TRAVAUX PRATIQUES

Introduction au shell

Dans le cours d'informatique de MP2I/MPI, nous utiliserons un système d'exploitation (*operating system*, ou *OS*) GNU/Linux : Ubuntu (version 22.04 LTS). Celui-ci a l'avantage d'être l'un des plus populaires auprès des nouveaux venus dans le monde merveilleux de Linux, et d'avoir beaucoup d'aide en ligne accessible.

Cous devrez l'installer sur votre machine [1]. Une install party aura lieu mardi 12 septembre prochain, entre 16h et 18h : venez avec votre machine, et nous installerons tout dessus.

Le but de ce TP est de vous apprendre à utiliser un *interpréteur de commande* (aussi appelé *terminal*, ou *shell* dans la suite). Il s'agit d'une sorte de "ligne directe" avec l'ordinateur, dans laquelle on pourra donner des instructions directes à l'OS.

Le terminal nous sera très utile : taper des commandes dedans est souvent plus rapide que d'enchainer les clics dans une fenêtre [2], et permet plus d'adaptabilité. C'est un outil indispensable à tout·e futur·e informaticien·ne.

Appelez-moi si quelque chose n'est pas clair ou si vous bloquez !

A Premiers pas

Commençons les bases de l'utilisation d'un terminal.

1. Ouvrez un terminal. Pour cela, cliquez sur la petite icone de fenêtre noire présente dans votre menu ; on peut aussi l'ouvrir en utilisant le raccourci 'Ctrl+Alt+T'.

Dans tout ce TP, les chevrons < > jouent le rôle de guillemets : ils ne font pas réellement partie des commandes, et il ne faut donc pas les recopier ou les chercher en vain.

Le terminal devrait contenir une ligne commençant par `<nom_d_utilisateur>@<nom_de_la_machine>:~$` :

- `<nom_d_utilisateur>` est... le nom de l'utilisateur actuellement connecté sur la machine.
- `<nom_de_la_machine>` est le nom de la machine sur laquelle on se trouve.
- `~` indique que l'on se trouve dans le dossier personnel de l'utilisateur. Quand on changera de dossier, sa valeur changera pour toujours indiquer le dossier courant.
- `$` indique le niveau de permissions actuel : `$` signifie que nos commandes s'effectueront avec les permissions utilisateur (on ne pourra donc modifier que ce qui nous appartient, et on ne pourra pas toucher au fonctionnement du système). Sur les machines du lycée, vous n'aurez jamais les permissions superutilisateur (si vous venez de Windows, pensez "droits administrateur"), mais vous devriez les avoir sur votre machine personnelle.

Pour lancer une commande, il faudra la taper au clavier et confirmer avec Entrée.

A.1 Arborescence des fichiers

Dans votre système, les dossiers sont organisés de manière arborescente : un dossier est inclus dans un autre, etc, et ce jusqu'à un dossier appelé racine. Chaque fichier appartient à un dossier.

La racine est appelée " `/` ". Elles contient différents dossiers comme par exemple `var`. `var` contient lui-même d'autres fichers et dossiers, comme par exemple `lib`, qui contient lui-même `systemd` [3]. A chaque fichier ou dossier, on associe son chemin d'accès (*access path* ou simplement *path*) : c'est son "adresse" . Ainsi, celle de `lib` a comme chemin : `/var/lib` .

Définition 1 : Chemins

Le chemin de la racine est `/` .
Le chemin d'un fichier ou d'un dossier qui n'est pas la racine se décompose ainsi :

1. Ordre de recommandation : dualboot avec votre OS actuel > machine virtuelle dans Windows/Mac > Windows Subsystem for Linux. Si vous avez déjà un autre Linux basé sur Debian, inutile d'installer Ubuntu en parallèle.
2. Sisi, vous allez finir par trouver la souris lente.
3. Si vous trouvez cet exemple polémique, vous pouvez probablement passer à section suivante.

> `<chemin_vers_le_dossier_parent_du_fichier>/<nom_du_fichier>`
>
> Pour différencier les chemins de dossier des chemins de fichiers, on fait souvent terminer les premiers par `/` : `/etc/iptables/` est un chemin de dossier, mais `/etc/group` peut-être un chemin de fichier ou de dossier. On distingue deux types de chemins :
>
> — les **chemins absolus**, c'est à dire adressés depuis la racine. Par exemple : `/home/<nom_de_l_utilisateur>/Documents/sous_dossier_favori` .
>
> — les **chemins relatifs**, qui ne sont pas adressés depuis la racine. Par exemple : `Documents/sous_dossier_favori` . Lorsqu'une commande essaye de les évaluer, il les complète en chemin absolu en ajoutant le dossier courant comme préfixe. Ainsi, si on se trouve dans `/etc` lorsqu'on évalue le chemin relatif précédent, il sera évalué comme `/etc/Documents/sous_dossier_favori` .
>
> Trois noms de dossiers sont réservés pour servir de raccourci à des dossiers particuliers :
>
> — `~` est le chemin absolu `/home/<nom_de_l_utilisateur>/` , qui est le chemin du dossier personnel.
>
> — `.` est un lien vers le dossier courant.
>
> — `..` est un lien vers le dossier parent.

`~` donc est votre dossier personnel d'utilisateur, où sont rangés vos dossiers et fichiers personnels. Les dossiers en dehors de `~` sont des dossiers du système d'exploitation et des programmes installés, où ils stockent tout ce qui est nécessaire à leur fonctionnement (y compris… eux-mêmes).

2. Dans le terminal, lancez la commande `pwd` (*Print Working Directory*). Qu'affiche-t-elle ? Quelle est le chemin affichée dans le préfixe du terminal ?

Savoir où l'on est est une chose, encore faut-il savoir ce qu'il y a ici. Pour cela, on utilise la commande `ls` (*LiSt*).

3. Lancez la commande `ls` . Quel est le résultat ? Comment le comprenez-vous ?

On reviendra à `ls` plus tard. Pour le moment, bougeons ! La commande `cd` (*Change Directory*) permet de changer de dossier.

4. Lancez `cd /Documents/` . À votre avis, quelle est le chemin absolu du dossier où vous vous trouvez maintenant ? Confirmez avec `pwd` .

5. Revenez dans votre dossier personnel avec `cd ~` . Tapez `cd Do` sans valider avec Entrée, puis appuyez sur Tab. Que constatez-vous ? Lancez la commande. Où êtes-vous ?

À partir de maintenant, utilisez l'autocomplétion de Tab dès que possible ! Le temps gagné n'est pas négligeable.

On peut en fait donner quatre types d'arguments à `cd`. Supposons pour l'exemple que vous soyez dans votre dossier personnel, `~` :

— un chemin absolu

— un chemin relatif

— `.` ou `..`

— aucun argument, ce qui est équivalent à `cd ~`

Quand il change de dossier, `cd` change son dossier courant au fur et à mesure (ce qui est important pour les chemins relatifs). Ainsi, `cd /var/..` se déplace d'abord dans `/` puis dans `/var/` et enfin dans le dossier parent de `var/` .

6. **a.** Si on enchaine les commandes `cd` puis `../../.` , où sera-ton ? Pourquoi ?

b. Testez et vérifiez.

7. **a.** Retournez dans votre dossier personnel. À l'aide de `ls` et `cd` , explorez les dossiers et sous-dossiers de votre dossier personnel. Tracez une représentation de son arborescence.

b. Retournez dans votre dossier personnel. Lancez la commande `tree` , que fait-elle ? Comparez avec votre résultat de la question précédente.

Voilà. Vous devriez maintenant savoir vous déplacer dans le terminal (et avoir commencé à prendre le réflexe d'**utiliser l'autocomplétion** !). Notez que l'autocomplétion n'est pas propre à `cd` : elle marche pour toutes les commandes.

B Jeu en shell

Pour la suite de ce TP, dans un nouveau terminal lancez la commande `bash gameshell.sh` . Elle va lancer un jeu pédagogique pour vous faire pratiquer les premières commandes vues jusqu'à présent, et vous en faire découvrir d'autres.

8. Lancer `bash gameshell.sh` depuis `~` , et faites les missions. Le but est d'aller le plus loin possible !

Vous n'irez pas tous aussi loin en fonction de votre pratique antérieure du shell. Ce n'est pas grave. L'important est d'apprendre et de progresser ; les compétitions toxiques sont à proscrire.

C À faire d'ici au prochain cours/Td/TP

C.1 De la lecture

Toutes les commandes ont une page de documentation expliquant leur fonctionnement ainsi que toutes leurs options. On accède à cette page de documentation grâce à la commande `man` [4]. Vous pouvez vous déplacer dans la documentation avec les flèches du clavier, et la quitter avec `q` .

9. Lancez `man cp` . Selon l'introduction, que fait la commande `cp` ? Vous pouvez quitter la documentation avec `q`.

10. Lancez `man mv` . Selon cette documentation, que fait `mv --version` ? Et `mv --help` ? Testez et vérifiez.

Quand la documentation est longue, on peut avoir envie de faire une recherche dans le texte. Pour cela, une fois la page de documentation ouverte, tapez `/mot_a_chercher` : les occurences vont être surlignées. Pour aller à la prochaine occurence du mot cherché, utilisez `n` (**N**ext) ; pour quitter le mode recherche, appuyez sur Échap .

11. Lancez `man man` , et cherchez ce que fait l'option `-f`

Les pages de manuels sont organisées en *section*, comme les rayonnages d'une bibliothèque. La section que nous utiliserons le plus souvent cette année est la section 3, qui contient la documentation des langages de programmation. Par exemple, `man 3 assert` demande de rechercher `assert` parmi les langages de programmation (et vous devriez trouver la documentation de `assert` en C… on y viendra bientôt !)

C.2 Et le reste ?

Nous verrons d'autres commandes au cours de l'année. Vous serez peut-être amenés à en découvrir d'autres par vous-même [5].

4. Oui, `man` est lui-même une commande, donc `man man` affiche la documentation de `man` . C'est méta.
5. Par exemple : mais comment diable met-on à jour son Ubuntu ? Et pour installer ou supprimer des programmes ?