Travaux Dirigés & Pratiques

TP 1 – Environnement Linux pour la programmation

ENSICAEN 1A MC Octobre 2024

Eric Ziad-Forest

Sommaire

Auteurs :

- Jalal Fadili [jalal.fadili@ensicaen.fr]
- Philippe Lefebvre [philippe.lefebvre@ensicaen.fr]
- Alain Lebret [alain.lebret@ensicaen.fr]
- Regis Clouard [regis.clouard@ensicaen.fr]
- Eric Ziad-Forest [ziad@ensicaen.fr]

Enseignants des groupes de TD/TP

- Groupe 1: Oumaima Assou [oumaima.assou231@ensicaen.fr] Philippe Lefebvre [lefebvre@ensicaen.fr]
- Groupe 2: Julien Rabin [julien.rabin@ensicaen.fr]
- Groupe 3: Julien Rabin [julien.rabin@ensicaen.fr]
- Groupe 4: Eric Ziad-Forest [ziad@ensicaen.fr]

Labor omnia vincit improbus (un travail opiniâtre vient à bout de tout)

Virgile

PARTIE I

(pas plus d'1 heure en suivant les conseils de votre enseignant(e))

I - Objectif

Le but de ce TD/TP est de se familiariser avec l'environnement *Linux*. Votre environnement de travail informatique dans vos futurs entreprise ne sera pas toujours un environnement *Windows*.

Comme tout système d'exploitation, *Linux* est le logiciel permettant la gestion de la machine. Il offre des fonctionnalités pouvant être appelées depuis l'interface graphique ou par programme. *Linux* est une des versions libres de *Unix*. « Libre » voulant dire, libre de droit quant à son exploitation et libre de droit quant à la modification de son code source, tant que les modifications sont diffusées.

Dans les salles de TP de l'école, nous utilisons `Ubuntu`, qui est un ensemble de logiciels (appelée distribution) basé sur *Linux*.

Pour interagir avec le système d'exploitation il existe plusieurs méthodes :

- l'interface graphique dont les détails peuvent être retrouvés sur ce document :

http://ubuntu ;

- la programmation par le biais d'un langage comme le Python. C'est ce que nous verrons dans les TP suivants.
- l'utilisation de commandes, par le biais d'un interpréteur de commandes (`Shell`), qui est l'objet entre autre de ce TP.

Remarque
Ce premier travail, essentiellement de lecture et d'apprentissage, est assez long pour les débutants, nous les invitons néanmoins à le finir pour une meilleure pratique dans les futurs TP...

> **Il n'est pas demandé de compte-rendu pour ce TD/TP.**

II - Connexion à l'ordinateur

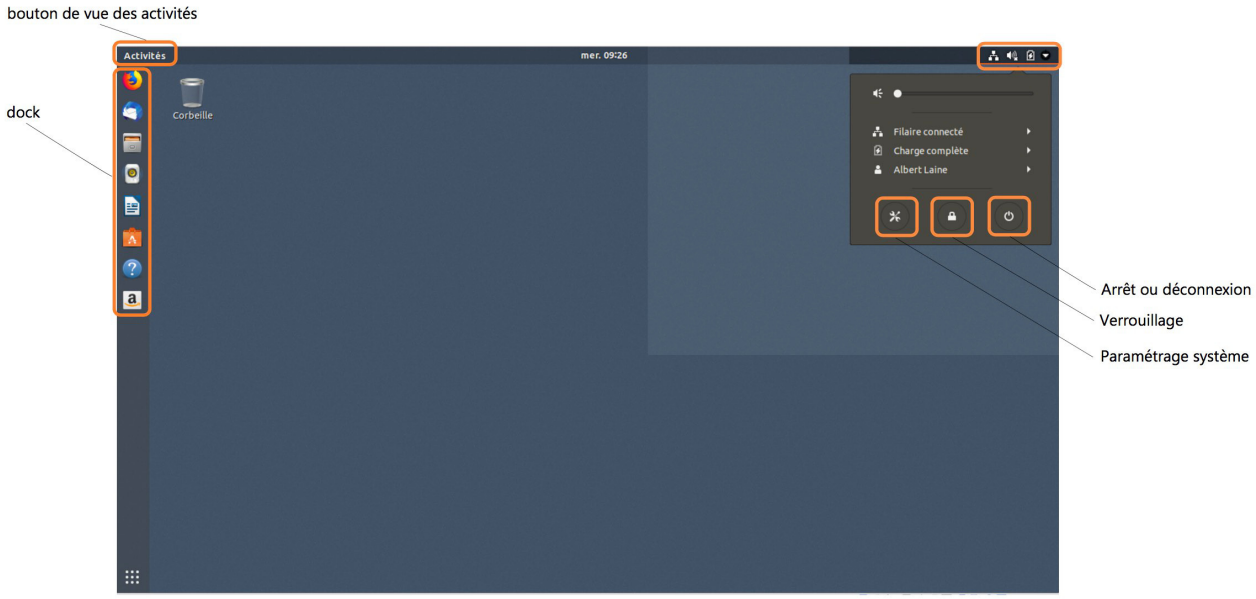
Les ordinateurs installés dans les salles machines de l'ENSICAEN permettent d'exécuter les systèmes d'exploitation Ms-Windows et Ubuntu Linux. Quelques secondes après avoir allumé l'ordinateur, un menu devrait s'afficher afin de vous permettre de choisir le système que vous souhaitez utiliser la sélection de l'un des deux étant réalisée au moyen des flèches et du clavier puis en appuyant sur Si vous n'agissez pas suffisamment rapidement, c'est le système d'exploitation Ms-Windows qui sera lancé par défaut. Sélectionnez le système Ubuntu.

II-1 Connexion

Une fois démarré, le système d'exploitation vous invite à renseigner votre identifiant et votre mot de passe. L'identifiant est un mot qui vous distingue des autres utilisateurs sur le système. En principe, celui que l'on vous a attribué devrait être constitué de l'initiale de votre prénom suivi de votre nom de famille, le tout en minuscules. Le mot de passe, quant à lui, garantira que vous serez le seul à pouvoir accéder à vos données. Après la procédure de connexion, votre session s'ouvre et vous avez alors accès à votre espace de travail ainsi qu'à vos données. Il est à noter qu'une session est personnelle et n'interfère pas avec celles des autres utilisateurs éventuellement connectés.

II-2 Verrouillage, déconnexion, arrêt Verrouillage

Une fois votre session ouverte, vous pouvez la verrouiller en cliquant sur le menu qui se trouve à droite de la barre supérieure de votre espace de travail (on parlera du bureau), puis en sélectionnant le bouton de verrouillage comme l'indique la figure ci-dessous.



Une fois votre session verrouillée, vous devez appuyer sur une touche quelconque pour la déverrouiller. Il est alors nécessaire de vous identifier de nouveau pour retrouver votre espace de travail tel que vous l'aviez laissé.

II-3 Fonctionnalités utiles

Cette section présente quelques fonctionnalités utiles des gestionnaires de fenêtres Unix.

Lancement d'application

Le Dock

La barre de lancement rapide appelée aussi «`dock`» (*dash en anglais*) est placée par défaut à gauche du bureau. Elle contient vos applications favorites. Vous voyez très certainement Firefox (un navigateur, le gestionnaire de fichier,...

Les applications

Dans la barre supérieure du bureau, on trouve sur la gauche le bouton `Activités` qui permet d'accéder à la vue des activités (*applications ouvertes sur le bureau*), mais aussi de rechercher des applications que l'on

souhaiterait lancer.

Exercice 1

1. Cliquez sur le bouton `Activités`, puis dans le champ de recherche qui s'ouvre, entrez le nom : `ter` et cliquez sur l'icône correspondante au `Terminal` qui apparaît en dessous.

Remarquez que l'icône du `Terminal` apparaît dans le `dock` une fois l'application ouverte. En cliquant sur l'icône avec le bouton droit de la souris, on fait apparaître un menu contextuel qui permet d'ajouter l'application au `dock`.

2. Ajoutez l'application au `dock`
3. Fermer l'aplication.

Remarque
Il est aussi possible de lancer des applications à l'aide de la combinaison de touches `Alt+F2` On obtient une fenêtre avec une invite dans laquelle on peut alors entrer le nom d'une application pour peu que l'on connaisse précisément celui-ci.

Exercice 2

Lancer le navigateur `Firefox` (tapez: **firefox**)
Puis aller sur Moodle [https://foad.ensicaen.fr/] et ouvrez ce document TP1 .

Souris et copier-coller

La plupart des applications Unix graphiques acceptent un copier-coller identique à celui qui existe sous Ms-Windows:

1. Sélectionnez à la souris ce que vous voulez copier.
2. Appuyez sur `Ctrl+[C]` , ou faites «Copier» dans le menu de l'application.
3. Placez la souris à l'endroit où vous souhaitez coller.
4. Cliquez avec le bouton gauche de la souris afin de mettre le curseur de l'application à l'endroit souhaité.
5. Entrez la combinaison de touches `Ctrl+V` ou faites« Coller» à l'aide du menu de l'application.

Sous Unix, le copier-coller avec la souris peut aussi s'obtenir avec le bouton du milieu ou la molette:

1. Sélectionnez à la souris ce que vous voulez copier.
2. Placez la souris à l'endroit où vous souhaitez coller.
3. Cliquez avec le bouton du milieu de la souris ou la molette.

Exercice 4

1. Lancez `Firefox` ainsi que l'éditeur de texte `geany`
2. Essayez de `copier/coller` une zone de texte depuis une page web `Firefox` vers `geany`, en utilisant les deux méthodes précédentes.

Gestion des fenêtres

Certaines touches ou combinaisons de touches permettent de modifier le comportement des interactions de la souris avec les applications ou avec le gestionnaire de fenêtres.

Exercice 5

1. Déplacez une fenêtre en maintenant enfoncée la touche **Super** `win Key` (deuxième touche à gauche de la barre d'espace), et en cliquant avec le bouton gauche n'importe où sur la fenêtre (pas uniquement sur le bandeau).
2. Redimensionnez une fenêtre en appuyant sur `win Key` et le bouton du milieu (ou la molette), ce sans avoir besoin d'attraper un des coins.
3. Maximiser une fenêtre en double-cliquant sur son bandeau, ou bien en faisant glisser ce bandeau vers le haut de l'écran.
4. Passer d'une fenêtre ouverte à l'autre en appuyant sur les touches `Alt` et $\to$ (Tab en haut à gauche)

Il existe encore de nombreuses manipulations possibles. Nous vous renvoyons aux documentations de Gnome ainsi qu'à la documentation francophone dUbuntu pour les découvrir (voir en fin de document).

Bureaux virtuels

Les bureaux virtuels (workspace en anglais) permettent de grouper les fenêtres et de pouvoir passer facilement d'un groupe de fenêtres à l'autre. Par exemple, il est possible d'utiliser un bureau pour naviguer sur Internet et lire son courriel, et un autre bureau pour développer etc. Sous Gnome (environement graphique Ubuntu), les différents bureaux virtuels sont accessibles en cliquant sur le bouton `Activités`. Les bureaux apparaissent alors en miniature à droite de l'écran. Par défaut, le nombre de bureaux est fixé à deux. La sélection d'un bureau en particulier est réalisable à l'aide des raccourcis clavier `Ctrl` +`Alt`+ $ \uparrow ou \downarrow$. Afin de déplacer une fenêtre d'un bureau à l'autre, il suffit de réaliser un clic droit sur le bandeau de la fenêtre qui fait alors apparaître un menu contextuel.

Exercice 6
Essayez d'ouvrir plusieurs applications et placez-les dans des bureaux différents.

III - Le shell

III - 1 Terminal et interpréteur de commandes

Un des principes d'Unix consiste à avoir un programme pour chaque tâche de manière à ce que chaque tâche soit réalisée au mieux. Ces programmes sont appelés des« commandes». Afin d'exécuter ces commandes, il faut utiliser un programme spécifique appelé interpréteur de commande (`shell`). L'interpréteur de commande présente une interface textuelle qui lit, interprète et exécute les commandes entrées au clavier. Concrètement, l'interpréteur de commande apparaît dans une fenêtre qui affiche une invite de commande (début de ligne non modifiable à la suite duquel vous entrez vos commandes). L'interpréteur de commande s'exécute à l'intérieur d'une application que l'on appelle `terminal` et exécute vos ordres.

III - 2 Première commande

Le shell est le nom donné au programme qui interprète les commandes écrites par un utilisateur, puis appelle les fonctionnalités adéquates du système d'exploitation (ou tout autre programme). Une commande est composée d'un *nom et d'arguments* permettant de préciser la commande. Le nom et les arguments sont séparés par des espaces. Par exemple, dans

```
ls -l
```

`ls` est la commande, `-l` est le premier argument. Souvent, les arguments commençant par un tiret, comme

dans l'exemple `-l`. Les arguments sont des options permettant de modifier l'exécution de la commande.

Lors de l'exécution d'un programme, celui-ci utilise des entrées et produit des sorties normales ou des erreurs. Par défaut, l'entrée est le clavier et la sortie normale et les erreurs sont produites sur l'écran.

Dans ce qui suit, chaque fois que vous verrez des lignes commençant par un dollar et écrite en police fixe, cela voudra dire qu'il faut taper la commande. Par exemple :

```
$ ls
```

Pour taper des commandes, vous devez au préalable ouvrir un `terminal` (appelé aussi console). Cela se fait en cliquant sur l'icône d'un « écran noir ». Si celui-ci n'est pas disponible, vous pouvez appuyer simultanément sur les touches `Ctrl`+`[Alt]`+`[t]`. Une fois la fenêtre du `terminal` entrez les 1ère commandes suivante:

Exercice 7

1. `ls` (*__list__*) suivi de la touche « Entrée ». Que constatez-vous?
2. Maintenant tapez :

```
$ lzs
```

un message d'erreur devrait apparaître à l'écran car le shell n'a pas trouvé la commande.

Peut-être vous demandez vous à quoi cela peut bien servir de taper des commandes alors qu'il existe, à l'instar de Windows, une **I**nterface **H**omme-**M**achine (**IHM**) permettant d'effectuer la plupart des opérations en quelques clics?
Le `shell` permet de coupler les commandes entre elles et d'exécuter des actions complexes qui ne sont pas possibles par de simples clics. Aussi, Linux est un système utilisé dans des box internet, des robots ... Sur de tels dispositifs, il n'y a pas d'écran et la seule solution pour communiquer avec `Linux` est d'utiliser le `shell` par l'intermédiaire d'un `terminal` en mode texte.

La suite de ce texte est une version adaptée du document suivant : Documentation d'Ubuntu

Dernier argument et non des moidres.

Les commandes emboités suivantes :

```
find /chemin/du/repertoire -type f -exec grep -l 'mot_a_rechercher' {} + | xargs du -h
| sort -rh | zip fichiers_recherches.zip -@
```

Permettent de réaliser les actions suivantes, irréalisable avec le gestionnaire de fichier et du clique souris...

Rechercher tous les fichiers dans un repertoire contenant un mot spécifique, puis les trier par taille décroissante et enfin créer une archive ZIP contenant dans l'ordre tous ces fichiers.

Pour les plus avancés en Python, cela ce ferait également avec le script python suivant: (Choisissez votre camp !)

In [ ]:

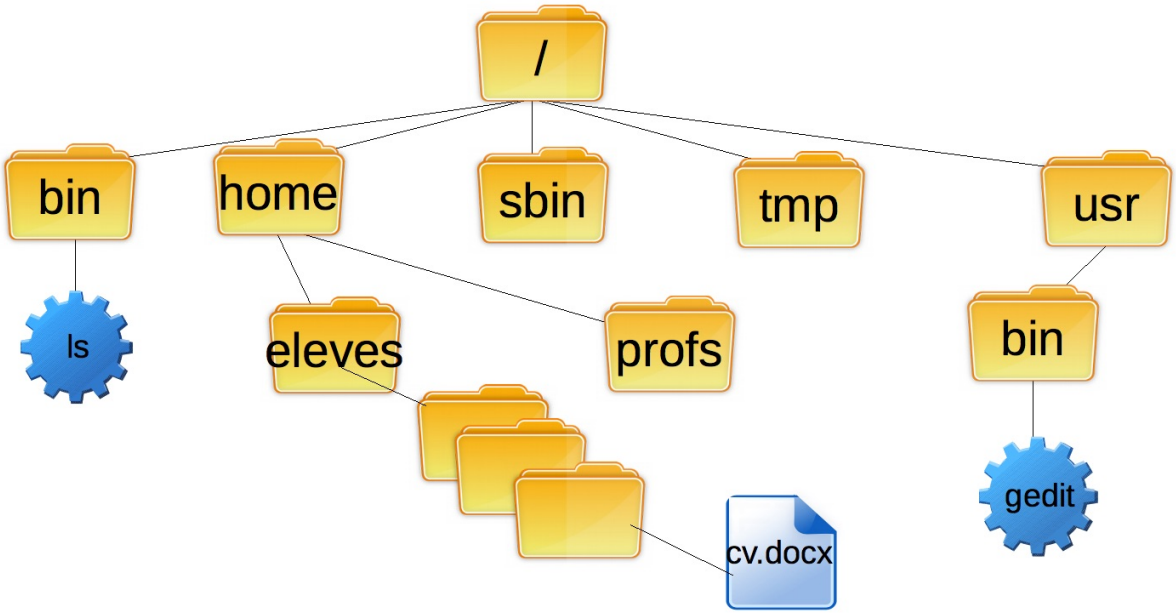
```
import os
import zipfile

# Définir le chemin du répertoire de recherche et le mot à rechercher
```

```
repertoire = '/chemin/du/repertoire'
mot_a_rechercher = 'mot_a_rechercher'

# Créer un répertoire temporaire pour stocker les fichiers trouvés
repertoire_temporaire = '/chemin/du/repertoire_temporaire'
os.makedirs(repertoire_temporaire, exist_ok=True)

# Recherche de fichiers contenant le mot spécifié et copie dans le répertoire
temporaire
for root, dirs, files in os.walk(repertoire):
    for file in files:
        file_path = os.path.join(root, file)
        with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
            if mot_a_rechercher in f.read():
                # Copie manuelle du fichier
                with open(os.path.join(repertoire_temporaire, file), 'wb') as
dest_file:
                    with open(file_path, 'rb') as src_file:
                        dest_file.write(src_file.read())

# Trier les fichiers par taille décroissante
fichiers_trouves = [os.path.join(repertoire_temporaire, file) for file in
os.listdir(repertoire_temporaire)]
fichiers_trouves.sort(key=os.path.getsize, reverse=True)

# Création de l'archive ZIP
with zipfile.ZipFile('fichiers_recherches.zip', 'w', zipfile.ZIP_DEFLATED) as archive:
    for fichier in fichiers_trouves:
        archive.write(fichier, os.path.basename(fichier))

# Suppression des fichiers temporaires
for fichier in fichiers_trouves:
    os.remove(fichier)

# Suppression du répertoire temporaire
os.rmdir(repertoire_temporaire)

# Suppression du répertoire source si nécessaire
os.rmdir(repertoire)
```

III - 3 Le système de fichiers

Comme *Windows*, *Linux* organise ses fichiers sur le disque dur par un système de répertoires. Le répertoire contenant tous les autres est appelé `/` ou répertoire racine.

Voici un exemple simplifié du système de fichier *Linux* :



`bin` : contient les commandes *Unix* de base ;
`home` : contient les répertoires des différents utilisateurs ;
`sbin` : contient les commandes *Unix* de base pour l'administration du système ;
`tmp` : contient des fichiers créés temporairement ;
`usr` : contient des commandes et des répertoires optionnels au fonctionnement de *Unix*.

Chemins relatifs / absolus et raccourcis

Lors de l'exécution d'un programme, il lui est attribué, par le système Linux, un répertoire de travail (**working directory**). Nous verrons comment le changer. Un fichier peut être référencé de manière **relative** , par rapport au répertoire de travail, ou de manière **absolue** par rapport à la racine. Par exemple,

- pour accéder au programme gedit de manière relative à partir du répertoire /usr :

```
bin/geany
```

- pour y accéder de manière absolue quel que soit le répertoire de travail :

```
/usr/bin/geany
```

Le premier caractère de la référence permet d'effectuer la distinction : les références absolues commencent toujours par le caractère `/` , alors que les références relatives commencent par tout autre caractère valide pour un nom de fichier.

Le `shell` est dit « *sensible à la casse* ».
Ce qui signifie que le répertoire «toTo» n'est pas le même que le répertoire «toto».
Les caractères spéciaux **`~`**, **`.`** et **`..`** ont des significations particulières :
~ (*tilde*) : **répertoire personnel** de l'utilisateur ;
`.` : **répertoire de travail** ;
`..` : **répertoire parent**.
Ils permettent tous les trois de simplifier l'expression de références absolues.
De plus pour continuer sur les `jokers` **`?`** désigne un unique caractère
`[]` correspond à une liste de caractères précisés entre les crochets;
`-` désigne un intervalle de caractères

Commandes de manipulation de fichiers et répertoires :

Exercice 8.
Essayez les 4 premières commandes commandes suivantes dans le 'terminal'.

`pwd` : (**p** ath of **w** orking **d** irectory) affiche le répertoire de travail dans lequel les commandes tapées sont exécutées.
`ls` : **lis**te le contenu du répertoire de travail.
`ls -l` : liste détaillée du contenu d'un répertoire (idem **ll)**.
`ls -al` : liste détaillée du contenu d'un répertoire y compris les fichiers cachés.
`cd` *nomDeRep* : (**c**hange **d**irectory) change le répertoire courant pour se placer dans le répertoire *nomDeRep*.
`cd ..` : remonte au répertoire parent
`cd` : sans paramètre, se place dans le répertoire personnel

`mkdir` *nomDeRep* : crée un répertoire nommé *nomDeRep* (**m**a**k**e **dir**ectory)
`rmdir` *nomDeRep* : efface le répertoire *nomDeRep* si vide (**r**e**m**ove **dir**ectory)
`rm` *nomFich* : efface un fichier *nomFich* (**r**e**m**ove).

Attention
La commande `rm` ne met pas à la poubelle. L'effacement est **définitif**!

`rm -i` *.c : efface tous les fichiers dont l'extension est `.c` # voir man de rm**
`mv` *toto titi* : renomme le fichier *toto* en *titi* (**m**o**v**e)
`mv /tmp/toto ~` : déplace le fichier *toto* de `/tmp` au répertoire personnel **`cp`** *toto titi* : copie le fichier « toto » et nomme la copie *titi*
`more toto.txt` : affiche page par page le fichier *toto.txt*.

l'appui sur `q` quitte l'appui sur `espace` affiche la page d'après l'appui sur `/` suivi d'un mot permet de faire une recherche du mot dans le fichier...

Pour en savoir plus, vous pouvez utiliser le manuel en ligne :

```
# pour se documenter sur la commande ls (# commentaire)
man ls
```

tapez `q` pour quitter_

Remarque
D'autres raccourcis claviers sous linux sont **extrêmement utiles**.

- **rappel de commandes linux**

Les flèches $\uparrow ou \downarrow$ permettent de naviguer dans les commandes déjà entrées.

- **complétion automatique**.

La touche `tab` permet de compléter un nom de commande ou un nom de fichier automatiquement. S'il y a ambiguïté, un deuxième appui propose la liste des possibles.

- **arrêt en cas de problème**.

La combinaisons `ctrl+c` arrête une commande en cours.

Exercice 9 Création d'un dossier de TP.

1. Regarder quel est votre dossier de travail. (**pwd**)
2. Lister les dossiers du répertoire courrant (**ls**)
3. Déplacer vous dans le dossier */Document* (**cd Doc** # complétion automatique (*tab*)
4. Créer un répertoire *InfoPython*
5. Dans ce répertoire créer deux répertoires *tp1* et *tp*
6. Comment afficher les répertoires dans l'ordre chronologique de création avec **ls** ? (lire le *__man__* de la commande **ls**)
7. Supprimer le répertoire *tp*

__Vous pourrez regarder dans le gestionnaire de fichier le résultat de vos manipulations__

III - 4 Les éditeurs de fichiers textes

Ils servent à créer des fichiers textes comme le logiciel Notepad sous windows. Il en existe de nombreux sous `linux` : `vi, emacs, geany, kedit, geany`... Pour ce TP nous utiliserons `geany` qui n'offre pas toutes les possibilités de `VIm`ou d'`emacs` mais qui en revanche est très simple d'utilisation. Pour le lancer :

```
$ geany fichier.txt &
```

- l'espeluette **&** est important car elle rend le `terminal` disponible pour entrer de nouvelles commandes!

Exercice 10

1. Quelle(s) différence(s) faites-vous entre un éditeur et un traitement de texte?
2. Lancer `geany` une fois *sans* **&** à la fin et essayez d'entrer la commande `ls`.
3. Fermez `geany` lancer-le *avec* **&** à la fin et essayez d'entrer la commande `ls`.

PARTIE 2

IV - Python

IV - 1 Introduction

Dans cette introduction, nous rappelons ici plusieurs points qui vous seront utiles :

1. Comme, il vous sera demandé dans un premier temps, d'entrer les instructions à l'interpréteur `Python`, notez ce que vous constatez sur les résultats obtenus et servez-vous-en pour répondre aux questions, voir résoudre les problèmes proposés par la suite.
2. Aucun des exercices suivant ne nécessite l'utilisation de fonctions.
3. Les exercices suivants ont pour vocation de vous familiariser avec **l'interpréteur** de Python et avec les commandes les plus élémentaires, n'hésitez pas à vous reporter au cours disponible sur Moodle.

J'insiste une nouvelle fois, nous utiliserons pour cette partie uniquement l'interpréteur de `Python`, qui est déjà par lui-même et comme

vous allez pouvoir en juger, un outil fort puissant.

IV - 2 Lancement de l'interpreteur Python

Dans le `terminal` ouvrez un onglet supplémentaires (touche `Ctrl+Maj+t`) puis lancez l'interpréteur `Python` en tapant tout simplement dans celui-ci:

```
python
```

Pour quitter l'interpréteur `Python` taper `Ctrl+d`

> **Exercice 11**
> Commencer par calculer quelques valeurs : 2+2, 'Tin'+'tin' , 2**1000

... Vous êtes prêt maintenant pour la suite ...

IV - 3 Premières instructions

Compréhension d'instructions et premières astuces

Dans les sous-chapitres de cette section, entrez les instructions demandées afin d'expliquer, par des commentaires sur une feuille ou un cahier, les résultats obtenus. Par la suite, certaines instructions et astuces seront utilisés. Revenez-y si besoin!

Donc, toutes les séquences d'instructions suivantes sont à entrer dans l'interpréteur (lui même dans le `terminal`) et sont à analyser.

Les types numériques

Utilisation du mot clé `type`

```
type(1)
type(1.0)
type(11111111111111111111)
s=1+1j
type(s)
type(1e10)
```

Conversion des types numériques

- `int()` pour convertir en entier;
- `float()` pour convertir en flottant;
- `complex()` pour convertir en complexe;
- `str()` pour convertir sous forme de chaîne de caractères

> **Exercice 12**
>
> 1. Essayer de convertir différent type de nombre pour en voir les effets.
> 2. Que vaut str(123)+str(321)? Pourquoi?

A propos des flottants

Utilisation de la fonction `print` (mot clé)

```
x=2**(1/2.0) # Racine carré de 2!
x
```

```
print ("%.50f" %x)
print ("%.80f" %x)
```

Exercice 13

1. Que signifie %.50f ?
2. Faire calculer à Python 1+1+1-3 puis 0.1+0.1+0.1-0.3
3. Comment expliquez-vous la différence?

Remarque. Ainsi, avant d'utiliser les résultats d'un programme informatique > donnés sous forme de "floats" , mieux vaut réfléchir à comment "pense" l'ordinateur ... Nous y reviendrons plus tard._

Calcul avec des grands nombres

Utilisation des puissances. `**`

```
457**15
457**231
(5**5)**5
7**7**7
```

Alors vos impressions?

Exercice 14 (*pour les plus avancés*)
Vous rappelez-vous, comment obtenir le nombre de chiffres des résultats de ces calculs?

Calcul avec des nombres complexes

```
z1=complex(2,1)
z=2+1j # j représente imaginaire pur i, ie j**2=-1
z*z
z**2
z*(3+4j)
abs(1+1j)
(1+1j).conjugate()
```

Pas besoin d'interface compliquée pour effectuer des calculs "complexes"!

Pour connaître d'autres fonctions et méthodes appliquées aux nombres complexes entrez :

```
dir(complex)
```

Remarque

Utiliser la commande `dir( )` avec le type dont vous souhaitez connaître les méthodes et fonctions.

Exercice 15
Vérifiez , sur au moins, un exemple non trivial (*par exemple 1-1j*) que l'inverse d'un nombre complexe est égale à son conjugué divisé par son module au carré!

Remarque

Nous venons d'utiliser déjà beaucoup de fonctionnalités mathématiques sans jamais avoir eu besoin

d’importer quoique ce soit!

Poursuivons...

Ne pas confondre : affectation (=) et égalité (==)

```
k=1+1+1;k
k==2
print('La valeur de k est: ',k)
```

Dans les 2 prochains sous-chapitres ci-dessous, nous continuons à utiliser la variable k.

Encore des calculs et des précisions sur les types...

(*__Pour les plus aguéris__*)

```
k+=k
k*=k
k=[k/5,k/0.5]
type(k)
```

Expliquer ici, pourquoi il y a une erreur, si l’on répète la commande précédente :

k=[k/5,k/0.5]

Et pour finir

```
h=4*"Na"+" Nere" ; h
type(h)
type('16')
type(int('16'))
```

PARTIE 3

S'il vous reste du temps, poursuivez vers d'autres commande `linux`

V - Approfondissement Linux

V - 1 Les droits

(*__peut-être sauté en 1er lecture__*)

`Unix` est un système multi utilisateurs. Les fichiers appartiennent donc à des utilisateurs. Les utilisateurs appartiennent également à des groupes. On peut donc donner l'accès en lecture **r**, en écriture **w** et/ou en exécution **x** à un utilisateur, à un groupe ou aux autres utilisateurs de la machine.
La machine utilise une base de données des utilisateurs hébergée sur un serveur. La commande `ls -l`

donne des informations sur les droits sous cette forme « rwxrwxrwx ». Par exemple :

```
ls -l
drwxr-xr--  7 toto bde            4096 oct.  14  2022 savDoc
-r--------  7 toto bde            4096 oct.  14  2022 mail.txt
```

Pour le répertoire *savDoc*,

- le propriétaire *toto* a les droits « rwx » c'est-à-dire tous les droits ;
- les utilisateurs du groupe *bde* ont les droits « r-x », c'est-à-dire tous sauf écriture ;
- les autres n'ont que les droits de lecture.
- Le premier caractère est la lettre « d » ce qui signifie que *savDoc* est un répertoire.

Le « x » signifie, dans le cas des répertoires, « cross », c'est-à-dire droits de traverser avec la commande « cd ».

- Pour le fichier « *mail.txt* » l'utilisateur *hubert* ne peut que lire le fichier. S'il veut l'effacer, il doit d'abord changer les droits.

-Pour changer les droits on utilise la commande `chmod` suivi de la valeur en octal des droits. **r** a le poids 4, **w** 2 et **x** 1.

```
chmod 741 mail.txt
```

donne les droits « rwxr----x » au fichier mail.txt.

> **Exercice 16**
> Changez les droits du répertoire *tp1* pour que seul le propriétaire ait les droits **rwx**.

V - 2 Les redirections

On l'a vu précédemment les commandes agissent sur des entrées (fichiers, clavier) et produisent des sorties (fichiers, écran). Le clavier et l'écran sont respectivement l'entrée standard (stdin) et la sortie standard (stdout). Mais on peut les rediriger.
Par exemple,

```
ls > toto.txt
```

crée le fichier *toto.txt* contenant le résultat de la commande `ls`.
Les redirections sont :

- > sortie redirigée vers un nouveau fichier
- >> sortie redirigée en ajout.

plus délicat:

- < entrée est prise à partir d'un fichier
- << `EOF` l'entrée reste « stdin » mais la commande se termine lorsqu'elle rencontre la ligne contenant les trois caractères `EOF` (**E**nd **O**f **F**ile).
- *commande1* **|** *commande2* Avec le caractère **|** (pipe), la sortie de *commande1* est redirigée sur l'entrée de *commande2*.

V - 3 D'autres commandes

Exercice 17 : Manipulation d'autre commande `linux`

Exécutez la suite de commandes suivantes et interprétez à chaque étape ce que vous observez :

Remarque
Ce qui suit le caractère # sont des commentaires à ne pas taper.
Ne pas taper $, il correspond au prompt de l'interpréteur `shell`

```
$ cd
$ mkdir temp; cd temp
$ ls -l ..
$ ls -l > fichier0.txt
$ cat fichier0.txt
$ cat > fichier1 << EOF
Mon texte 1
EOF
$ more fichier1
$ cat fichier
$ ls -al
$ nomVar="le monde"
$ echo Bonjour $nomVar
$ echo "Mon texte 2" > fichier2
$ cat * # Concaténation de tous les fichiers vers la
        # sortie standard.
$ cat fichier1 fichier2 > resultat # Concaténation et redirection.
$ more resultat
$ echo "separateur" | cat - fichier1 fichier2 >> resultat
# Concaténation et ajout.
# ici le tiret – en argument de cat siginifie entrée standard
$ more resultat
$ diff -w -y fichier1 resultat # compare les deux fichiers.
$ tail -n 2 resultat # Affiche la fin d’un fichier.
$ tail -c 4 resultat # Le caractère de fin de ligne compte
                     # pour un caractère.
$ hexdump -C resultat
$ head -1 resultat # Affiche le début d’un fichier.
$ head -2 < resultat
$ wc -l resultat # Compte les lignes, caractères, mots.
$ cat resultat | wc -l
$ wc -c resultat
$ wc -w resultat

$ cat > annuaire.txt << EOF
Nom1:Prenom1:num1:adresse1
Nom2:Prenom2:num2:adresse2
Nom3:Prenom3:num3:adresse3
EOF
$ cat annuaire.txt | cut -d":" -f 2 # Sélection de colonnes
                                    # (ici celle des prénoms).
$ phrase="toto titi tutu"
$ for i in $phrase ; do echo $i ; done
$ for i in `cat resultat` ; do echo $i ;done
```

```
# notez les quotes inversés (accent grave) qui exécutent
# la commande précédente.
```

Exercice 18
Écrire une commande qui écrit dans le fichier *resultat2.txt* tous les mots des 2 dernières lignes du fichier *resultat.txt* de l'exercice précédent.

Il n'y aura qu'un mot par ligne.

Comme quoi, pas besoin de tout programmer avec Python, le `shell` peu souvent suffire...

V - 4 Le filtre grep et les expressions régulières

Exercice 19.

1. Exécutez la suite de commandes suivantes et interprétez à chaque étape ce que vous observez :

```
$ grep " 1" resultat # ligne contenant la chaine " 1".
$ grep -n " 1" resultat # numérote les lignes.
$ grep -v sepa resultat # exclut les lignes avec "sepa".
$ grep "e.t" < resultat # . n’importe quel caractère.
$ grep "e.*t" resultat # * répète le caractère précédant
                       # n fois (avec n>=0).
$ grep "^t" resultat # lignes débutant par "t".
$ grep "1$" resultat # lignes finissant par "1".
$ grep "[a-z]$" resultat # lignes finissant par une lettre
                         # (donc pas un numéro).
$ grep -e "1" -e "2" resultat # lignes contenant 1 ou 2.```
> 2. Écrivez une ligne de commande qui affiche les mots contenant 2 lettres **a** et
pas de lettre **x**, et une autre ligne
qui compte le nombre de mots commençant par **T**.
```

V - 5 Commande find

La commande find recherche un fichier dans un dossier et ses sous-dossiers en appliquant l’expression donnée.

Exercice.

1. Essayez les commandes suivantes pour vous familiariser avec celle-ci :

```
$ find . -name "*.c" # recherche les fichiers dont le nom se termine ~par c.
$ find . -type d # recherche uniquement les réper- toires.
$ find . -name "*.c" -o -type d # -o pour faire un OU.
$ find . -name "*.c" -o -name "r*" #  se terminant par c OU commençant par un r.
$ find . -perm 755 # fichiers dont les droits sont positionnés  à 755.
$ find . -name "*.c" -exec ls -l {} \; # exécute la commmande « ls -l » sur chaque
fichier trouvé par  find.
```

Remarque

La dernière commande ci-dessus doit se terminer par un point virgule. Cependant, comme le `shell`

interprète le `;` comme un séparateur de commandes `shell`, il faut préfixer le `;` par un `\` afin qu'il ne soit pas interprété.
(\ est appelé **caractère d'échappement**.)

In [1]:

```
ls
```

```
TP1-Linux-Python-23.ipynb  images/
```

Pour aller plus loin:
http://www.linux-france.org/article/memo/memo.html