UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Outils Info. TP0: prise en main

Soyez attentif à la lecture du sujet.

Ce premier TP est très verbeux parce qu'il a pour but de vous apprendre ou réapprendre les bases de l'utilisation de la ligne de commande (disponible sous Linux et Mac, et installable sous Windows). Pour les instructions d'installation sur votre machine (si vous utilisez Windows), référez vous à la page du cours.

Aide : utilisation du poste de travail

Les TPs de programmation en salle machine se font sur des ordinateurs, dont le système d'exploitation est Linux. Chaque utilisateur est reconnu par un nom (ou *login*) et dispose d'un mot de passe.

Chaque utilisateur dispose d'une zone de stockage sur les disques durs pour y garder ses fichiers personnels. Cette zone de stockage est un répertoire (ou dossier) dans lequel l'utilisateur peut créer librement des sous répertoires et des fichiers. Les répertoires et sous répertoires forment une structure arborescente dans laquelle chaque fichier ou dossier a un parent unique.

Rappel : une notation est utilisée pour désigner un dossier ou fichier, par exemple `tp0/ex1/aide.txt` désigne un fichier `aide.txt` contenu dans un dossier `ex1` lui même contenu dans un dossier `tp0`. Le dossier parent est appelé « `..` », le dossier courant est toujours appelé « `.` ».

Aide : principales commandes utiles dans un terminal

Lorsque vous ouvrez un « terminal », vous êtes par défaut à la racine de votre système de fichier (votre `home`). De nombreuses commandes peuvent alors être exécutées, les plus fréquentes sont listées ci-dessous. Il peut être intéressant de lire ces descriptions, utiles pour l'exercice 1 et par la suite :

- pwd : (*print working directory*) connaître le répertoire courant
  - `pwd` : affiche le chemin du répertoire courant
- cd : (*change directory*) changer le répertoire courant
  - `cd tp0` : se déplace dans le répertoire `tp0`, s'il existe
  - `cd ..` : se déplace dans le répertoire parent du répertoire courant
  - `cd .` : se déplace dans le répertoire courant… inutile
  - `cd` : se déplace dans votre *home*
- mkdir : (*make directory*) créer un nouveau répertoire dans le répertoire courant
  - `mkdir tp0` : crée un répertoire `tp0` dans le répertoire courant
  - `mkdir tp0/ex1` : crée un répertoire `ex1` dans le répertoire `tp0`
  - `mkdir -p tp0/ex1` : idem, mais crée `tp0` si besoin
- rmdir : (*remove directory*) supprimer un répertoire (uniquement s'il est vide)
- rm : (*remove*) supprimer un ou des fichiers et/ou dossiers
  - `rm ancien.txt` : supprime le fichier `ancien.txt` du répertoire courant
  - `rm -r ancien` : DANGER ! supprime le dossier `ancien` et tous ses sous dossiers et fichiers
- ls : (*list*) lister les fichiers dans un répertoire
  - `ls` : liste les fichiers dans le répertoire courant
  - `ls tp0` : liste les fichiers dans le répertoire `tp0`
  - `ls -la` : (avec un `L` minuscule) liste avec détails des fichiers dans le répertoire courant (y compris les fichiers cachés)
- touch : créer un ou des fichiers vides
  - `touch films.txt` : crée un nouveau fichier `films.txt`, vide, dans le répertoire courant
- man : (*manual*) obtenir un manuel d'une commande
  - `man ls` : affiche le manuel de la commande `ls`
  - `man man` : affiche le manuel de la commande `man`

  - `man cd` : ¡ (cd n'est pas une commande normale)
- mv : (*move*) déplacer un ou des fichiers, renommer un fichier
  - `mv ancien.txt nouveau.txt` : renomme le fichier `ancien.txt` en `nouveau.txt`
  - `mv a.txt tp0/b.txt` : déplace le fichier `a.txt` dans le dossier `tp0` en le renommant en `b.txt`
- cp : (*copy*) copier un ou des fichiers et/ou répertoires
  - `cp fichier.txt sauvegarde.txt` : réalise une copie de `fichier.txt` et la nomme `sauvegarde.txt`
  - `cp -r modele nouveau` : réalise une copie récursive du dossier `modele` et la nomme `nouveau`
- find : rechercher des fichiers récursivement, et bien plus encore
  - `find` : liste les fichiers du répertoire courant et tous les sous répertoires
- cat : afficher le contenu d'un fichier
- less : consulter le contenu d'un fichier, appuyer sur « q » pour quitter
- history : afficher la liste des commandes exécutées

Pour la suite du semestre, comme demandé dans l'exercice 1, il est demandé de créer un répertoire pour chaque TP (e.g., tp0, tp1, tp2, …) et un sous répertoire pour chaque exercice (e.g., tp0/ex1, tp0/ex2, …).

Dans les exercices, certains points vous donnent simplement une commande à lancer. D'autres points demandent de trouver la commande à lancer. Notez les commandes qui permet de répondre à chaque question.

Exercice 1 – Manipulation de fichiers

Q1) Ouvrez un terminal à la racine de votre arborescence de fichier avec Activités>Applications>Terminal.

Q2) Créez un dossier `outils-info` .

Q3) Déplacez vous dans ce dossier.

Q4) Créez-y un dossier `tp0`.

Q5) Déplacez vous dans ce dossier.

Q6) Vérifiez à l'aide de `pwd` que le chemin courant se fini bien par `/outils-info/tp0`.

Tout le reste de ce TP se déroulera dans ce dossier qui vient d'être créé. Il est demandé de suivre cette *convention pour l'ensemble des TP* et de leur exercices. Nous allons maintenant utiliser d'autres commandes.

Q7) Dans le dossier `tp0`, créez trois fichiers (vides) nommés `a-lire.txt`, `programme.py` et `cr.txt` à l'aide de la commande `touch`.

Q8) Utilisez `ls` pour vérifier la création des fichiers.

Dans un vrai cas, ces fichiers pourraient être : un fichier de notes, un programme et un compte rendu. Nous allons maintenant re-organiser ces éléments.

Q9) Créez un répertoire `src` et déplacez-y le fichier `programme.py`.

Q10) Créez un répertoire `cr` et déplacez-y le fichier `cr.txt`, renommé en `compte-rendu.txt` .

Q11) Copiez le fichier `a-lire.txt` en tant que `readme.txt` .

Q12) Vérifier avec `ls` que vous avez la structure suivante (à partir de `tp0`) :

```
.
├── a-lire.txt
├── cr
│   └── compte-rendu.txt
├── readme.txt
└── src
    └── programme.py
```

Nous avons oublié le dossier `ex1` pour l'exercice 1… et finalement nous ne voulons plus de `a-lire.txt` .

Q13) Supprimez le fichier `a-lire.txt` .

Q14) Créez un répertoire `ex1` dans `tp0` .

Q15) Déplacez les fichiers et dossiers dans `ex1` .

Q16) Vérifiez avec `ls` que vous obtenez cette structure (à partir de votre « dossier maison »):

```
outils-info
└── tp0
    └── ex1
        ├── cr
        │   └── compte-rendu.txt
        ├── readme.txt
        └── src
            └── programme.py
```

Q17) Utilisez `find` pour réaliser cette même vérification.

Pour terminer, nous ne voulons qu'un seul fichier de compte rendu par TP:

Q18) Déplacez le fichier `compte-rendu.txt` dans le dossier `tp0`.

Q19) Supprimer le dossier `cr` qui est maintenant vide.

Q20) Utilisez `find` pour vérifiez cette structure finale (à partir de votre « dossier maison »):

```
outils-info
└── tp0
    ├── compte-rendu.txt
    └── ex1
        ├── readme.txt
        └── src
            └── programme.py
```

Exercice 2 – Le premier programme python

Cet exercice va vous amener à saisir votre premier programme avec un éditeur de texte et à l'exécuter à l'aide de l'interpréteur python (version 3). Pensez à *créer un dossier* `ex2` au bon endroit pour cet exercice.

Q21) Créez un fichier nommé `bonjour.py` .

Q22) Ouvrez ce fichier avec un éditeur de texte, par exemple en utilisant la commande : « `emacs bonjour.py &` »

Emacs est un éditeur un peu déroutant au départ mais très efficace et très configurable. Emacs s'ouvre alors pour l'édition du fichier `bonjour.py`.

Le « `&` » en fin de commande lance la commande en tâche de fond, laissant le terminal libre pour exécuter d'autres commandes. Si vous avez oublié le « `&` », vous pouvez taper « Ctrl-z » dans la fenêtre du Terminal pour mettre Emacs en pause, suivi de la commande « `bg` » pour le mettre en tâche de fond.

Q23) Saisissez le programme suivant (sans les numéros de ligne) et sauvez le fichier. Vous pouvez garder votre éditeur ouvert.

```
1 print("Hello !!!!")
2 print("==========")
```

Q24) Dans le terminal, tentez de lancer votre programme avec la commande : « `python3 bonjour.py` »

Lisez le texte possiblement retourné par l'interpréteur python. Si celui-ci signale des erreurs, vous avez peut être mal recopié le programme. Corrigez votre programme, sauvez le fichier et relancer l'interpréteur `python3` à nouveau.

Si l'exécution de votre programme s'est bien déroulée, l'exécution de « `python3 bonjour.py` » doit afficher un texte commençant par « Hello ».

Q25) Changer le message affiché dans le fichier `.py` et ré-exécutez le programme. Vérifiez que la sortie est celle attendue.

Exercice 3 – Quelques opérations arithmétiques

Pour cet exercice, travaillez encore une fois dans un nouveau dossier, `ex3`.

Q26) Copiez le fichier `bonjour.py` de l'exercice précédent et donnez lui un nouveau nom `calculs.py` .

Q27) Ouvrez le fichier et modifiez le pour obtenir la version suivante :

```
x = 1
y = 100
a = 0.2

res = x + a * y

print("Hello !!!!")
print("==========")

print("res = ", res)
```

Q28) Réalisez une copie de sauvegarde de ce fichier, appelez la par exemple `calculs.py.save`

Q29) Vérifiez que la copie a fonctionné de 3 façons : avec `ls`, `cat` et `less`

Q30) Exécutez votre programme à l'aide de l'interpréteur `python3`, corrigez éventuellement les erreurs reportées par l'interpréteur `python3`.

Q31) Expérimentez avec différentes expressions pour `res`, avec plus de variables, etc.

Exercice 4 (optionnel) – Droits d'accès aux fichiers (`chmod`)

Chaque fichier et dossier de l'arborescence a des droits d'accès. Cet exercice vous apprend à interpréter et manipuler ces droits d'accès.

Q32) Allez dans dossier `outils-info/tp0` .

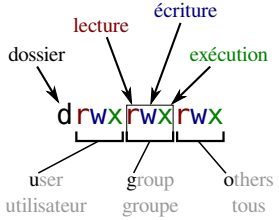
Q33) Copiez le dossier `ex1` en tant que `ex4` .

Q34) Utilisez `find` pour vérifier que la copie a fonctionné.

Q35) Déplacez vous dans `ex4` .

Q36) Utilisez `ls -l` (c'est un `L` minuscule) pour avoir une liste détaillée des fichiers.

Le premier bloc de chaque ligne est expliqué ci dessous. Certaines lettres sont remplacées par des tirets, indiquant que ce droit d'accès est refusé.



Nous allons maintenant changer les droits d'accès à l'aide de la commande dédiée nommée `chmod` .

Q37) Utilisez `chmod o-r readme.txt` pour empêcher les gens qui ne sont ni vous ni dans votre groupe d'accéder au fichier `readme.txt` .

Q38) Vérifiez avec `ls -l` que l'opération a eu l'effet attendu.

Q39) Redonnez les droits de lecture avec `chmod o+r readme.txt` et vérifiez.

Q40) Enlevez maintenant les droits à tous les autres que vous avec `chmod go-r readme.txt` et vérifiez.

Q41) Lancez `chmod g+rw,o+r readme.txt` et comprenez avec `ls -l` l'effet de cette combinaison.

L'attribut `x` (exécution) donne le droit d'exécuter un fichier, par exemple un programme. Dans le cas d'un dossier, l'interprétation est un peu différente…

Q42) Enlevez vous les droits en lecture sur le dossier `src` (utilisez `chmod` avec les bons paramètres).

Q43) Déplacez vous dans le dossier `src` .

Q44) Essayez de lister les fichiers qu'il contient, quelle est l'erreur obtenue dans ce cas ?

Q45) Essayez de créez un fichier (vide) appelé `haha`, cela fonctionne-t-il ?

Q46) Revenez dans le dossier parent avec `cd ..` (rappel : le terminal interprète « `..` » comme le parent).

Q47) Changer vos droits sur `src` pour obtenir `rw-` (lecture, écriture mais pas exécution)

Q48) Essayez de vous déplacer dans le dossier `src`, quelle est l'erreur ?

Q49) À quoi correspondent donc les droits en exécution sur un dossier ?

Q50) Redonnez les droits en exécution à `src` mais enlevez ceux en écriture.

Q51) Allez dans `src`, et listez son contenu. Quels fichiers contient-il ?

Q52) Toujours dans `src`, essayez de créer un fichier `hoho`. Que se passe-t-il ?

Q53) Résumez en quelques points le rôle des droits `r`, `w` et `x` dans le cas d'un dossier.

Exercice 5 – Tester Qtido

Pour cet exercice, travaillez encore une fois dans un nouveau dossier, `ex5`.

Q54) Sur le site du cours téléchargez et enregistrez les fichiers dont les noms ressemblent à `qtido-.....zip` et `qtido-....-exemples.zip`.

Q55) Essayer d'identifier à quel endroit (dans quel dossier) votre navigateur enregistre les fichiers, par exemple `~/Téléchargements` (le dossier Téléchargement dans votre compte utilisateur).

Q56) En étant sûr que vous êtes dans le dossier de l'ex5, pour chacun des fichiers zip, lancez la commande `unzip` avec comme paramètre le fichier en question. Par exemple, `unzip ~/Téléchargements/qtido-........zip`.

Q57) Vérifier avec `ls` que les fichiers ont été extraits.

Q58) À l'aide de `python3` et `emacs`, testez les programmes d'exemples fournis (`minitest.py`, `test-ex1.py`, …) et lisez leurs code source.