TP 1 – Environnement Linux pour la programmation

1. Objectif

Le but de ce TP est de se familiariser avec l'environnement Linux pour réaliser des programmes. 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-manual.org/ ;
- la programmation par le biais d'un langage comme le C. C'est ce que nous verrons dans les TP suivants ;
- l'utilisation de commandes, par le biais d'un interpréteur de commandes, qui est l'objet de ce TP.

2. Le shell

1. Première commande

Le shell est le nom donné au programme qui interprète les commandes écrites par un utilisateur, puis appelle les fonctionnalités adéquat du systèmes 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 *.c` » « ls » est la commande, « -l » est le premier argument et « *.c » le deuxième 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 la commande, vous devez au préalable ouvrir un terminal (appelé aussi console) . Cela se fait en cliquant sur l'icône « écran noir ». Si celui-ci n'est pas disponible, déplacez votre souris en haut à gauche de l'écran, puis cliquez sur l'icône « tableau de bord »  puis en tapant « term ». Vous pouvez également appuyer simultanément sur les touches [ctrl][Alt][t].

En tapant « ls » suivi de la touche « Entrée », apparaît la liste des fichiers de votre répertoire de travail.
Maintenant tapez :

```
$ lzs
```

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

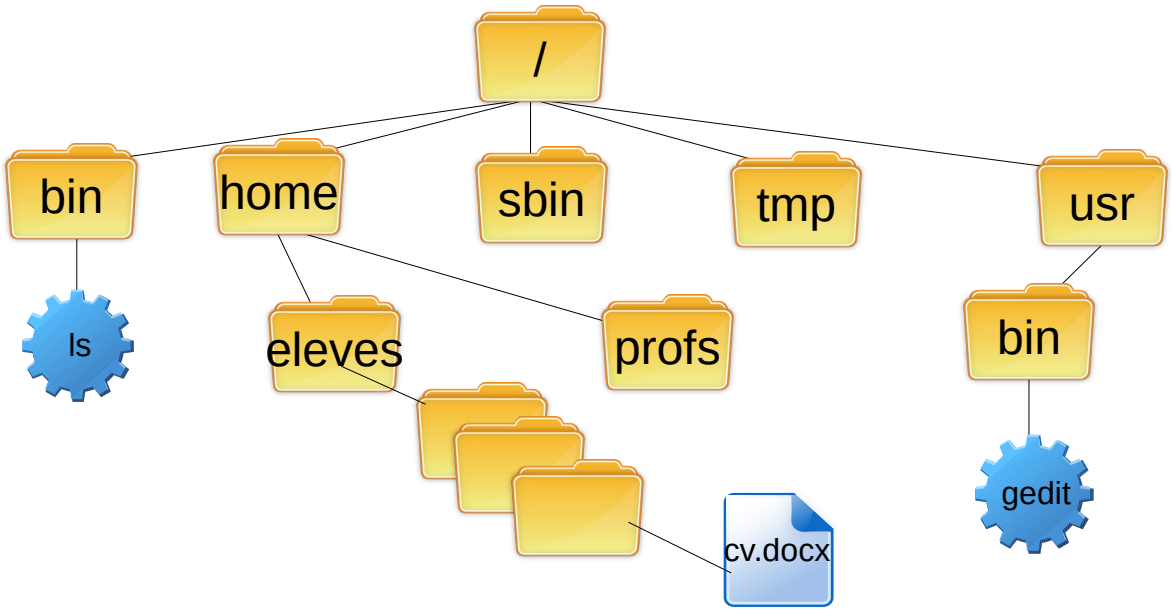
Peut-être vous demandez vous à quoi cela peut bien servir de taper des commandes alors qu'il existe, à l'instar de Windows, une interface Homme-machine (IHM) permettant d'effectuer la plupart des opérations en quelques clics ? Le shell permet de coupler les commandes entre elles et d'exécuter des actions complexes qui ne sont pas possibles par de simples clics. Aussi, Linux est un système utilisé dans des box internet ou 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 : http://doc.ubuntu-fr.org/projets/ecole/scripting/initiation_au_shell

2. 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 de 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/gedit`
- pour y accéder de manière absolue quelque soit le répertoire de travail : `/usr/bin/gedit`

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 :

- **~** : **répertoire personnel** de l'utilisateur ;
- **.** : **répertoire de travail**;
- **.. : répertoire parent**.

Ils permettent tous les trois de simplifier l'expression de références absolues.

Voici quelques commandes de manipulation de fichiers et répertoire :
pwd : affiche le **répertoire de travail (p**ath of **w**orking **d**irectory) dans lequel les commandes tapées sont exécutées
ls : liste 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* : 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*
rmdir *nomDeRep* : efface le répertoire *nomDeRep* si vide
rm *nomFic* **:** efface un fichier *nomFic*
ATTENTION, la commande rm ne met pas à la poubelle. L'effacement est définitif !
rm **.c* : efface tous les fichiers dont l'extension est « .c »
mv *toto titi :* renomme le fichier toto en titi
mv */tmp/toto ~ :* déplace le fichier « *toto* » du répertoire « /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 :
`$ man ls`
tapez q pour quitter

Exercice :
- créer un répertoire tpc
- dans ce répertoire créer deux répertoires tp1 et tp2
- comment afficher les répertoires dans l'ordre chronologique de création avec ls ? (lire le man)

3. Les raccourcis claviers sous linux

1. Dans une console

-  Les flèches haut/bas permettent de naviguer dans les commandes déjà entrées.
-  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.
- [ctrl][c] arrête une commande en cours.

2. Partout

-  La combinaison des 2 touches [alt][tab] permet de passer d'une fenêtre à l'autre.
- [ctrl][win][d] permet d'afficher le bureau
- [ctrl][alt][fleche droite] permet de changer de bureau
- Lorsque du texte est surligné avec la souris, l'appui sur la molette le copy/colle à l'endroit du pointeur.

4. Les droits

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 hubert bde          4096 oct.  14  2011 savDoc
-r--------  7 hubert bde          4096 oct.  14  2011 mail.txt
```

Pour le répertoire « *savDoc* »,
- le propriétaire « *hubert* » a les droits « r w x » 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 peut utiliser le gestionnaire de fichier (comme sous window's) ou utiliser la commande « chmod » suivi de la valeur en octal des droits. « r » a le poids 4, « w » 2 et « x » 1.

```
$ chmod 741 a.out
```
donne les droits « rwxr----x » au fichier a.out.

Exercice : **Changez** les droits du répertoire tp1 pour que seul le propriétaire ait les droits r w x.

5. 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, gedit, kedit, geany…
Pour ce TP nous utiliserons geany qui n'offre ni la souplesse de *emacs*, ni la convivialité de *sublime*, mais qui est très simple d'utilisation. Pour le lancer :
```
$ geany fichier1.txt &
```

Exercice : **créer** un premier fichier contenant
```
Siyahamba
is a South African hymn
```
et un deuxième contenant
```
written in
Zulu language.
```

La commande *cat* affiche le contenu d'un fichier sur la console. **Vérifiez le contenu** de vos 2 fichiers créés (vous pouvez passez en argument de *cat* les 2 noms de fichiers. ils seront ainsi affichés l'un après l'autre.

6. Les redirections (basé sur le TP de Jalal Fadili)

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.
`2>`	sortie erreur redirigée vers un nouveau fichier
`<`	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 »
`commande1 | commande2`	Avec le caractère | (pipe), la sortie de commande1 est redirigée sur l'entrée de commande 2.

Exercice : concaténé (mettre bout à bout) les 2 fichiers avec la commande *cat*. Le nouveau fichier s'appellera *resultat.txt*

7. Le filtre grep et les expressions régulières

grep est une commande qui recherche un motif et affiche toutes les lignes comportant ce motif. Par exemple
```
$ grep -E "an" resultat.txt
```
affiche les lignes du fichier *resultat* contenant la chaine "an". Le premier paramètre est le motif et le deuxième le nom du fichier (ou l'entrée standard si omis).

Le motif est exprimé par une grammaire, appelée expression régulière contenant des caractères spéciaux (méta-caractères).

. : correspond à un caractère quelconque sauf le retour à la ligne (de code ASCII 0x0D ou 0x0A)
***** : n'importe quel nombre d'occurrences du caractère précédent l'astérisque
+ : une occurrence ou plus de l'expression régulière qui précède ;
? : zéro ou une occurrence de l'expression régulière qui précède ;
[...] : défini un ensemble de caractères possibles pour l'occurrence. On utilise un tiret pour définir un intervalle de caractères. **[^...]** défini un ensemble de caractères exclus pour l'occurrence.

^ : en première position d'une expression précise que l'expression doit se trouver en début de ligne.
$: en dernière position d'une expression précise que l'expression doit se trouver en fin de ligne.
{n} : exactement n occurrences du caractère qui suit immédiatement ;
{n,} : au moins n occurrences du caractère qui suit immédiatement ;
{n,m} : entre n et m occurrences du caractère qui suit immédiatement. ;
**** : protection du caractère qui suit, pour spécifier un méta-caractère comme caractère ;

| : OU logique ; le motif correspond soit à l'expression qui précède, soit à celle qui suit ;
( ) : permet de regrouper des expressions régulières ;

Si l'option **-E** n'est pas utilisée, les caractères **( ) { } | + ?** sont interprétés comme des caractères normaux.

L'option **-R** permet de parcourir l'ensemble des fichiers et sous répertoires d'une arborescence.
D'une manière générale l'option -r des commandes Linux permet souvent de parcourir de manière *récursive* l'ensemble d'un répertoire et des sous-répertoires comme le très dangereux « rm -rf ».

L'option **-o** permet d'afficher le motif plutôt que la ligne complète contenant le motif.

Exemple :
`grep -E "i.t" resultat.txt`
affiche les lignes contenant la lettre 'i' suivie d'un caractère quelconque suivi de la lettre 't'
`grep -E "i.*t" resultat.txt`
affiche les lignes contenant la lettre 'i' suivie ou pas de caractères quelconques suivis de la lettre 't'

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

```
$ grep -E "i.+y" resultat.txt
$ grep -E "i.*y" resultat.txt
$ grep -E -n "i.*t" resultat.txt
$ grep -E -o "i.*t" resultat.txt
$ grep -E "^[A-Z]" resultat.txt
$ grep -E -v "^[A-Z]" resultat.txt
$ grep -E "n$" resultat.txt
$ grep -E -e "n$" -e "\.$" resultat.txt
$ grep -E "n$" resultat.txt |grep [aA]
```

8. Commande find

La commande find recherche un fichier dans un dossier et ses sous-dossiers en appliquant l'expression donnée. 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 répertoires.
$ 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. La commande doit se terminer par un point virgule. Cependant, comme le shell interprète le « ; » comme un séparateur de commandes shell, il faut préfixé le « ; » par un « \ » afin qu'il ne soit pas interprété.
```

Créer 2 fichiers avec l'extension « .c » dont l'un contient le mot « scanf ». **Écrire** une commande qui recherche tous les fichiers « *.c » qui contiennent le mot « scanf ». Utilisez l'option -l de grep.(*corrigé disponible)*