

TP 6 (Révisions)

=====

Le but de ce TP est de réviser les notions principales pour voir si vous les maîtrisez. Donc si un moment vous ne comprenez pas quelque chose ou que vous n'arrivez pas à répondre à une question, demandez-moi ou faites-vous aider par un autre étudiant, internet peut également vous aider. C'est la dernière séance, profitez-en.

Si vous voulez vérifier que ce que vous faites dans le terminal entraine bien une modification des dossier ou des fichiers, allez voir graphiquement les dossiers et les fichiers, comme vous aviez l'habitude de faire avant d'utiliser un terminal.

Préliminaire

Dès que vous écrivez des commandes, testez la touche TAB pour voir quand est-ce qu'elle peut être utile. Parfois il est nécessaire de la presser deux fois à la suite.

Lorsque vous ouvrez le manuel d'une commande, pour pouvoir faire une recherche d'un mot, pressez la touche / puis tapez votre mot, puis appuyez sur ENTRER. Pour vous déplacer, d'une occurrence du mot recherché à une autre, utiliser les touches n et SHIFT+n. C'est utile notamment quand vous cherchez des options. Par exemple si vous chercher comment inverser l'ordre du tri que propose la commande ls lorsqu'elle liste les fichiers d'une dossier, il suffit d'aller dans le manuel de ls et de rechercher le mot tri.

Chemin absolu, chemin relatif

Sous Linux, on a vu que tous les dossiers et les fichiers sont dans le répertoire racine qu'on note /. Ainsi, pour tout dossier ou fichier, il existe un chemin depuis la racine qui représente la suite des dossiers qu'on doit parcourir pour y arriver.

Par exemple, si on considère l'organisation suivante : (Pour différencier les dossiers des fichiers, un / a été mis à la fin des dossiers.)

```
/
|-- bin/
|   |-- bzless
|   |-- udevadm
|-- tmp/
|-- net/
|   |-- cremi/
|       |-- utilisateur/
|           |-- Unix/
|               |-- toto
|               |-- titi
|               |-- script/
|                   |-- hello.sh
|           |-- TPs/
|               |-- TP1
|               |-- TP2
|               |-- TP5/
|                   |-- capitales.txt
```

(Dans les prochaines questions on utilisera ce diagramme, si besoin n'hésitez pas à le recopier pour vous faciliter la vie.)

6.1) Généralement on parle d'organisation en arborescence, voyez-vous pourquoi ?

Le chemin absolu du fichier udevadm est

```
/bin/udevadm
```

Celui du dossier utilisateur c'est

```
/net/cremi/utilisateur
```

On dit que le chemin est absolu parce qu'il ne dépend pas de où vous vous trouvez dans l'arborescence.

6.2) Donnez le chemin absolu des dossiers tmp et TPs, et des fichiers toto et titi.

Pour afficher le chemin absolu du répertoire dans lequel vous vous trouvez se fait à l'aide de la commande pwd. On peut aussi lire cette information directement. L'endroit où vous tapez vos commandes est de la forme

```
utilisateur@PC:chemin_absolu_dossier_courant:
```

La deuxième type de chemin est le chemin relatif. On dit relatif car il va dépendre de où vous vous trouvez dans l'ordinateur. Pour pouvoir écrire ce chemin, on a besoin de deux choses

- . correspond au répertoire courant (le répertoire où vous vous trouvez)
- .. correspond au répertoire qui contient le répertoire courant

Le chemin relatif d'un dossier ou d'un fichier va décrire la suite des dossiers qu'il faut parcourir, depuis le dossier courant, pour y arriver. Par exemple, supposons que vous vous trouviez dans le dossier utilisateur, le chemin relatif du fichier bzless est

```
../../bin/bzless
```

Les .. servent à remonter d'un cran, comme on se trouve dans utilisateurs ../ correspond à cremi, ../../ à net et ../../.. à /. Supposons qu'on soit toujours dans le répertoire utilisateur, le chemin relatif jusqu'au fichier toto est

```
./Unix/toto
```

Le . sert à dire qu'on part du répertoire courant, en effet, ici . correspond à

```
/net/cremi/utilisateur
```

6.3) Supposons que vous êtes dans Unix. Donnez les chemins relatifs jusqu'à TP1, capitales.txt et hello.sh.

Si vous avez besoin du chemin d'un dossier ou d'un fichier à l'autre bout de l'arborescence les chemins absolus sont mieux. Mais s'il est assez proche, le chemin relatif est une meilleure solution.

C'est grâce à ces chemins que vous pourrez expliquer à votre terminal ce que vous voulez faire quand vous écrirez des commandes.

Quand êtes en train d'écrire un chemin relatif ou absolu, n'oubliez pas d'essayer d'utiliser la touche TAB pour profiter de l'autocomplétion.

Gestion de l'arborescence

La commande pour créer un dossier est mkdir (make directory).

Si on reprend l'arborescence de tout à l'heure, quelque soit l'endroit où je suis je peux créer le dossier TP4 dans le dossier TPs en écrivant la commande

```
mkdir /net/cremi/utilisateur/TPs/TP4
```

Si on se trouve dans le dossier utilisateur,
on pourrait écrire

```
mkdir ./TPs/TP4    ou    mkdir TPs/TP4
```

Si on se trouve dans le dossier TPs, il suffit d'écrire

```
mkdir TP4
```

Quand on utilise le chemin relatif, on peut ne pas mettre
"./" au début du chemin.

6.3,5) Peut-on supprimer le dossier script à l'aide de la
commande rmdir ? Si on le pouvait et qu'on se trouvait
dans le dossier cremi, que faudrait-il écrire.

6.3,7) Pour supprimer un dossier non vide il faut utiliser la commande
rm muni d'une certaine option, quelle est cette option ?

6.4) Si vous êtes dans le répertoire Unix que devez vous taper
pour créer le répertoire sauvegarde dans le dossier utilisateur.
Donnez la réponse en utilisant les deux types de chemins.
Faites de même pour créer le dossier test dans le dossier script
en supposant que vous êtes dans le dossier utilisateur.

La commande cd marche de la même manière. Essayez de vous déplacer
dans votre arborescence.

Pour afficher l'arborescence d'un dossier vous pouvez utiliser
la commande tree. Regardez le manuel pour répondre à la question
suivante.

6.5) Affichez l'arborescence de votre dossier personnel en n'affichant
que les dossiers et avec une profondeur de deux. Si vous ne comprenez
pas la notion de profondeur, faites des tests en changeant la valeur.

6.6) Listez les fichiers du dossier bin en utilisant un chemin absolu.

6.7) Depuis le dossier /bin, listez les fichiers, en format long, de votre
dossier personnel, en utilisant un chemin relatif. Il ne faut pas que
les groupes apparaissent.

6.8) Depuis la racine, listez les fichiers normaux et les fichiers cachés
de votre répertoire utilisateur. Les fichiers cachés, sont ceux qui commencent par .

La commande qui sert à copier un fichier ou un dossier est cp.

6.9) Quelle option faut-il rajouter pour pouvoir copier un dossier ?

Voici quelques exemple pour comprendre comment cp marche.
Supposons qu'on soit dans le répertoire utilisateur,

```
cp Unix/toto TPs/TP5 : on copie le fichier toto dans le répertoire TP5,  
                      on garde le nom toto.
```

```
cp Unix/toto TPs/coucou : on copie le fichier toto dans le répertoire TPs,  
                        on lui donne le nom coucou.
```

```
cp Unix/toto TPs/TP2 : on copie le fichier toto dans le répertoire TPs  
                      en lui donnant le nom TP2, donc on écrase le  
                      fichier TP2 qui existait.
```

6.10) Quelle option de cp permet d'avoir d'être prévenu si on va écraser un fichier ?

La commande mv marche de la même manière que cp. Elle permet de déplacer/renommer
les fichiers et les dossiers. Il n'y a pas besoin d'option pour l'appliquer
à des dossiers.

6.11) Donnez plusieurs commandes pour visualiser le contenu d'un fichier.

Pour la troisième réponse de la correction, on peut faire une recherche, de la même manière que dans le manuel.

6.12) Quel commande permet de modifier les permissions d'un fichier ou d'un dossier.

Pour pouvoir visualiser les permissions d'un fichier/dossier, on peut utiliser la commande `ls -l`. Par exemple si on veut juste les permissions du fichier toto et pas celles de titi ou du dossier script, on peut taper la commande

```
ls -l toto
```

Le problème si on fait ça avec un dossier c'est qu'il va lister le contenu du dossier. Pour éviter ce problème, on peut rajouter l'option `-d` qui empêche le listage des dossiers.

Dans le TP1 on a vu qu'on pouvait affiner les recherches en utilisant les caractères `* ? [ ]`. Si vous voulez plus d'informations à ce sujet chercher "Glob bash" sur internet.

Variables, historique, alias

6.13) Créez la variable `var1` contenant la valeur toto et titi, sans utiliser la commande `read`. Affichez la valeur de cette variable.

6.14) Créez la commande `var2` contenant la valeur plage en utilisant la commande `read`.

6.15) En utilisant `var1` et `var2`, affichez à l'écran

```
toto et titi vont à la plage
```

Affichez maintenant

```
$var1 vont à la $var2
```

6.16) Quelle commande faut-il utiliser pour supprimer une variable ?

6.17) Comment affiche-t-on l'historique des commandes ?

On peut remonter l'historique avec la flèche du haut, ça évite de tout retaper si on a fait une erreur par exemple.

6.18) Comment fait-on pour créer un alias ?
Pour le supprimer ?

Si vous trouvez des alias intéressants que vous voulez garder sans avoir à les redéfinir à chaque fois que vous démarrez votre PC, vous pouvez l'écrire dans le fichier `.bashrc` qui se trouve dans votre répertoire utilisateur.

Si vous voulez plus d'information sur comment utiliser ce fichier, chercher "configurer .bashrc" sur internet.

grep, find

grep permet de rechercher des chaînes de caractères dans les fichiers. La syntaxe basique est la suivante

```
grep "chaîne à chercher" liste_de_fichier
```

Pour créer une liste de fichiers, on peut utiliser les caractères `* ?` et `[]`, comme dans le TP2. Si vous voulez faire des tests, créez des fichiers et avec `echo` regardez ce que vous obtenez. Par exemple

```
echo *az*[a-e]???
```

La commande

```
grep "maison" toto/*
```

va afficher les lignes des fichiers du répertoire toto qui contiennent maison.

```
grep "coucou" ?[a-e]*
```

va afficher les lignes contenant le mot coucou des fichiers, qui commence par un lettre quelconque puis une lettre entre a et e et qui finissent comme on veut.

Pour ne regarder que les fichiers et pas les dossiers, on rajoute l'option "-d skip".

On peut rajouter l'option -l pour n'afficher que la liste des fichiers qui contiennent la chaine qu'on a cherche. Ça peut être pratique pour créer une liste de fichiers qu'on va envoyer à une autre commande par exemple.

Typiquement, si on fait

```
cat $(grep -l -d skip "maison" *)
```

Cette commande affiche le contenu de tous les fichiers du répertoire courant qui contiennent le mot maison. \$() permet d'exécuter d'abord ce qu'il y a à l'intérieur. C'est ce qui permet dans un premier temps de créer la liste des fichiers qui nous intéresse, avant de la donner à cat pour les afficher.

Si au lieu de chercher une chaine de caractère simple, on veut chercher un motif, c'est à dire qu'on va chercher tous les mots qui respectent le motif qu'on a mis, on peut utiliser les expressions régulières. C'est dans la même veine que l'utilisation de * ? [], mais beaucoup plus complet. C'est l'option -e qui permet d'utiliser les motifs avec grep. On peut également utiliser les expressions régulières dans les éditeurs de textes comme emacs et vim par exemple pour chercher des mots dans un fichier.

On peut également utiliser les expressions régulières avec la commande find pour chercher des fichiers de manière plus polyvalente.

Archivage/Compression

À partir du terminal il est possible d'archiver des fichiers et des dossiers c'est à dire qu'on va créer un fichier qui va contenir des fichiers et des dossiers qu'on aura sélectionné. Un intérêt de faire ça, c'est qu'après, on peut compresser ce fichier (on ne peut pas compresser un dossier).

Pour pouvoir archiver un fichier ou un dossier la syntaxe est la suivante

```
tar -cvf archive.tar fichier      ou      tar -cvf archive.tar dossier
```

-cvf correspond au fait qu'on a utilisé les options c, v et f.
L'option c correspond au fait qu'on veut créer une archive,
v qu'on veut que soit affichée la liste des fichiers qui sont archivés et f qu'on utilise le fichier ou le dossier donné en argument.
Attention à l'ordre des lettres !!
L'archive sera créée s'appellera archive.tar et sera dans le répertoire courant car on a pas précisé de chemin relatif ou absolu.

Si on veut archiver plusieurs fichiers, on met d'abord tous ces fichiers dans un même dossier et après on utilise la commande décrite au dessus.

6.19) Proposez une commande pour archiver le dossier Unix, depuis le dossier cremi sachant que l'archive doit être créée dans le dossier /tmp et doit s'appeler archive.

Pour extraire une archive il faut utiliser l'option x, la syntaxe est la suivante

```
tar -xvf archive
```

L'archive sera extraite dans le dossier courant.

6.20) Si on veut choisir dans quel dossier on veut extraire l'archive il faut utiliser une option. Trouvez-là dans le manuel de tar.
Proposez une commande pour extraire, depuis le répertoire utilisateur l'archive archive qu'on vient de créer dans le dossier /tmp, dans

le répertoire TPs.

Pour compresser un fichier, on peut utiliser la commande gzip.

```
gzip fichier
```

Pour décompresser un fichier c'est la commande gunzip.

On peut archiver et compresser un dossier avec une seule commande en ajoutant l'option z à tar.

```
tar -zcvf archive.tar.gz dossier
```

De même pour décompresser et extraire l'archive d'un coup on écrit

```
tar -zxvf archive.tar.gz
```

Faites des tests pour voir si vous avez bien compris, créez un dossier avec des fichiers et des sous-dossiers puis essayer de l'archiver de le compresser et de revenir en arrière.

Gérer processus

6.22) Lancez xeyes, suspendez-le et mettez le en fond de tâche.

6.23) Remettez-le au premier plan. Puis arrêtez-le avec une combinaison de touches.

6.24) Lancez xeyes, tuez-le avec son numéro de job.

6.25) Lances xeyes, tuez-le avec son PID.

< , > , 2> , Pipe

< permet de donner un fichier à un processus. Par exemple lorsque vous faite la commande

```
read x
```

le terminal attend que vous lui donnez une chaîne de caractère pour l'assigner à x. Au lieu de taper une chaîne de caractère vous pouvez lui dire d'aller chercher l'information dans un fichier en utilisant la commande

```
read x < fichier
```

< peut également être utilisé d'une autre manière, si une commande demande un fichier en argument, au lieu de lui écrire directement le fichier, vous pouvez le lui donner à l'aide <. Par exemple, au lieu d'écrire

```
wc -w fichier
```

vous pouvez écrire

```
wc -w < fichier
```

Parfois la forme du résultat peut changer comme c'est le cas ici.

> permet d'écrire dans un fichier ce que le programme aurait du afficher. Par exemple

```
ls > liste
```

Crée le fichier liste dans lequel il y aura la liste des fichiers courants. En particulier, si vous voulez créer un fichier juste une phrase dedans, vous pouvez faire comme suit

```
echo "Aujourd'hui il fait gris" > meteo
```

Un fichier meteo a été créé et dedans il y aura la phrase Aujourd'hui il fait gris.

6.25,5) Créez des fichiers contenant des phrases de votre choix, dont certains seulement possèdent le mot toto.
En une seule commande et en utilisant grep, créez un fichier contenant la liste des fichiers qui contiennent le mot toto.

Lorsqu'on écrit `cmd1 | cmd2`, au lieu que la `cmd1` affiche à l'écran son résultat, elle va envoyer son résultat à `cmd2` sous forme de fichier et `cmd2` va être exécuté comme si on l'avait appelé avec

```
cmd2 fichier
```

où fichier correspond la sortie de `cmd1`. C'est comme si on avait fait

```
cmd1 > toto ; cmd2 toto
```

Sauf que là on a créé un fichier `toto` pour rien. Par exemple, si on crée un fichier `la_vie_de_toto_et_titi` dont les lignes sont

```
toto va à la plage
titi aime le foot
toto va à la piscine
toto mange une glace
titi aime bien sa nouvelle cage
```

et qu'on fait

```
grep "toto" la_vie_de_toto_et_titi
```

on obtient

```
toto va à la plage
toto va à la piscine
toto mange une glace
```

Si on veut afficher les lignes en partant du bas, on peut faire (tac permet de lire un fichier à l'envers)

```
grep "toto" la_vie_de_toto_et_titi | tac
```

Si en plus on veut le sauvegarder dans un fichier `la_vie_de_toto` on peut faire

```
grep "toto" la_vie_de_toto_et_titi | tac > la_vie_de_toto
```

6.26) À l'aide de la commande `ps -aux` puis de la commande `grep`, afficher tous les processus liés à votre nom d'utilisateur. Vous ne devez utiliser qu'un pipe.

6.26) En utilisant en plus `wc -l` et un deuxième pipe, comptez le nombre de processus qui ont votre nom.

6.27) En enregistré ce nombre dans un fichier `nombre_de_processus` en utilisant `>`.

--

Patxi Laborde Zubieta (patxi.laborde-zubieta@labri.fr)