IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Premiers pas avec le terminal/shell

Semaine d'intégration – TAF DCL

Objectif

L'objectif de cette session est de se familiariser avec les commandes Unix/Linux que vous pouvez être amenés à utiliser au cours de votre scolarité, voire de votre vie professionnelle.

L'interface en ligne de commande

Le travail à l'aide des interfaces graphiques est souvent facile et intuitif, même si elles sont finalement assez mouvantes (le nombre de boutons à cliquer, leur nom, le rôle, peut varier d'une version à l'autre...). Cependant sous Unix il existe un très grand nombre d'outils accessibles de préférence en *ligne de commande* dans un terminal ou un émulateur de terminal. Ce sont *les commandes Unix*. Il est souvent préférable de les utiliser pour un travail rapide et efficace, mais leur manipulation nécessite un apprentissage.

1 Notion de Shell et de commande

Ouvrez un émulateur de terminal : dans la barre de menus du haut, choisissez `Applications`. Là vous trouverez l'application `Terminal` soit dans l'onglet `Favoris`, soit dans l'onglet `Utilitaires` [1]

Une fenêtre apparaît à l'écran. Elle affiche un petit texte abscons appelé l'*invite* ou *prompt* qui comme son nom l'indique vous invite à saisir une commande. En fait, vous avez deux choses à ce niveau : vous avez (1) une application graphique qui sait afficher une fenêtre, gérer un petit ascenseur de défilement, afficher des caractères ASCII et sous différents encodages, et vous avez (2) un interpréteur de commande qui porte le nom de Shell.

Le Shell associé à votre fenêtre vous demande d'entrer une commande au clavier. Lorsque vous allez le faire, il va analyser cette commande puis gancer son exécution. Cette commande peut être complexe, elle correspond généralement à un fichier exécutable mais on peut paramétrer son

1. Cela peut changer d'une version à l'autre de l'environnement graphique. Parfois c'est `Accessoires` ou `Système`.

exécution de différentes manières. L'analyse préalable à l'exécution consiste à prendre en compte ce paramétrage.

2 Les commandes Unix et le Shell

2.1 Les répertoires . et ..

Dans la fenêtre d'émulation de terminal tapez la commande `ls -al`. Vous devez voir apparaître la liste du contenu du répertoire dans lequel vous êtes, incluant les fichiers et répertoires cachés. Par rapport à la liste affichée dans le navigateur Nautilus vous avez en plus les répertoires `.` et `..`

Exécutez les commandes suivantes :

- `pwd` (print working directory)
- `cd ..` (change directory)
- `pwd`
- `cd .`
- `pwd`

Qu'en déduisez-vous sur le rôle de ces répertoires :

- le répertoire `..`
- le répertoire `.`

2.2 La racine

Remontez au niveau le plus haut de l'arborescence (soit avec de multiples `cd ..`, soit avec `cd /`), puis utilisez la commande `ls -ld`.

- Comment s'appelle la racine ?
- Qui est le propriétaire de la racine de l'arborescence ?
- Quels sont vos droits sur ce répertoire ?

2.3 Sauter de branche en branche

Placez-vous dans le répertoire `/usr/bin`. (Commande `cd`)
Sautez dans le répertoire `/usr/lib` (puis revenez) en utilisant chacune des deux méthodes : le *nommage absolu* et le *nommage relatif*.
Amusez à refaire la même chose en sautant de `/usr/share/man/man1` à `/usr/share/man/man2` et réciproquement.

- Dans quel cas peut-on préférer le *nommage absolu* et dans quel cas peut-on préférer le *nommage relatif* ?

2.4 Vos fichiers et ceux des autres (machines de l'école)

- Revenez à votre répertoire de travail : `cd`
- Remontez au répertoire parent : `cd ..`
- Vérifiez tout ça : `pwd` puis `ls`
  En principe, vous devriez voir un sous-répertoire, celui à votre nom, votre répertoire de travail.

— Essayez d'aller dans le répertoire d'un collègue : `cd` *`login_du_collègue`*
— Vérifiez tout ça : `pwd` puis `ls`
— Remontez au répertoire parent : `cd ..`
— Vérifiez tout ça : `pwd` puis `ls`
C'était comme ça avant ? Que s'est-il passé ?

Vos fichiers ne sont pas présents physiquement sur chacun des PC d'école. Ils sont localisés dans une baie de stockage. Chacun des PC peut y accéder en réseau. Lorsqu'un PC en a besoin, il procède à un montage, puis, après un certain temps d'inutilisation les fichiers sont démontés. C'est automatique et transparent. Typiquement, lorsque l'on se connecte à un PC, il a besoin de nos fichiers, et procède au montage. Mais si l'on demande à accéder au répertoire du collègue, il procède au montage également. (Ensuite, le fait que le collègue ait positionné des permissions Unix sur ses fichiers est une autre histoire.) C'est un bon moyen pour partager simplement des fichiers (pas besoin de clef usb ou de transfert réseau, c'est déjà en réseau).

2.5 Les méta-caractères

Lancez la commande suivante : `ls -l /bin/mk*`
— Que constatez-vous ?
— En déduire le rôle du caractère `*`
— Lancez maintenant la commande `ls -ld /usr/bin/zi?` et comparez avec `ls -ld /usr/bin/zi*`
— En déduire le rôle du caractère `?`
— Comparez par exemple `ls -ald .*` et `ls -ald .????*`

2.6 La complétion

Apprenez à taper plus vite que votre ombre !

Note : l'utilisation de la complétion dépend fortement du type de shell que vous avez sur votre système (tapez `echo $SHELL` pour savoir).

— Entrez la commande `ls -l /bin/mk` suivie immédiatement par la touche <`Tab`>. La touche tabulation est à gauche du clavier, et souvent représentée ⇆
Que constatez-vous ?
— Faites à nouveau <`Tab`> et voyez le résultat. Quelle aide est alors apportée ?
— Complétez la commande avec quelques caractères pour lever toute ambiguïté et réutilisez la touche <`Tab`> pour afficher finalement automatiquement le nom `/bin/mkdir`

Notez que la complétion est une fonctionnalité de confort offerte par le shell que vous utilisez (`bash`, `tcsh`, etc.). Avec des shells plus anciens comme `sh`, cela ne marche pas... Avec les plus modernes on peut faire de la complétion très subtile, sur des noms d'options de commande ou des arguments (par exemple après `man` on n'aura de la complétion que sur des noms de commande dont on a le manuel en ligne) ou encore de la correction orthographique.

2.7 Les redirections

— Retournez dans votre répertoire de travail en tapant simplement : `cd`

— Demandez l'heure : `date`
Que constatez-vous ?
— Redirigez maintenant la sortie de la commande précédente dans le fichier `maintenant.txt` dans votre répertoire d'accueil : `date > maintenant.txt`
Que constatez-vous ?
— Listez le contenu de votre répertoire de travail : `ls -l`
Alors ?
— Affichez le contenu de ce fichier : `cat maintenant.txt`
Comparez avec la date maintenant : `date`

2.8 Les tubes de communication (les *pipes*)

Combien de fois avez-vous tapé la commande `cd` depuis tout à l'heure ?

Non, ne cherchez pas dans votre tête, l'ordinateur a meilleure mémoire que vous.

— Consultez l'historique de votre interpréteur de commande : `history`

C'est à la fois le contenu du fichier caché `~/.bash_history` et le contenu de ce qu'il a en mémoire depuis que vous avez ouvert votre terminal. Lorsque vous fermez votre terminal, ce fichier `.bash_history` est mis à jour.
— Tout s'affiche trop vite à l'écran. Paginez via la commande `more` ou la commande `less`[2] de la manière suivante : `history | less`

Vous pouvez paginer avec les touches classiques <`Page Up`> <`Page Down`>. De plus, pour diverses raisons historiques (les terminaux n'ayant pas toujours eu des claviers si sophistiqués...), vous pouvez utiliser la touche <`Espace`> pour afficher page par page ou avec la touche <`Entrée`> pour afficher ligne par ligne. Pour sortir vous tapez <`Q`> (Quit).
— Bon, reprenons, nous n'allons pas faire le travail à la main, mais le faire faire par l'ordinateur. C'est pour cela qu'il est là. Faisons-le chercher les lignes qui contiennent `cd` :
`history | grep cd`
— Il ne reste plus qu'à lui faire compter les lignes :
`history | grep cd | wc -l`

2.9 Les commandes en premier et arrière plan (*foreground*, *background*)

— Lancez la commande `xeyes`. Dans le même terminal, lancez maintenant une commande quelconque (`pwd` par exemple). Que constatez-vous ?
— Faites <`Ctrl-C`> pour tuer la commande `xeyes` et relancez-là en arrière plan (cherchez comment faire soit dans le cours, soit sur la page web indiquée précédemment). Que constatez-vous comme différence ? Que pouvez-vous faire, alors, qui vous était impossible lorsque la commande était lancée en premier plan ?

2. Le monde d'Unix est un monde d'humour, en particulier au niveau des noms de commande. Un humour un peu élitiste parfois...

3 La documentation des commandes Unix

Ayez le réflexe RTFM : *Read That Fine Manual*. Chaque commande Unix est documentée dans le manuel Unix en ligne et accessible via la commande `man`.

Dès que vous avez un problème avec une commande, faites : `man` *`nom_de_la_commande`*

Vous pouvez chercher un terme dans la page de manuel en tapant `/mot_a_rechercher` qui indique la première occurrence du terme. Pour visualiser les occurrences suivantes, utilisez la touche `n` (*next*). Pour quitter la page de manuel, utilisez `q` (*quit*). (En fait ces touches sont gérées par votre *pageur*, typiquement la commande `more` ou `less`, et dont vous pouvez lire le man...).

Chaque documentation est structurée de la même manière. Vous trouverez essentiellement :
- le nom de la commande et sa description courte ;
- un synopsis, résumé de la manière d'utiliser la commande ;
- une description plus complète du fonctionnement de la commande, avec souvent des exemples concrets d'utilisation ;
- la liste des options acceptées, avec leur signification ;
- à la fin de la page de manuel, la section SEE ALSO peut vous guider vers d'autres commandes connexes.

De manière générale, prenez le temps et la peine de consulter ces pages de manuel. (J'insiste car c'est comme cela que l'on apprend beaucoup !)

À l'aide du manuel de référence, répondez aux questions suivantes :
- Comment faire une copie récursive de fichiers et de répertoires (commande `cp`) ?
- Comment afficher le contenu d'un répertoire dans l'ordre alphanumérique inverse (commande `ls`) ?
- Quelles sont les options de la commande `man`, et plus particulièrement, que fait l'option `-k` ?
- Quelles sont les pages du manuel traitant du mot *listing* ?
- Quelles sont les pages du manuel traitant du mot *image* ?

Notez également que l'interpréteur de commande (le shell) comprend également quelques commandes internes (`cd pwd ..`). C'est donc dans le `man` de votre shell que vous trouverez des explications sur ces commandes. De plus certains shell modernes peuvent vous fournir de l'aide en ligne sur leurs commandes internes. Par exemple sous `bash` on peut faire `help cd`.

4 Mini-exercice : préparer l'espace de travail DCL, uniquement avec le terminal

Si vous ne l'avez pas encore fait, il est temps de préparer votre espace de travail pour la TAF DCL et pour chaque UE en utilisant uniquement le terminal. Et dans l'ídéal, en minimisant le nombre de commandes utilisées. Vous utiliserez les commandes suivantes : `cd`, `ls`, `pwd`, `mkdir`, `rmdir`, `mv` et `rm`.

Proposition d'organisation de fichiers (visualisé grâce à la commande `tree`) :

```
DCL/
├── CONC
│   ├── miniprojet
```

```
│   └── tp
├── ECO
├── IDL
│   ├── projet
│   └── tp
├── MAPD
│   ├── filrouge
│   └── tp
└── adm
```

5 Paramétrage de votre environnement de travail en mode terminal (sur les machines de l'école)

Votre environnement est déjà configuré pour vous permettre d'utiliser la plupart des commandes classiques. Cependant, si l'on veut utiliser une commande non-classique (p.ex. pour faire un exercice de TP particulier, où parce que c'est une commande que l'on a installée soi-même), il faut paramétrer un certain nombre de choses dans notre environnement.

La chose la plus importante est de savoir où se trouve cette commande. Ainsi on va positionner une variable de votre environnement de façon à ce que le shell trouve cette commande sans problème. La variable en question est `PATH` (en majuscules) et contient la liste des répertoires dans lesquels se trouvent les commandes que l'on peut appeler par leur nom simple. Pour consulter son contenu, utilisez la commande `"echo $PATH"`.

Notez que cette variable `PATH` peut être vide, par exemple si votre shell a été installé (et compilé) avec un chemin par défaut comme `/bin:/usr/bin`

5.1 La commande `SETUP`

La modification environnementale à effectuer est complexe pour le néophyte, aussi, à l'école, nos administrateurs système ont développé une commande spéciale appelée `SETUP` (en majuscules) qui permet de faciliter les choses.

- Exécutez cette commande pour lister les logiciels accessibles
- Affichez le *chemin* connu par votre environnement pour rechercher les commandes utilisateur : `echo $PATH`
- Faites la commande `java -version`, et repérez la version par défaut de votre java.
- À l'aide de la commande `SETUP`, demandez à changer de version (p.ex. `SETUP JAVA17` ou `SETUP JAVA18`)
- Vérifiez que désormais vous travaillez avec une autre version de java : `java -version`
- Vérifiez les changements dans votre configuration : `echo $PATH`

Attention, la commande `SETUP` ajoute mais n'enlève pas... Par exemple, si dans un même terminal vous faites `SETUP JAVA17` puis `SETUP JAVA18`, vous vous retrouvez avec un environnement configuré pour rechercher *d'abord* la version 1.7 de java, et ensuite la version 1.8... et qui donc va toujours trouver d'abord java 1.7 ! (Pour vous en convaincre, faites des `java -version` ou `echo $PATH`.)

Ceci dit, rien ne vous empêche de faire `SETUP JAVA17` dans un terminal, et `SETUP JAVA18` dans un autre...