Introduction à la ligne de commande sous linux et à l'utilisation de git

Arnaud Pêcher

04/09/2019

Ligne de commande

La ligne de commande n'est pas requise pour faire du développement logiciel. Toutes les opérations qui sont décrites dans ce document peuvent être faites en utilisant des outils graphiques plus "conviviaux". Néanmoins, pour faire du développement logiciel, il est utile de maîtriser les notions de base, car cela permet souvent d'être plus efficace. De plus, lorsqu'on aborde des technologies qui sont plus avancées ou peu matures, la ligne de commande est parfois incontournable. Par exemple, le déploiement d'applications sur un casque de réalité virtuelle "Oculus Go/Quest" nécessite d'exploiter les utilitaires "bas niveau" (adb) d'Android Studio, via la ligne de commande ...

Pour utiliser la ligne de commande, on utilise un "terminal": il existe plusieurs logiciels de ce type. Un des plus commodes est le terminal `guake`.

Les fichiers

Les répertoires (ou dossiers)

Un répertoire contient un ensemble de fichiers. Il peut également contenir d'autres répertoires, qui sont appelés sous-répertoires. Les répertoires (et par extension les fichiers) sont donc organisés sous la forme d'un arbre, possédant une racine.

- le répertoire racine : /

La répertoire racine de l'arbre est dénoté "/" .

- le répertoire personnel : ~

Chaque utilisateur possède un répertoire personnel, dénoté "~" ou encore `$HOME`.

- le répertoire courant : .

On peut se déplacer de répertoire en répertoire: le répertoire courant (actuel) est dénoté "." .

- le répertoire parent : ..

A l'exception du répertoire racine /, tout répertoire possède un répertoire parent, dénoté ".."

- le répertoire temporaire : /tmp

Ce répertoire sert à stocker des fichiers qui sont destinés à être utilisés pendant un laps de temps court. Ils peuvent être notamment effacés lors d'un redémarrage du système.

- le répertoire des téléchargements : ~/Téléchargements ou ~/downloads
- le répertoire des documents : ~/Documents
- le répertoire du bureau : ~/Bureau ou ~/Desktop

Manipulation des répertoires

Voici la liste des pirncipales commandes permettant de manipuler les répertoires. Pour chaque commande, un manuel est accessible via la commande `man`.

- voir les fichiers d'un répertoire: ls

La commande `ls` (list) affiche la liste des fichiers et des sous-répertoires du répertoire courant.

- changer de répertoire: cd

La commande `cd` (change directory) permet de changer de répertoire: `cd repertoire` : permet d'aller dans le répertoire désigné.

Exemples : `cd ..`, `cd .`, `cd`, `cd -`

- créer un nouveau répertoire: mkdir

La commande `mkdir` (make directory) permet de créer un sous-répertoire du répertoire courant.

Manipulation de fichiers

Pour modifier le contenu d'un fichier texte, de nombreux éditeurs de texte sont disponibles: `gedit`, `nano`, `code` (visual-studio) ...

Conseil: éviter d'utiliser des caractères accentués ou des espaces dans les noms de fichier.

- créer un nouveau fichier (vide): touch

La commande `touch` permet de créer un nouveau fichier. Exemple:

```
touch adresses.txt
```

crée un fichier dont le nom est `adresses.txt`. L'extension du fichier (`txt`) sert à indiquer à l'utilisateur la nature des données du fichier, mais contrairement à `windows`, le système d'exploitation n'en tient pas compte (pour connaître de manière fiable le type du fichier, il vaut mieux utiliser la commande `file` plutôt que l'extension du fichier).

- déplacer/renommer un fichier: mv

La commande `mv` (move) permet de déplacer un fichier d'un dossier à un autre et/ou de le renommer.

Ainsi

```
mv x/a.txt y/b.txt
```

déplace le fichier `a.txt` du répertoire `x` dans le répertoire `y` en le renommant `b.txt`.

- supprimer un fichier: rm

La commande `rm` permet de supprimer un fichier.

- copier un ficher: cp

La commande `cp` (copy) permet de copier un fichier. La copie créée est indépendante du fichier original: une modification du contenu de l'un n'a pas d'impact sur le contenu de l'autre (contrairement aux liens symboliques mais c'est une autre histoire).

Ainsi

```
cp x/a.txt y/b.txt
```

copie `a.txt` du répertoire `x` en tant que fichier `b.txt` du répertoire `y`.

Quelques exercices

Exercice 1

Réaliser la suite de manipulations suivantes:

1. créer un répertoire `exercices`
2. créer un sous-répertoire `exercice1` du répertoire `exercices`
3. se positionner dans le sous-répertoire `exercice1`
4. créer un fichier vide de nom `exo1.txt`
5. modifier son contenu à l'aide d'un éditeur de texte de telle sorte que le fichier contienne

```
Viva la Vida !
```

6. renommer le fichier avec le nom `exo1-v2.txt`
7. créer un sous-répertoire `archives` du répertoire `exercices`
8. recopier le fichier `exo1-v2.txt` dans `archives`
9. supprimer le fichier `exo1-v2.txt` du répertoire `exercice1`
10. supprimer le répertoire `exercice1`

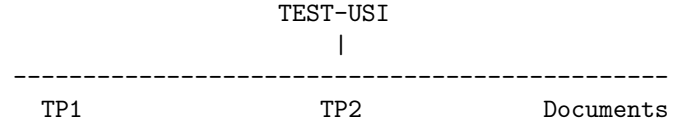
(Indication : utiliser l'option `-r` de la commande `rm`)

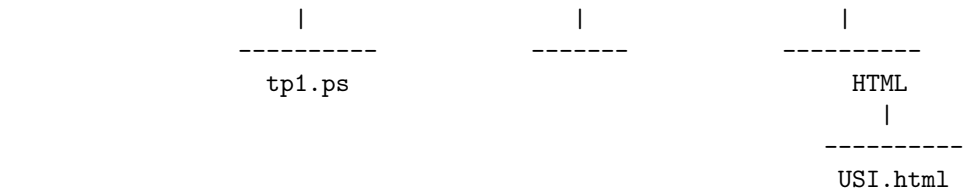
Exercice 2

L'objectif est de fabriquer l'arborescence ci-dessous dans votre répertoire d'accueil :

- Quelle est la série de commandes à taper pour construire cette arborescence, en supposant qu'une copie des fichiers `tp1.ps` et `USI.html` est dans `/net/exemples/USI` ?
- Proposez une version dans laquelle n'apparaît pas la commande `cd` et une autre dans laquelle la commande `cd` est autorisée.

(Indication : afin de réduire le nombre de commandes, utiliser l'option `-p` de `mkdir`)





On voudrait maintenant effacer la sous-arborescence Documents. Comment procéder ?

Quelques trucs et astuces à savoir

Pour arrêter un programme:

- Utiliser la combinaison des deux touches du clavier (dans l'ordre) `<Ctrl-C>` afin d'arrêter (to Cancel) à partir du terminal un programme en exécution;
- Afin d'arrêter une application graphique (X) qui refuse de s'arrêter de manière classique (fermeture de la fenêtre inopérente), utiliser la commande `xkill` pour forcer le programme à se terminer.

Pour saisir plus rapidement des commandes:

- la *complétion* offre gain de temps et évite les erreurs de saisie. Principe : `<TAB>` permet de compléter les noms de fichiers, de répertoires ou de commandes au moment de leur frappe au clavier.
- *historique des commandes*: la touche flèche verticale permet de rappeler une commande précédemment tapée. Voir aussi la combinaison des deux touches `<Ctrl-R>`

Gestion de version / développement collaboratif: git

`git` est un logiciel de gestion de versions décentralisé. Vous allez l'utiliser tout au long de l'année:

- pour accéder aux dernières versions des ressources associées à un cours;
- pour réaliser des projets à plusieurs.

Au sein de l'Université de Bordeaux, nous utilisons une plateforme pédagogique à distance (Moodle), qui sera également exploitée en parallèle ou dans certains cours.

Le but de cette séance est de prendre connaissance du minimum requis pour pouvoir travailler avec `git`.

Cycle de développement en utilisant git

Le cycle normal d'un cycle de travail est le suivant:

1. récupérer la dernière version des données stockées sur le serveur: `git pull`
2. travailler sur sa machine locale sur ces données (modifications de fichiers, ajouts de nouveaux fichiers)
3. enregistrer les nouvelles versions dans la liste des données à envoyer au serveur: `git add`
4. grouper les modifications en attente en vue d'un envoi au serveur: `git commit -m "blabla"`

5. transmission des modifications locales au serveur: `git push`

En pratique: récupération des fichiers du cours à partir du gitlab du département

Nous allons utiliser `git` pour récupérer les fichiers de ce cours à partir du serveur `git` (gitlab-ce) du département.

1. le site web du serveur
   - s'authentifier sur le serveur: https://gitlab-ce.iut.u-bordeaux.fr/users/sign_in
   - s'inscrire dans le projet `S1107-POO-Java.2019`
   - prendre en connaissance de l'ensemble des fichiers stockés dans ce projet
   - regarder la liste des révisions (commits)
   - regarder la liste des branches, ainsi que le graphe de celles-ci
   - regarder l'activité du projet
2. créer un répertoire `~/git` qui servira à stocker les répertoires des projets gérés avec `git`
3. à l'intérieur de celui-ci, récupérer une copie des données du projet `S1107-POO-Java.2019` (ce ne sera à faire qu'une seule fois)

```
git clone git@gitlab-ce.iut.u-bordeaux.fr:arpecher/s1107-poo-java.2019.git S1107-POO-Java
```

4. se déplacer dans le répertoire `S1107-POO-Java` et visualiser les différents supports du cours
5. faire un cycle de développement complet visant à rajouter dans le sous-dossier `tests` un nouveau fichier `essai-votreNom.txt` contenant le texte `J'ai réussi !`. Le message associé à votre version (révision) devra être: `votreNom: test`