Séance 1: Premiers Pas

Université Paris-Diderot

Objectifs:

- S'inscrire au Moodle du cours
- Apprendre à manipuler le Shell
- Utiliser l'interpréteur Python
- Savoir exécuter un programme donné

Première connexion

Se connecter est la première chose à faire. Lorsque vous démarrez l'ordinateur, choisissez `FreeBSD` comme système d'exploitation. Le système démarre et affiche une fenêtre vous invitant à taper votre identifiant (*login*) et votre mot de passe (*password*), ce qui permet d'ouvrir une session.

Une fois votre nom d'utilisateur et mot de passe vérifiés, le gestionnaire de bureau (le programme qui gère l'affichage des menus et des fenêtres, ici `XFCE`) apparaît. Le menu principal (bouton en bas à gauche) permet de lancer des applications, d'accéder aux outils de paramétrage du système ou encore de fermer la session ou d'éteindre l'ordinateur.

La plupart des programmes ainsi que le menu principal disposent d'une entrée « aide » (*help*) permettant d'accéder à l'aide en ligne. Si vous êtes coincé(e), n'hésitez pas à la consulter.

Exercice 1 (Inscription sur le site du cours IP1 Python, ⋆)

1. *Lancez le navigateur* Firefox.
2. *Ouvrez la page de la plateforme* Moodle *:* http://moodle.script.univ-paris-diderot.fr/.
3. *Cliquez sur «* Connexion *»(en haut à droite) pour vous identifier (vus êtes renvoyé à la connexion à l'ENT).*
4. *Allez dans «* Sciences *»puis «* Département de formation de Licence L1 L2 sciences exactes (SE) *» ; puis «* L1 *».*
5. *Cliquez sur «* Initiation à la Programmation en Python *» et inscrivez-vous.*
6. *Dans la sections, «* Supports TP *», vous trouverez les énoncés des différents TP ainsi que les programmes donnés.*

□

À la découverte du *Shell*

Commandes. Il est possible d'interagir avec le système de manière plus fine que via l'interface graphique, en utilisant un *terminal* (ou *console*) dans lequel peuvent être tapées des *lignes de commande*. Ces lignes sont interprétées par un programme appelé *shell* dont le rôle est d'attendre que vous lui demandiez d'exécuter une commande pour le faire ; la fin de la saisie d'une ligne de commande est indiquée en appuyant sur la touche `Entrée` (`Enter`).

Le shell indique qu'il est prêt en affichant en début de ligne une *invite de commande* (ou *prompt*) terminant en général par le caractère « dollar »($) ou « supérieur »(>).

La forme générale d'une ligne de commande est la suivante :

`cmd` opts args

où

- `cmd` est le nom de la commande à exécuter ;
- `opts` est une liste (éventuellement vide) d'options, permettant d'affiner le comportement de la commande ; en général, une option est de la forme « tiret lettre »(par exemple, `-a`, `-l`...) ;
- `args` est une liste (éventuellement vide) d'arguments en fonction desquels la commande agit.

La commande peut en particulier être le nom de n'importe quelle application qui pourrait être lancée depuis l'interface graphique (comme `firefox` par exemple), mais il existe également tout un ensemble de *commandes UNIX* qui interagissent avec le terminal.

Édition de ligne. Si on se trompe en tapant une commande, et qu'on s'en aperçoit avant d'appuyer sur Entrée, on peut utiliser les touches ← et → pour déplacer le curseur à l'endroit où est l'erreur.

Historique. Si on ne s'aperçoit de l'erreur qu'après avoir démarré la commande, on veut souvent lancer une autre commande corrigée. Au lieu de tout retaper, on peut utiliser la touche ↑, qui rappelle la commande précédente (puis la commande d'avant, etc., si on appuie plusieurs fois). La touche ↓ permet de redescendre dans l'historique des commandes, vers les commandes les plus récentes.

Complétion. Lorsqu'on veut taper le nom d'un fichier existant, on peut taper le début du nom du fichier puis appuyer sur la touche *tabulation* (marquée Tab ou désignée parfois avec une grande flèche vers la droite). Le shell insère alors la fin du nom (s'il y a plusieurs possibilités, le shell complète seulement le plus long préfixe commun). La complétion a deux avantages : elle permet de moins taper, et elle assure que le nom complété existe.

La commande `man`. Le manuel en ligne pour toutes les commandes accessibles depuis le terminal. Il suffit de taper `man` *`cmd`* pour accéder à la description complète de la commande *cmd*.

Par exemple, la ligne de commande `man ls` permet d'obtenir la documentation de la commande `ls`. On peut faire défiler le texte à l'aide de la barre d'espace et des flèches ↑ et ↓, et quitter à l'aide de la touche `q`.

Exercice 2 (Utilisation de man, ⋆)

1. *Tapez `man ls` et analysez la structure de la page de manuel.*
2. *Que fait la commande `ls` ?*
3. *À quoi sert l'option `-l` ?*

□

Fichiers. Un fichier est une suite de données, représentant par exemple un texte, une image etc. Chaque fichier possède un nom, conventionnellement terminé par un point et une suite de caractères indiquant le type de données qu'il contient. Par exemple, le fichier qui contient l'énoncé de ce TP s'appelle `tp1-ip1-python.pdf`, et son nom indique qu'il est au format PDF.

Les systèmes Unix (comme ceux de la salle TP) font une différence entre majuscules et minuscules : `tp0.pdf`, `TP0.pdf` et `TP0.PDF` désignent trois fichiers différents.

Répertoires. Sur les systèmes unix, les fichiers sont organisés sous forme d'un *arbre* : chaque fichier est stocké dans un *répertoire* (aussi appelé *dossier*) et les répertoires peuvent eux-mêmes contenir d'autres répertoires.

Le répertoire *home* Le répertoire dit *home* (« maison », ou parfois *répertoire personnel*), noté « ~ », est l'endroit où vous pouvez stocker vos fichiers personnels. Où que vous soyez, si vous tapez « `cd ~` », vous vous retrouverez dans le répertoire *home*.

Quelques commandes utiles

- La commande `mkdir` *rep* permet de créer un répertoire *rep*, sous-répertoire de votre répertoire courant.
- La commande `cd` *rep* permet de « descendre »dans le répertoire *rep* (**Attention :** il faut que ce répertoire soit un sous-répertoire du répertoire où vous vous trouvez).
- La commande `cd ..` permet de remonter au répertoire parent.
- La commande `pwd` permet de savoir où vous trouvez dans l'arbre.

Exercice 3 (Création de répertoires, ⋆)

1. *Allez à votre répertoire* home.
2. *Faîtes* `ls -l`.
3. *Créez un répertoire* `IP1-Python`.
4. *Faîtes* `ls -l`. *Qu'observez-vous ?*
5. *Descendez dans le répertoire* `IP1-Python`.
6. *Faîtes* `pwd`.
7. *Faîtes* `cd ...` *Où vous trouvez-vous dans l'arborescence ? Vérifiez-le en faisant* `pwd`.
8. *Descendez dans le répertoire* `IP1-Python` *et créez un sous-répertoire* `TP1`.
9. *Faîtes* `ls -l` *et vérifiez ainsi que le répertoire* `IP1-Python` *contient un sous-répertoire* `TP1`.

□

Important : Prenez l'habitude de créer un sous-répertoire par TP.

Le système d'exploitation que vous utilisez met à votre disposition différents éditeurs de texte (*kwrite*, *gedit*, *vi*, *Emacs* etc.). Nous vous proposer d'utiliser *Emacs*, qui est particulièrement bien adapté à la programmation.

Pour lancez le programme Emacs, il suffit de taper « `emacs &` ».

Exercice 4 (Création d'un premier fichier avec Emacs, ⋆)

1. *À l'aide d'Emacs, créez un fichier* `poeme.txt` *qui sera à mettre dans le répertoire* `IP1-Python/TP1/` *; assurez-vous que le tampon est bien en mode* Text *(regardez en bas de la fenêtre). Tapez votre poème favori (quelques vers suffiront). Sauvegardez le fichier dans le repértoire* `IP1-Python/TP1/poeme.txt`*, mais ne fermez pas.*
2. *Toujours dans le même Emacs, créez un fichier* `chanson.txt` *où vous taperez les paroles de votre chanson favorite (quelques vers suffiront). Sauvegardez.*
3. *Revenez au tampon du fichier* `poeme.txt`. (Indication : regardez dans le menu *Buffers*.)
4. *Dans le terminal, allez voir avec la commande* `ls -l` *si dans le répertoire* `IP1-Python/TP1/` *il y a bien les fichiers* `poeme.txt` *et* `chanson.text`.
5. *Fermez Emacs.*
6. *Dans le terminal, dans le répertoire,* `IP1-Python/TP1/`*, tapez* `emacs poeme.txt &` *. Qu'observez-vous ?*
7. *Fermez Emacs.*

□

Il est possible d'utiliser `Python3` depuis le terminal grâce à l'interpréteur `Python3`. Pour cela il suffit de lancer l'interpréteur avec la commande `python3`. Une fois cela fait, vous n'êtes plus dans le shell mais tout ce que vous tapez sera interprété comme du `Python`. Pour quitter l'interpréteur, il suffit de taper `quit()`.

Exercice 5 (Utilisation basique de l'interpréteur, ⋆)

1. *Lancez l'interpréteur* `Python3`.
2. *Tapez* `2/3` *puis Entrée. Que voyez-vous ?*
3. *Tapez* `6//3` *puis Entrée. Que voyez-vous ?*
4. *Tapez* `2//3` *puis Entrée. Que voyez-vous ? La différence avec* `2/3` *vient du fait que la division* `//` *est sur les entiers.*
5. *Tapez* `type(2/3)` *puis Entrée et ensuite* `type(2//3)` *et Entrée. Qu'observez-vous ?*
6. *Tapez* `x =2` *puis Entrée.*
7. *Pour voir la valeur de* `x`*, vous pouvez tapez* `x` *puis Entrée.*
8. *Faîtes* `x = x + 1`*. Quelle est la nouvelle valeur de* `x` *?*
9. *Faîtes* `x = x * x`*. Quelle est la nouvelle valeur de* `x` *? Quel est le type de* `x` *?*

□

Exercice 6 (Évaluation d'expression, ⋆)
Utilisez l'interpréteur pour savoir comment les expression suivantes sont évaluées par `Python3`.

```
6 * 7 + 3
6 * (7 + 3)
45 // 7
3 * 7 // 4
(3 * 7) // 4
(45 // 7) * 7 + 45 (1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10 - 11 + 12 - 13) // (1 - 2
    + 3 - 4 + 5 - 6 + 7 - 8 + 9 - 10 + 11 - 12 + 13)
```

□

Exercice 7 (Première boucle, ⋆)

1. *Tapez le code suivant dans l'interpréteur (**Attention :** La deuxième ligne il faut utiliser Tab pour faire l'espace d'indentation et ne pas oubliez de faire Entrée à la fin.)*

```
for i in range (0, 100, 1) :
  print (i)
```

2. *Qu'observez-vous ?*
3. *Quittez l'interpréteur.*

□

Dans ces TPs, l'interpréteur sera utilisé comme aide, mais les programmes que nous vous demanderons d'écrire devront être dans des fichiers. Un fichier contenant un programme `Python` se finit par l'extension `.py`. Pour exécuter avec `Python` un programme `programme.py` se trouvant dans un répertoire `rep`. Il suffit d'aller dans le répertoire `rep` et de faire `python3 programme.py`. Pour écrire les programmes, nous utiliserons Emacs.

Exercice 8 (Premier programme, ⋆)

1. *Créez un fichier* `monPremierProgramme.py`*, dans le répertoire* `IP1-Python/TP1` *qui contiendra le code suivant :*

```
print (" Quel est votre prenom ? ")
s = input ()
print ("Bonjour  " + s)
```

2. *Sauvegardez votre programme, exécutez-le (en rentrant votre prénom quand cela vous est demandé).*

□

Exercice 9 (Exécuter un programme donné, ⋆)

1. *Récupérez sur* `Moodle`*, le programme du TP1* `nBonjour.py` *et sauvegardez-le dans le répertoire* `IP1-Python/TP1`.
2. *Exécutez-ce programme plusieurs fois. Que fait-il selon vous ?*
3. *Ouvrez ce programme avec Emacs et remplacez* `Bonjour` *par* `Salut`. *Sauvegardez et exécutez de nouveau ce programme.*

□