Introduction :
Initiation aux commandes Unix / Linux

Qu'est-ce que Unix, pourquoi faire de l'Unix en terminal (console) ?

Un système Unix est un système d'exploitation répondant à un **standard** garantissant un ensemble de fonctionnalités utilisables de manière normalisée. Par exemple l'ensemble des **commandes** étudiées dans ce TP sur Linux Ubuntu restent valables sur le système Unix BSD (à la base de OS X) ou sur le système Unix Solaris etc...

Cette séance est un 1er contact avec les commandes et schémas usuels d'une utilisation d'Unix en **ligne de commande** : nous envoyons des ordres et recevons des réponses en mode textuel dans une console ou **terminal** selon le vocabulaire Unix.

Que vous utilisiez les machines école avec connexion à distance sur Linux, ou que vous utilisiez un Linux installé sur machine perso, vous serez sur un Bureau avec icônes, lanceur d'applications, navigateur de fichier etc... tout le confort moderne. C'est pratique mais ce qui nous intéresse sur ce sujet ce sont **les commandes en terminal, que vous trouverez sur tout système Unix.**

Lancer votre session Linux et ouvrir un terminal

Si vous avez un système MacOS, vous avez un Unix ! Les manips décrites dans ce sujet sont en principe valables. Vous trouverez le terminal dans le dossier Utilitaires situé dans Applications.

Sur machine école ou perso sous Windows :
voir « Connexion à Linux à distance depuis machine Windows » sur campus (Initiation Linux).

En haut à gauche vous accédez au menu Applications, dérouler → Accessoires → Terminal

Comme nous allons manœuvrer dans le **système de fichiers** (arborescence de répertoires) à partir du terminal, il peut être utile de vérifier ce qui se passe depuis une interface plus intuitive :
menu Raccourcis en haut du bureau → Poste de travail.

Le terminal et l'interpréteur de commande : shell

Le terminal est une fenêtre textuelle où l'utilisateur entre les commandes à faire exécuter par le système. Une ligne de commande commence par le nom de la commande et est éventuellement suivie par des options ou paramètres. La commande est validée (l'action est déclenchée) par retour ligne (touche Entrée). Les lignes de commande ne sont pas directement transmises du terminal au système mais passent par l'interpréteur de commande ou ***shell*** qui est le véritable interlocuteur, le terminal n'étant qu'une boite de saisie et d'affichage de texte. Le shell vérifie la cohérence syntaxique de la ligne, gère le contexte des commandes (répertoire courant où a lieu l'action...) et lance effectivement l'exécutable associé à la commande.

Dans le terminal vous devez voir une invite de commande ou ***prompt*** signalant que le shell est prêt à recevoir vos ordres. Les commandes s'écrivent directement après ce prompt. Vous validerez les **`commandes suivantes`** en vérifiant que le résultat décrit correspond bien à ce qui est indiqué.

1/ Qui suis-je ?

`whoami`

Indique votre identifiant d'utilisateur (logging)

2/ Où suis-je ? J'habite où ? Bonjour ? Il y a quelqu'un ?

`pwd`

Print Working Directory : affichage du répertoire de travail, dit aussi "répertoire courant"
Le répertoire courant est par défaut celui où les actions sur les fichiers sont lancées.
Essayez de retrouver ce répertoire courant dans l'explorateur de fichiers (poste de travail)
Sous Unix un répertoire (ou un fichier) est désigné par un **chemin (path)** qui est la séquence des répertoires qu'il faut traverser depuis la racine / pour y accéder en descendant l'arborescence.

`echo $HOME`

Affichage du chemin du répertoire personnel. Au démarrage d'un nouveau terminal, ce répertoire est le répertoire courant. La commande echo demande un affichage immédiat des paramètres. Les noms commençant par $ sont des variables du système, qui sont des chaînes, on demande leur valeur.

`echo bonjour le monde`

`echo je m'appelle $USER et j'utilise le shell $SHELL`

Cette dernière commande semble ne pas marcher : les guillemets simples utilisés en apostrophes encadrent **'appelle $USER et j'** comme une chaîne à utiliser telle quelle (*strong quoting*).
On peut adopter une "quotation" plus faible pour garder la substitution des variables par leur contenu mais échapper le guillemets simples :

`echo "je m'appelle $USER et j'utilise le shell $SHELL"`

Le shell que nous utilisons est le BASH, un shell répandu mais pas le seul à exister.

`ls`

Lister le contenu du répertoire courant

`ls -l`

(**`l`** comme "long") Lister le contenu du répertoire courant, option longue = infos détaillées
La colonne de gauche indique pour chaque fichier ou dossier :
d pour "directory" → dossier
r pour "read" droit en lecture w pour "write" droit en écriture x pour "eXecute" droit en exécution
Le 1er bloc rwx concerne le propriétaire,
Le 2ème concerne le groupe (ici nous n'entrons pas dans les détails)
Le 3ème concerne tout le monde
exemple : **`-rw-r--r--`** Fichier (pas de d) lecture pour tous, écriture propriétaire, pas d'exécution

`ls -a`

Lister le contenu du répertoire courant, option tous (*all*) = afficher les fichiers cachés
Les fichiers ou répertoires commençant par . sont des fichiers de configuration, cachés par défaut.
Les répertoires . et .. sont des répertoires spéciaux qui représentent :
. répertoire courant .. répertoire parent (au dessus du répertoire courant dans l'arborescence)

2.5/ Qu'est-ce que … ?

whatis ls
whatis whatis

La commande whatis donne un bref descriptif du rôle joué par la commande donnée en paramètre

man ls

Pour un descriptif plus complet on utilise le **man**uel, qui explique en détail toutes les options d'une commande. Vous pouvez retrouver -a et -l. Inutile de s'attarder (on ne va pas apprendre tout ça!) appuyer sur q pour quitter le manuel et revenir au shell.
Essayer **man man** (manuel du manuel) et jeter un œil à la 1ère page : on comprend pourquoi Unix en ligne de commande peut parfois impressionner les âmes non technophiles...

which ls
which which

La commande which indique l'emplacement (le chemin) de la commande donnée en paramètre.

3/ Où vais-je ?

Vous allez naviguer dans le système de fichier, ajouter des répertoires, créer/voir des fichiers...

ls ..
pwd

Affiche le contenu du répertoire parent du répertoire courant. Le répertoire courant est inchangé.

ls .. -l -a
pwd

Affiche la totalité du répertoire parent avec infos détaillées (on peut écrire -la au lieu de -l -a)
Le répertoire courant est inchangé.

cd ..
pwd
ls

cd = Change Directory : Change le répertoire courant. Ici on atterrit dans le répertoire parent.

cd /
ls -la

Aller à la racine du système de fichier et voir en détail tout ce qui s'y trouve.

cd bin
ls

Aller dans le répertoire des commandes exécutables bin (*binaries*) et voir les commandes disponibles. Il y en a davantage à d'autres emplacements, voir par exemple dans le répertoire /usr/bin : **ls /usr/bin** (la commande which permet de savoir où se trouve une commande)

cd ~
pwd

Retour au répertoire HOME : le ~ représente votre répertoire HOME
Pour voir ce qui se passait dans /bin on aurait pu rester à la maison et faire **ls /bin** directement : on ne se "déplace" par changement du répertoire courant avec **cd** que si on a beaucoup de choses à faire dans un certain répertoire, sinon on regarde "à distance" en précisant le chemin en paramètre.

A noter que les chemins qui commencent par / sont des chemins absolus (on part de la racine) alors que ceux qui ne commencent pas par / sont des chemins relatifs (on part du courant)
Par exemple **`/tmp`** se réfère au répertoire tmp à la racine du système
tandis que **`tmp`** se réfère à un répertoire tmp supposé être dans le répertoire courant.

4/ Complétion et Navigation dans l'historique des commandes

Il est possible de retrouver (et relancer, ou éditer et relancer) une commande précédente en utilisant les **flèches haut et bas** : essayer de lancer **`echo Rebonjour le monde`** en retrouvant la commande initiale et en ne modifiant que le "Re"

`history`

cette commande affiche un historique numéroté de vos commandes.
`history 10` pour avoir seulement les 10 dernières...

Vous pouvez relancer une commande en tapant ! immédiatement suivi du numéro de commande de l'historique. Essayez avec une des commande de listing (ls) déjà entrée...

Quand on tape une commande qui nécessite d'indiquer un chemin un peu long il est possible de demander la **complétion de nom** en appuyant sur la touche Tab :
Nous allons voir (sans déplacements avec cd) les fichiers en-têtes standards du C à notre disposition dans le répertoire /usr/include
`ls /usr/inc`[Appuyer sur Tab] → on obtient la complétion **`ls /usr/include/`**

Donc on écrit le début d'un nom de répertoire (ou de fichier) et en appuyant sur tab on a la suite.
Si il y a plusieurs noms correspondant Tab ne complète pas, Tab une 2ème fois pour avoir la liste...

`ls /u`[Tab] → **`ls /usr/`** ajouter juste s **`ls /usr/s`**[Tab][Tab] → plusieurs réponses compléter **`ls /usr/sh`**[Tab] → **`ls /usr/share/`** (liste de différents utilitaires du système)

L'efficacité du terminal/shell pour piloter un système tient à l'utilisation assidue de ces raccourcis.
Par exemple vous pouvez compléter les commandes : **`his`**[Tab] → **`history`**

5/ Le système de fichier est votre terrain de jeu et le shell est votre bulldozer...

Il est temps de créer un espace de travail.
Vérifiez que vous êtes bien dans votre HOME avec **`pwd`**, refaire **`cd ~`** si nécessaire.

`mkdir tp1`

Make Directory : créer répertoire tp1 dans le répertoire courant (dans HOME donc)
Attention pour vos choix de noms de fichiers : utiliser des caractères spéciaux (autres que _ ou -) accents ou même seulement des espaces c'est chercher les ennuis. Il reste possible de "quoter" pour gérer des chemins avec espaces : cd "Mon TP1 avec espaces" par exemple mais ça complique.

```
cd tp1
mkdir test1
mkdir test2
cd test1
```

Vous pouvez vérifier qu'on a bien dans home un sous-répertoire tp1 qui lui même contient 2 sous-répertoires test1 et test2 en utilisant l'explorateur de fichiers (poste de travail).

6/ Redirection des flux standards stdin et stdout

La commande cat prend les caractères du flux d'entrée stdin (par défaut le clavier) et les répercute directement sur le flux de sortie stdout (par défaut l'affichage du terminal).

```
cat
```

Vous remarquez qu'il n'y a plus de prompt : nous ne sommes plus en communication avec le shell mais ce que nous écrivons est maintenant intercepté par cat en cours d'exécution.
Entrer le texte suivant (↵ représente un appui sur la touche Entrée) :

```
mon message↵
est reproduit↵
```

[Appuyer sur la combinaison Ctrl-D pour terminer la saisie]
Ctrl-D envoie une "fin de fichier" (End Of File, EOF) ce qui marque la fin de notre entrée, normalement vous retrouvez le prompt du shell. En cas de blocage dans une commande qui ne rend pas la main au shell il est toujours possible en dernier recours de tuer brutalement la commande avec Ctrl-C.

L'intérêt de la commande cat apparaît avec l'utilisation des redirections de flux :

```
cat >fichier.txt
```

puis entrer le texte suivant (terminer par Ctrl-D)

```
mon message↵
est enregistré↵
```

[Ctrl-D]

Le symbole **>** redirige le flux de sortie vers un fichier (créé pour l'occasion si il n'existait pas)

```
ls -l
```

On a bien créé fichier.txt dans le répertoire courant. Vérifions son contenu :

```
cat <fichier.txt
```

(vous avez pensé à n'utiliser que **<f** puis Tab pour compléter le nom ?)

Cette fois ci on redirige le flux d'entrée pour que l'information vienne du fichier au lieu du clavier.
Ces redirections < et > … sont valables quelque soit la commande (l'exécutable) utilisé.

En fait cat peut recevoir directement un nom de fichier en paramètre (à la place de stdin)

```
cat fichier.txt
```

C'est la façon usuelle d'afficher le contenu d'un court fichier texte. La commande s'appelle cat car elle permet de concaténer plusieurs fichiers, par exemple si on disposait de 3 fichiers on pourrait grouper leur contenu `cat fic1.txt fic2.txt fic3.txt >tout.txt`

Si le fichier à voir est trop grand on peut demander moins avec la commande less …

```
ls -l /bin >commandes.txt
ls -l
```

On obtient le listing des commandes principales du répertoire /bin dans le fichier commandes.txt

```
less commandes.txt
```

On peut scroller tranquillement dans le fichier (haut/bas page_haut/page_bas) q pour quitter

Lorsqu'il est utilisé sans fichier en paramètre, less prend comme contenu à traiter ce qu'il reçoit sur stdin (c'est le cas de nombreuses commandes Unix). On aurait pu écrire `less <commandes.txt`
En résumé on a redirigé le flux de sortie de ls sur un fichier et ensuite on redirige le flux d'entrée de less depuis ce fichier. Dans ce genre de situation on peut aussi bien se passer de fichier et brancher directement le flux de sortie de ls sur le flux d'entrée de less avec un **tube (pipe)** **|**

```
ls -l /bin | less
```

on peut chaîner de nombreux traitements élémentaires ainsi :
`ls -l /bin | tr -s ' ' | cut -d ' ' -f5,9 | tr ' ' '\t' | less`

Facultatif : consulter le manuel de ces différentes commandes pour comprendre ce qui se passe...
On va retravailler avec fichier.txt ...

`cat >fichier.txt`
`un nouveau message`↵
`redirigé vers le fichier`↵[Ctrl-D]
`cat fichier.txt`

On a effacé/remplacé l'ancien contenu par le nouveau. Pour ajouter il faut utiliser la redirection >>

`cat >>fichier.txt`
`et ceci`↵
`vient s'ajouter à la suite!`↵[Ctrl-D]
`cat fichier.txt`

On vérifie bien qu'il y a ajout.

7/ Et ensuite ?

Pour quitter le shell et fermer le terminal : **`exit`**

Il manque à cette introduction à Linux les essentiels suivants (et bien d'autres) que vous rechercherez en autonomie :

`cp` : copie de fichiers et répertoires

`mv` : déplacer ou renommer fichiers et répertoires

`rm` et **`rmdir`** : suppression de fichiers et répertoires

`chmod` et **`chown`** : changer les droits d'accès et propriétaires sur fichiers et répertoires

`clear` : effacer le terminal

`ps` et **`jobs`** et **`top`** : voir les commandes en cours d'exécution (gestion des processus)

`kill` : tuer un exécutable (marche aussi sur les processus zombies)

Les **wildcards** qui permettent de filtrer des ensembles de noms :
exemple **`ls *.c`** lister seulement les fichiers .c dans le répertoire courant

Cette introduction doit beaucoup à l'ouvrage de poche suivant :

Parlez-vous Shell ?
Initiation à la ligne de commande Unix
Thomas Hugel éditions ellipses