborescence du système de fichiers. De la même manière que le dossier `Home` est la racine des fichiers destinés à l'utilisateur, '/' est la racine des fichiers destinés au système d'exploitation. Vous pourriez donc taper `cd /` pour vous déplacer dans ce répertoire, mais vous n'aurez normalement jamais besoin de l'utiliser.

On peut donc taper, par exemple :

```
cd .. pour revenir au répertoire parent.
```

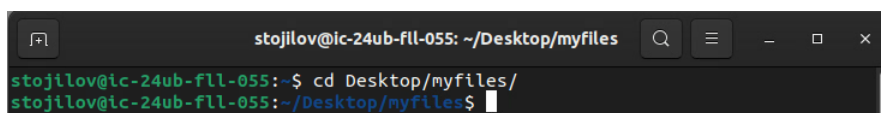
```
cd ../../ pour revenir au répertoire parent du répertoire parent.
```

```
cd ../../icc pour revenir au répertoire icc situé dans le répertoire parent du répertoire parent.
```

```
cd / pour aller à la racine de l'arborescence de fichiers.
```

```
cd (tout seul) pour revenir au répertoire Home de l'utilisateur.
```

Testons maintenant tout cela : ouvrez le terminal si ce n'est pas déjà fait et utilisez-le pour aller dans votre répertoire `myfiles/`, comme dans l'image ci-dessous :



```
stojilov@ic-24ub-fll-055: ~/Desktop/myfiles
stojilov@ic-24ub-fll-055:~$ cd Desktop/myfiles/
stojilov@ic-24ub-fll-055:~/Desktop/myfiles$
```


.....

Voyez comment le chemin d'accès du répertoire courant affiché par le terminal, en bleu, est mis à jour !

Essayez maintenant de revenir à votre repertoire «home» en utilisant la commande `cd ..`.

Pour afficher le chemin d'accès complet du répertoire courant dans le terminal, on utilise la commande `pwd` (pour *print working directory*, ou *afficher le répertoire courant*). Essayez-la ! Vous verrez que votre dossier Home se situe en fait quelque part dans le dossier `'/'`.

Découvrons maintenant la commande `ls` (pour *list contents*, ou *lister contenu*), qui affiche une liste des dossiers et fichiers présents dans le dossier courant. Essayez-la :

```
ls
```

Attention, le premier caractère est un L minuscule, pas le chiffre 1. Notez aussi le parallèle avec le gestionnaire de fichiers : ce que le terminal affiche avec `ls` est exactement la même chose que ce qu'affiche le gestionnaire de fichiers quand on ouvre un dossier !

Info pratique: Le terminal propose la complétion automatique, c'est-à-dire qu'en tapant la touche TAB, le terminal va tenter de compléter le mot que vous êtes en train de taper. Cela est très utile pour naviguer dans l'arborescence et pour éviter les fautes typographiques !

Quelques raccourcis utiles :

- CTRL + SHIFT + c — copy
- CTRL + SHIFT + v — paste
- CTRL + a — se déplacer au début de la ligne
- CTRL + e — se déplacer à la fin de la ligne
- CTRL + u — effacer tout jusqu'au début de la ligne
- CTRL + k — effacer tout jusqu'à la fin de la ligne
- CTRL + x + e — éditer la ligne dans un éditeur

Exercice 5 : Créer des répertoires

Pour créer des répertoires avec le terminal, on utilise la commande `mkdir` (pour *make directory*, ou *créer répertoire*).

`mkdir` prend un seul argument, le nom du répertoire à créer :

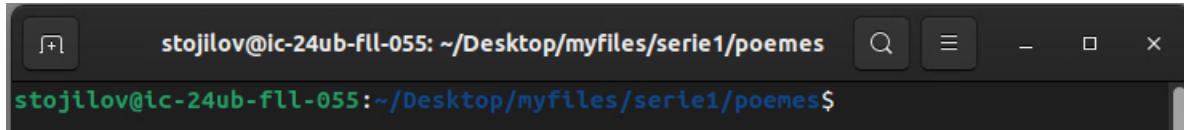
```
mkdir <répertoire>
```

Attention, comme toujours, cette commande s'exécute dans le dossier courant : le nouveau répertoire sera créé là où vous vous situez au moment d'entrer cette commande.

Remarque : Évitez les espaces dans les noms des fichiers ! Si, néanmoins, vous en voulez, il faut spécifier le nom du fichier entre guillemets.

À vous de jouer maintenant : dans `myfiles`, créez un nouveau répertoire appelé `serie1` et dans celui-ci un second répertoire appelé `poemes`.

Déplacez vous dans ce nouveau répertoire `poemes`.



```
stojilov@ic-24ub-fll-055: ~/Desktop/myfiles/serie1/poemes
stojilov@ic-24ub-fll-055:~/Desktop/myfiles/serie1/poemes$
```

Ensuite, à l'aide des commandes Unix suivantes, téléchargez les fichiers `poeme1.txt` et `poeme2.txt` dans le répertoire `myfiles/serie1/poemes`:

```
wget https://www.dropbox.com/s/syt2os03f8s5yte/poeme1.txt
wget https://www.dropbox.com/s/pyz16jlqyvrx84/poeme2.txt
```

Le nom de la commande `wget` vient de World Wide Web et de GET qui est le nom de la commande utilisée dans le protocole HTTP pour récupérer un fichier. `wget` est donc une commande qui télécharge un fichier depuis une adresse URL donnée en argument.

Utilisez maintenant la commande `ls` pour voir le contenu du répertoire `poemes`. Combien de fichiers y a-t-il dedans ?

Exercice 6 : Voir le contenu de fichiers

Pour lire les poèmes que vous venez de télécharger, on a plusieurs possibilités:

- ligne par ligne, avec la commande `less` (pour en sortir, appuyez sur 'q' comme "quit", ou tapez CTRL+C)
- page par page, avec la commande `more` (pour en sortir, appuyez sur 'q' comme "quit", ou tapez CTRL+C). On privilégiera souvent `less`, qui est plus puissant que `more`.
- avec un éditeur de texte (`gedit` ou un autre). Par exemple, la commande `gedit poeme1.txt` ouvrira le premier poème dans l'éditeur.

Lors de la visualisation de fichiers texte avec `more`, vous pouvez utiliser les contrôles suivants pour naviguer :

- Touche Entrée : fait défiler d'une ligne à la fois.
- Barre d'espace : passe à la page ou à l'écran suivant.
- Touche 'b' : revient d'une page en arrière.
- Touche 'q' » : comme déjà mentionné, quitte la commande `more` et ferme la vue.

Exercice 7 : Créer, copier, déplacer, renommer et détruire les fichiers

Pour créer un fichier vide, on utilise la commande `touch`, qui prend en argument le nom du fichier à créer. Essayez-la.

Pour copier un fichier d'un endroit à un autre, on utilise la commande `cp` (pour *copy*). Pour déplacer un fichier, on utilise `mv` (pour *move*). Leur fonctionnement est un peu particulier. Commençons par `mv`.

Tout d'abord, créez un répertoire appelé `icc` sur le bureau (Desktop) et un fichier `a.txt` avec un texte quelconque.

La commande `mv` permet de déplacer un fichier d'un répertoire à un autre, tout en renommant le fichier au passage si on le souhaite. Elle prend deux arguments :

```
mv <chemin1> <chemin2>
```

où `<chemin1>` et `<chemin2>` sont deux chemins d'accès, dont la forme et le fonctionnement sont identiques aux chemins d'accès que nous avons vus dans le contexte de la commande `cd`. Comme c'était le cas pour `cd`, ces chemins dépendent donc bien entendu aussi du répertoire courant.

- `<chemin1>` est le chemin d'accès du fichier à déplacer
- `<chemin2>` est le nouveau chemin d'accès du fichier.

Cela signifie qu'après avoir exécuté la commande `mv <chemin1> <chemin2>`, le fichier auparavant accessible à travers le chemin `<chemin1>` sera donc accessible à travers le chemin `<chemin2>`. Par exemple, après exécution de `mv Downloads/a.txt icc/a.txt` le fichier qui était accessible dans `Downloads` sous le nom `a.txt` sera accessible dans `icc` sous le nom `a.txt`.

Notez que la commande ne nous force pas à réutiliser le même nom de fichier dans le deuxième argument. Ainsi, exécuter `mv Downloads/a.txt icc/nouveau_a.txt` va non seulement déplacer `a.txt` de `Downloads` vers `icc`, mais va aussi le renommer en `nouveau_a.txt` ensuite ! `mv` peut donc aussi être utilisé pour renommer un fichier !

Pour finir, puisqu'il arrivera souvent de vouloir déplacer un fichier sans le renommer, la commande `mv` accepte aussi que le deuxième argument corresponde à un répertoire : elle déplacera alors simplement le fichier dans celui-ci, sans le renommer. La commande `mv Downloads/a.txt icc/a.txt` est donc équivalente à `mv Downloads/a.txt icc`.

Voici quelques exemples concrets d'utilisation :

```
mv icc/a.txt b.txt
```

 déplace `a.txt` de `icc` vers le dossier courant et le renomme `b.txt`,

```
mv a.txt ../b.txt
```

 déplace `a.txt` vers le dossier parent, en le renommant `b.txt`,

```
mv a.txt ../a.txt
```

 déplace `a.txt` vers le dossier parent, sans le renommer,

```
mv a.txt ..
```

 fait exactement la même chose, sans devoir retaper `a.txt`,

```
mv a.txt b.txt
```

 renomme simplement `a.txt` en `b.txt`, sans le déplacer.

Voyons maintenant la commande `cp` :

.....

`cp` s'utilise exactement comme `mv`. La seule différence entre les deux est que `cp` va garder une copie du fichier d'origine au lieu de le déplacer. Par exemple, dans un répertoire ne contenant qu'un fichier nommé `a.txt`, après avoir tapé `cp a.txt b.txt`, il existera deux fichiers identiques appelés `a.txt` et `b.txt`.

Enfin, pour détruire un fichier, on utilise la commande `rm` (*remove*) :

`rm <chemin1>` permet de détruire le fichier dont le chemin d'accès est `<chemin1>`.
Attention ! Il n'existe pas de "poubelle" dans le terminal : un fichier supprimé avec cette commande est irrémédiablement perdu !

Utilisons maintenant ce que nous avons appris :

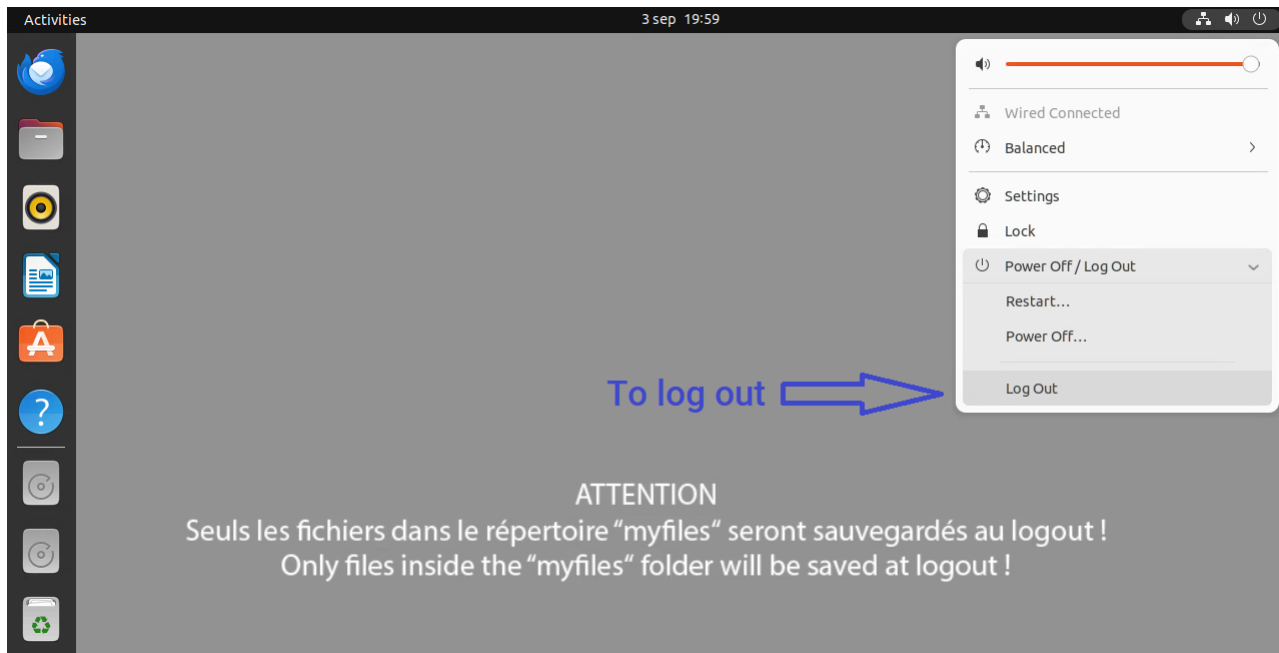
- Déplacez `poeme1.txt` de `myfiles/serie1/poemes` à `myfiles` en utilisant la commande `mv`. Aide: pour aller en arrière une fois dans le chemin, il faut entrer deux points (`..`), pour aller en arrière deux fois dans le chemin, il faut entrer deux points encore une fois (`../..`).
- Déplacez `poeme2.txt` de `myfiles/serie1/poemes` à `myfiles` en utilisant les commandes `cp` et `rm`.

Utilisez la commande `ls` pour voir le contenu du répertoire `poemes`. Combien de fichiers y-a-t-il dedans maintenant?

Se déconnecter (au logout)

ATTENTION ! Pour vous déconnecter de votre session de travail, il vous faut vous déconnecter **deux fois !**

Une première fois de la machine virtuelle : option Log Out du menu sous votre nom dans le menu apparaissant en cliquant sur le bouton en haut à droite :



et une seconde fois du serveur de machines virtuelles, là où il y a la liste des choix des VMs (dont la IC-CO-IN-SC).

N'oubliez pas non plus que **tout ce qui n'est pas sauvegardé dans le répertoire myfiles sera perdu lors de votre prochaine connection !**

Littérature

[Les raccourcis clavier d'Ubuntu.](#)