Initiation à GNU/Linux
TP en ligne de commande

M1 Modélisation statistique, 2023–2024

Avant toute chose, ouvrir une (ou plusieurs !) fenêtre(s) de “terminal”, ou “console”, en cliquant sur l’icône correspondante (ou raccourci clavier `Ctrl+Alt+T` sous Ubuntu).

Deux conseils *extrêmement* importants dans l’utilisation d’un terminal de manière efficace :

- Ne pas hésiter à utiliser l’**autocomplétion** grâce à la touche `[Tab]`. C’est en particulier utile pour éviter de devoir taper un nom de commande ou de fichier particulièrement long ou compliqué (par exemple un fichier téléchargé contenant un numéro de version à rallonge).
- Après avoir tapé quelques commandes, on peut parcourir l’**historique des commandes** utilisées en utilisant les touches `[↑]` et `[↓]`. Une fois sur une ligne, on peut la modifier en revenant en arrière avec les touches `[←]` et `[→]`, voire `Ctrl+[←]` et `Ctrl+[→]` pour aller plus vite.

1 Les basiques

Sous Linux comme sous Windows ou Mac, l’information est structurée en **arborescence** : des fichiers (texte, images, audio, vidéos, programmes) sont rangés dans des dossiers, eux-mêmes sous-dossiers de leur dossier parent, etc., jusqu’à remonter à la racine du système (désignée par un slash “/” sous Linux).

Vous pouvez, comme sur les autres systèmes, utiliser :

- un navigateur web (Firefox) équivalent à Edge / Safari.
- un client de messagerie (Thunderbird) équivalent à Outlook / Mail.
- un gestionnaire de fichiers (Fichiers ou `nautilus`) équivalent à l’explorateur Windows / Finder.
- un éditeur de fichiers (`gedit`) équivalent au Bloc-Notes / TextEdit.
- etc.

Mais le TP, sauf exception, se déroule **entièrement en ligne de commande, dans une fenêtre de terminal, ou console**. En ligne de commande, on a accès aux mêmes fichiers qu’en utilisant les logiciels décrits ci-dessus, mais leur utilisation se fait par le biais de programmes différents (des programmes en ligne de commande !). De plus, certaines opérations peuvent se faire très simplement dans la console, et on y a souvent recours plutôt qu’à un logiciel spécialisé disposant d’une interface graphique.

Surtout, il est parfois utile d’utiliser la ligne de commande pour **modifier un fichier sur un serveur** (un ordinateur qui n’est pas physiquement là mais sur lequel vous pouvez vous connecter). Par exemple, en entreprise vous pourriez être amené·es à changer la configuration d’un programme qui tourne en permanence sur un serveur. Il peut être plus simple de modifier un fichier de configuration *en ligne de commande et à distance* plutôt que de mettre en place des moyens pour pouvoir modifier les paramètres de la configuration depuis la fenêtre d’un logiciel. Le but de ce TP est de vous donner les bases qui vous permettront de faire cela (et plein d’autres choses).

1.1 Se repérer

Par défaut, la console est une sorte de gestionnaire de fichiers : quand on l’ouvre, on se trouve dans un dossier spécifique, on peut se déplacer, etc. Dans cette section, on utilisera les commandes suivantes :

- `pwd` : *print working directory* (où suis-je ?)
- `cd` : *change directory* (changer de dossier)
- `head` : afficher le début d'un fichier
- `cat` : afficher un fichier en entier
- `ls` : *lister* les fichiers (du dossier courant)
- `man` *`commande`* ou *`commande`* `--help` : *manuel d'utilisation* de la commande, ou aide rapide

Important : consulter le manuel ! Tout au long des TP, ne pas hésiter à recourir au manuel (`man` *`commande`* – naviguer avec les flèches puis taper `q` pour quitter l'interface qui s'affiche) et/ou à l'aide rapide (*`commande`* `--help`) pour plus d'information sur une *`commande`* en particulier et ses différents usages et options.

Exercice 0.

a) Télécharger l'archive du TP, puis, avec le gestionnaire de fichier, la déplacer du dossier "Téléchargement" jusqu'au dossier "Home" (ou `/home/etudiant/`).

b) Ouvrir une fenêtre de terminal. Par défaut on se trouve dans le dossier "Home". Utiliser `$ pwd` pour le vérifier.
NB : j'écris un signe dollar pour signaler une commande à entrer dans la console car c'est le signe qui est normalement affiché dans la console avant la partie ou vous pouvez écrire. Donc il ne pas taper le signe dollar, seulement `pwd`*, puis "Entrée".*

c) Utiliser `$ tar -xf tp.tar.gz`
Ceci extrait le contenu de l'archive dans le dossier courant.
Tout est prêt pour pouvoir vraiment commencer le TP !

Exercice 1.

a) La commande `cd` permet de se déplacer de dossier en dossier. Utiliser la commande
`$ cd "tp/1.1 Se repérer"`

b) La commande `ls` permet d'afficher le contenu d'un dossier. Utiliser `$ ls`

c) Vous devriez voir que le dossier "`1.1 Se repérer`" dans lequel vous êtes contient un fichier `README.md`. Utiliser `$ head README.md` pour afficher ses dix premières lignes, puis suivre les instructions.
NB : on est rarement obligé de tout écrire ! Au lieu de `head README.md`*, il suffit de taper* `head R` *puis d'appuyer sur* `[Tab]`.

Exercice 2. La commande `ls`, comme de nombreuses commandes, permet d'utiliser des *options*. Celles-ci changent le fonctionnement de la commande et sont en général signalées par une lettre suivie d'un tiret, ou par un mot suivi de deux tirets. Par exemple, les deux options de `ls` que nous verront ici sont les suivantes :

- `-l` : Afficher sur plusieurs colonnes des informations (droits d'accès, propriétaires, taille, date de modification) sur les fichiers en questions.
- `-a` ou `--all` : Afficher les fichiers cachés (sur Linux, ce sont simplement les fichiers dont le noms commencent par un point).
- `--help` : Afficher l'aide de la commande. Cette option est disponible sur la majorité des commandes.

Les options peuvent être combinées (quand cela a du sens), et de plus les options à une lettre peuvent être combinées en utilisant le même tiret, par exemple les trois commandes suivantes sont équivalentes :

```
$ ls --all -l
$ ls -a -l
$ ls -al
```

a) Toujours depuis le dossier "`1.1 Se repérer`", tester la commande `ls` avec (et sans !) les différentes combinaisons possible parmi les options suggérées.

   Cette commande peut aussi prendre des arguments, le plus simple étant de donner le nom d'un dossier : sans vous déplacer, afficher le contenu du dossier "`tp`" (dossier parent du dossier dans lequel vous êtes).

b) Revenons au dossier "`1.1 Se repérer`". Vous avez dû voir qu'il contient une archive `coffre.zip`, que l'on veut dézipper : essayer la commande `$ unzip coffre.zip`

   Vous avez trois essais de mot de passe, mais si vous ne le connaissez pas, vous pouvez annuler la commande en tapant sur `Ctrl-C`. *Cette combinaison de touche arrête l'exécution de n'importe quel programme qui vient d'être lancé depuis la console et qui n'est pas terminé.*

c) Trouvez le mot de passe (il est présent dans le dossier ou vous êtes...), extraire le contenu de l'archive et lire le fichier extrait.

1.2 Manipuler les fichiers et dossiers

Exercice 3. On utilisera ici :

- `mkdir` : *make directory* (créer un répertoire)
- `touch` : créer un fichier vide, ou signaler la modification d'un fichier existant
- `echo` : Afficher du texte dans la console
- `>`, `>>` redirection de sortie, écrire dans un fichier
- `head`, `tail`, `less` : gérer l'affichage de longs fichiers

Se déplacer dans le dossier "`tp/1.2 Manipuler`" et **suivre les instructions du fichier `README.md`** qui s'y trouve.

Exercice 4.

a) Toujours depuis le dossier "`1.2 Manipuler`", afficher le contenu de `long_texte`. Il est très long et peu pratique à lire... Expérimenter les commandes `head`, `tail` et `less` pour se faciliter la lecture.

   Pour afficher un certain nombre de lignes avec `head` *ou* `tail`*, utiliser l'option* `-n`*. Pour* `less`*, on navigue avec les flèches et on quitte avec* `q` *(c'est un des programmes qu'on ne peut pas arrêter avec* `Ctrl-C`*).*

b) Ouvrir une deuxième console (afficher les deux consoles côte-à-côte) et se déplacer dans le dossier "`1.2 Manipuler`" avec celle-ci. Créer un fichier `temps_reel` avec le contenu que vous voulez, et taper `$ tail -f temps_reel` dans l'un des terminaux. Avec l'autre terminal, rajouter des lignes à la fin de ce fichier (grâce à `echo` et l'opérateur `>>`), pour comprendre ce que fait l'option `-f` de `tail`.

   À votre avis, à quoi ça peut servir ?

1.3 Ceci n'est pas une |

Exercice 5. L'opérateur *pipe* | (anglais pour *tuyau(terie)*) permet de rediriger la sortie d'une commande vers une autre. Exemple en pratique avec la commande `grep`.

a) Se déplacer dans le dossier "`tp/1.3 Recherche`" puis utiliser la commande `cp` pour copier le fichier `long_texte` qui se trouve dans le dossier "`1.2 Manipuler`" jusqu'à votre emplacement.

Indication : on utilise cette commande comme ceci :

- `cp` *`fichier1 dossier`*, ce qui a pour effet de copier `fichier1` tel quel (en gardant le même nom) dans `dossier`,
- `cp` *`fichier1 fichier2`*, (où l'adresse `fichier2` peut ne pas encore exister) ce qui a pour effet de copier `fichier1` à la nouvelle adresse `fichier2`, donc en changeant potentiellement de nom.

Rappel : le nom `.` est un nom de dossier valide.

b) Taper `$ grep Charlie long_texte`. Que fait la commande `grep` ici ? Comparer avec la commande `$ cat long_texte | grep Charlie`.

c) Quelle option de `grep` permet de trouver le numéro de la ligne du fichier `long_texte` contenant le mot `Charlie` ?

d) Par défaut, `grep` est *sensible à la casse* : vérifier que `$ grep charlie long_texte` ne renvoie rien. Quelle est l'option qui permet d'ignorer la casse, c'est à dire quelle est la lettre □ qui fait que `$ grep -□ charlie long_texte` renvoie quelque chose ?

e) Se déplacer dans le dossier `gros_dossier`, qui contient un grand nombre de fichiers. En utilisant `ls` et `grep`, trouver le nom complet du seul fichier dont le nom contient le mot "Charlie".

On pourra vérifier qu'une autre manière de faire et d'écrire `$ ls *Charlie*`. L'astérisque est un caractère spécial qui est interprété comme *n'importe quelle chaîne de caractère*.

f) Suivre les instructions contenues dans le fichier dont le nom contient "Charlie".

1.4 Manipuler les fichiers, épisode 2

Exercice 6. On utilisera ici :

- `cp`, `mv`, `rm` : respectivement copier, déplacer/renommer, supprimer des fichiers

a) Se déplacer dans le dossier "`tp/1.4 Manipuler2`".

b) *(Copier un fichier)* Utiliser la commande `cp` pour copier `a_copier` vers un nouveau fichier `copie`, et vérifier que leur contenus sont identiques.

c) *(Déplacer un fichier)* Créer un nouveau sous-dossier `exo` puis utiliser la commande `mv` pour déplacer le fichier `copie` vers ce nouveau sous-dossier.

La commande `mv` s'utilise comme `cp`, mais elle coupe au lieu de copier.

d) *(Renommer un fichier)* Utiliser encore `mv` pour renommer le fichier `exo/copie` en `renommage`.

NB : on peut déplacer et renommer en même temps.

e) *(Supprimer un fichier)* Utiliser `rm` pour supprimer le fichier `renommage`, puis pour supprimer le dossier `exo`.

Indication : il faut rajouter une option à `rm` pour supprimer un dossier (car on doit faire une récursion dans l'arborescence).

1.5 Créer des liens avec `ln`

Exercice 7. Se déplacer d'abord dans le dossier "`tp/1.5 Liens`". On peut créer un *lien symbolique* (vous connaissez peut-être cela sous le nom de "raccourci") vers un fichier des deux manières suivante :

```
$ ln -sr fichier dossier
$ ln -sr fichier1 fichier2
```

(à nouveau, même genre de comportement que pour `cp`).

a) Utiliser les deux méthodes pour créer deux liens "`miroir/Alice`" et "`miroir/Bob`" vers le même fichier `Alice` du répertoire courant.

b) Avec Nano, modifier `Alice` pour lui rajouter du texte, puis utiliser `$ cat miroir/Bob` pour vérifier que les changements sont bien répercutés.

c) Jusque-là, on a créé des liens *relatifs*. Taper `$ ln -s $(pwd)/Alice Charlie`, et utiliser `$ ls -l` pour voir que `Charlie` est un lien absolu (comparer avec ce que donne `$ ls -l miroir`).
Un lien relatif, par exemple `miroir/Alice`*, perd son sens quand on le déplace. A contrario, on peut déplacer* `Charlie` *dans le dossier* `miroir`*, il gardera le même sens.*

d) Deviner l'usage de `$(...)` en ligne de commande.

e) Aller dans le dossier `terrier_lapin_blanc` et suivre les instructions.

1.6 Exécution (de scripts)

Sous Linux, les scripts écrits dans le langage `bash` se reconnaissent généralement à leur extension en `.sh`. Contrairement à Windows, sous Linux l'extension est seulement une convention destinée à l'utilisateur, elle ne sert pas au système d'exploitation pour repérer le type des fichiers : on pourrait renommer `script.sh` en `script.jpg` ou simplement en `script`, cela resterait un script.

Exercice 8. Aller dans le dossier "`tp/1.6 Scripts`".

a) Afficher le contenu de `script.sh` et deviner ce que devrait afficher l'exécution du script.
La première ligne du fichier indique à l'ordinateur comment exécuter le script (/bin/bash *désigne un programme).*

b) Exécuter le script avec `$ ./script.sh` pour vérifier votre prédiction. Modifier-le avec Nano et ré-exécuter le script quelques fois pour vous refamiliariser avec les commandes vues jusque là.

c) Exécuter et *utiliser* de manière correcte `copie.sh` (regardez le contenu du fichier pour voir que le langage `bash` est assez cryptique...).
En pratique, un script s'utilise comme une commande (on peut lui donner des arguments, rediriger ses sorties avec > et >>, utiliser le pipe |, etc).

d) Exécutez `quiz.sh` et appelez-moi pour me dire où vous vous êtes trompez.

e) En copiant et modifiant le fichier, créer son propre quiz de 3 questions portant sur ce cours, et le rendre sur Moodle.

2 Linux en réseau

Jusqu'ici, on a travaillé sur son propre ordinateur, en local. On va maintenant voir comment se connecter à un serveur en tant qu'utilisateur à distance, pour pouvoir agir sur ce serveur. Alors on ne passe plus par l'interface graphique du tout et l'on utilise uniquement la ligne de commande.

N.B. : Vous disposez d'un accès à un compte *administrateur* sur un serveur "bac à sable" qui se trouve *chez moi* (donc inaccessible physiquement...). Ainsi pour le bon déroulement du TP, veuillez ne pas supprimer de fichiers système (c'est-à-dire quoi que ce soit hors du répertoire personnel) ou utiliser des commandes que vous ne connaissez pas (et qui ne sont pas demandées).

2.1 Connexion SSH

SSH – *Secure SHell* – est un *protocole réseau* qui permet des connections sécurisées entre différents ordinateurs d'un réseau.

Nous allons voir ici :

- `ssh`, la commande permettant une connexion SSH.
- `scp`, pour la copie de fichiers au travers d'une connexion SSH.
- `nano`, un éditeur de texte basique en ligne de commande.
- `rsync`, pour des transferts de fichiers plus intelligents.
- L'utilisation du fichier `~/.ssh/config` pour se simplifier la vie.

Exercice 9. Il existe un compte `admin`, associé au mot de passe `mdpadmin` (← ne pas utiliser ce genre de mot de passe dans la vraie vie), sur un serveur dont l'adresse IP est le `77.132.241.23`.

a) Taper `$ ssh admin@77.132.241.23`.
Vous devriez voir apparaître :
```
The authenticity of host '77.132.241.23 (77.132.241.23)' can't be established.
ECDSA key fingerprint is SHA256:kFStLpNcBYbxO3saDzTdnpQVyG5NLc6eUKBPRmTAvJw.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```
Confirmer en entrant `yes`, puis entrer le mot de passe donné plus haut (rien n'apparaît lors de l'écriture du mot de passe, c'est normal).

b) Remarquer que les informations à gauche du signe $ ont changé. Créer un fichier vide, d'un nom choisi au hasard. Taper `$ exit` pour se déconnecter, puis vérifier que le fichier n'apparaît pas dans le répertoire de travail.

c) Se reconnecter au serveur, puis afficher le contenu du dossier courant. Que remarque-t-on ?
Le système Linux permet la connexion simultanée de plusieurs utilisateurs, même s'ils et elles utilisent le même compte.

d) On pourrait utiliser les méthodes vues dans les premières séances pour rajouter un contenu quelconque au fichier créé à la question b), mais utiliser plutôt `nano`, qui fonctionne comme un éditeur de texte "classique" (sauf pour les raccourcis claviers).
À partir de maintenant, toutes les éditions de fichiers se feront en ligne de commande.

e) Se déconnecter puis tâcher d'utiliser `scp` pour récupérer le fichier en question sur la machine locale et vérifier son contenu.

f) Pour des transferts basiques, `scp` convient, sinon on préférera `rsync`, qui permet par exemple de ne transférer que les fichiers modifiés grâce à l'option `-u`. Rédiger un script `backup.sh` qui permette de transférer tous les fichiers du dossier personnel (et de ses sous-dossiers) vers un dossier `~/backup/` du serveur (ne pas chercher à l'exécuter).

g) On en a marre de taper l'adresse IP du serveur. Pour remédier à ça, sur la machine locale créer un fichier `~/.ssh/config` avec le contenu suivant (*attention à respecter l'indentation*) :

```
Host serveur
    User admin
    Hostname 77.132.241.23
```

Vérifier que `$ ssh serveur` fonctionne (ce sera similaire pour l'utilisation de `scp` et `rsync`).

2.2 Comptes utilisateurs et super-utilisateurs

Exercice 10. On se connecte au serveur pour commencer.

a) `admin` est un compte super-utilisateur (ou administrateur), ce qui veut dire que l'on peut faire quasiment tout sur l'ordinateur. Vérifier d'abord qu'afficher le contenu du fichier `/etc/shadow` est interdit.

b) La commande `sudo` permet justement à un super-utilisateur de faire des choses interdites : taper `$ sudo cat /etc/shadow`.
Ce fichier contient les mots de passe des comptes utilisateurs, mais ils sont de toute façon chiffrés...

c) Créer un compte utilisateur avec un identifiant choisi, grâce à `$ adduser` *`login`* (remplacer *`login`* par son choix d'identifiant).

d) Changer d'utilisateur en tapant `$ su` *`login`*, et vérifier que ce nouveau compte n'est pas super-utilisateur. Utiliser `exit` pour "redevenir" l'utilisateur précédent (c'est-à-dire `admin`).
La commande `su` *fonctionne similairement à* `ssh`*, c'est-à-dire qu'on ne change pas véritablement de compte (ce qui impliquerait une déconnexion du premier compte), mais on crée en quelque sorte une console subordonnée à la première — si on la quitte, on revient à la console parente, on l'on reste connecté.*

e) Les super-utilisateurs forment simplement un groupe d'utilisateurs appelé `sudo`. Lister les groupes existant en affichant le fichier `/etc/group` et vérifier qui est présent dans le groupe `sudo`.

f) Toujours grâce à `adduser`, rajouter *`login`* au groupe `sudo`, vérifier que *`login`* est bien super-utilisateur

g) Taper `$ deluser` *`login`* `sudo` pour supprimer les droits administrateurs de *`login`*. **Attention !** Ne pas taper autre chose, le groupe `sudo` *ne doit pas être supprimé.*

h) Créer un groupe `m1stat` *(s'il existe déjà, vous n'êtes juste pas le·a plus rapide, c'est pas grave).*, et rajouter *`login`* à ce groupe.

2.3 Connexion SSH rapide

On va maintenant laisser tomber le compte `admin` et utiliser le compte *`login`* nouvellement créé.

Exercice 11.

a) Se déconnecter du serveur. Modifier le fichier `~/.ssh/config` pour pouvoir se connecter automatiquement au serveur avec le compte *`login`*.

b) Ce qui nous embête maintenant, c'est de taper son mot de passe à chaque fois pour se connecter au serveur. On va utiliser un système de clés publiques/privées pour se connecter à travers SSH sans utiliser de mot de passe.
Taper `$ ssh-keygen -t ecdsa` pour générer une clé aléatoire et cryptée sur l'ordinateur local : garder la localisation du fichier par défaut, ne pas mettre de mot de passe.

c) La clé privée (ne pas la diffuser !) est dans `~/.ssh/id_ecdsa`, est la clé publique est `~/.ssh/id_ecdsa.pub`. Cette clé publique fonctionne en fait comme une serrure pour SSH, avec la clé privée comme véritable clé. Maintenant il s'agit d'installer la serrure sur le serveur :

Ajouter le contenu de la clé publique (une seule ligne) à la fin du fichier (qui n'existe peut-être pas) `/home/login/.ssh/authorized_keys` sur le serveur.

d) Vérifier que `$ ssh serveur` permet maintenant de se connecter au serveur très rapidement.

e) Défi : essayer d'apparaître sur la page du site 77.132.241.23.

3 Droits et propriétés

3.1 Lecture, écriture, exécution

Sous Linux, il y a trois droits d'accès différents aux fichiers de l'ordinateur : droit de lecture, d'écriture et d'exécution. Cet ordre (lecture, écriture, exécution) est à retenir car une combinaison de droits s'exprime en binaire, par exemple :

- `110`, soit l'entier 6 pour des droits de lecture et écriture.
- `100`, soit l'entier 4 pour un droit "lecture seule".
- `101`, soit l'entier 5 pour des droits de lecture et d'exécution.
- `111`, soit l'entier 7 pour tous les droits.

De plus (cela serait trop simple), chaque fichier est la propriété d'un utilisateur et d'un groupe de l'ordinateur. Par exemple, un fichier créé avec touch ou à la suite d'une commande `echo "..." > fichier` sera la propriété de l'utilisateur qui lance la commande et de son groupe principal (en général du même nom que l'utilisateur). Trois catégories d'utilisateurs peuvent alors accéder différemment au fichier :

- l'utilisateur propriétaire du fichier,
- les autres membres du groupe propriétaire,
- et les autres utilisateurs du système.

Cet ordre est à nouveau important car chaque fichier contient l'information des droits d'accès de chacune de ces catégories : on lit cette information en utilisant `ls -l`. Exemple :

```
$ ls -l
total 4
-rwxr-xr-- 1 etudiant etudiant    0 Sep  3 15:24 bar
-r-x-w--w- 1 etudiant etudiant    0 Sep  3 15:24 foo
drwxr-xr-x 2 etudiant etudiant 4096 Sep  3 15:25 zzz
```

Ici, c'est plus simple qu'annoncé plus haut (mais ça va se corser) : au lieu d'afficher `101`, on lit `r-x`, les 0 deviennent '-' et les 1 deviennent la première (ou deuxième) lettre du droit qu'il symbolise : **r**ead, **w**rite, e**x**ecute. De plus il y a un caractère tout à gauche qui nous dit si le fichier est en fait un dossier. Dans l'exemple ci-dessus, on a donc

- Un fichier `bar` pour lequel l'utilisateur a tous les droits (`rwx` donc 7), le groupe a les droits de lecture et d'exécution (`r-x` donc 5) et les autres, seulement le droit de lecture (`r--` donc 4).
- Un fichier `foo` pour lequel l'utilisateur a les droits `r-x` donc 5, et le groupe et les autres ont le droit `-w-` donc 2.
- Un dossier `zzz` pour lequel l'utilisateur a les droits `rwx` donc 7, et le groupe et les autres ont les droits `r-x` donc 5.

Exercice 12. Il y a plusieurs façons d'utiliser `chmod` pour modifier les droits d'un fichier, mais voyons la plus efficace : on détermine les droits que chaque catégorie doit avoir pour le fichier `foo` sous forme de chiffre, par exemple 7, 5 et 5, puis on tape `$ chmod 755 foo`.

a) Dans le répertoire personnel, créer un dossier et faire en sorte d'y recréer la situation ci-dessus.

b) Tester les effets des différents droits de l'utilisateur sur la manipulation de fichiers.

Une autre utilisation pratique de `chmod` est le changement rapide du statut exécutable d'un fichier (pratique pour les scripts). En quelques exemples, on procède comme cela :

- `chmod +x script.sh` pour rendre un script exécutable.
- `chmod -x script.sh` pour enlever le droit d'exécution d'un script.
- `chmod u+x script.sh` pour rendre un script exécutable seulement pour l'utilisateur propriétaire.
- etc.

3.2 Propriété

Exercice 13. Pour cet exercice, on se connecte au serveur avec le compte créé dans la partie 2 du TP. On vérifiera l'existence d'un dossier `/zone_commune` à la racine de l'ordinateur dont les propriétaires sont l'utilisateur `duchamps` et le groupe `m1stat`. *Si ce n'est pas le cas, me prévenir*.

a) Créer un fichier `message_secret_pour_`*`login`* dans son répertoire personnel, ou *`login`* est l'identifiant de son ou sa voisin·e, puis écrire un message à sa destination.

b) Retirer tous les droits de ce fichier à qui que ce soit d'autre que l'utilisateur propriétaire, puis le déplacer dans le dossier `/zone_commune`.

c) Utiliser `chown` pour transférer la propriété de ce fichier à votre voisin·e.
Les droits administrateurs sont nécessaires pour cela donc on empruntera brièvement l'identité d'`admin` *grâce à la commande* `su` *Pour information, la commande* `chgrp` *permet de changer seulement le groupe propriétaire d'un fichier*.

d) Sous son propre compte, déplacer son message dans son dossier personnel puis le lire.

e) Défi : mettre en place un système de chat plus pratique (c'est-à-dire où la communication est instantanée) avec le reste de la salle.
Me demander pour un exemple.