Cours Perl

Les objectifs de ce cours sont :

Se familiariser avec l'environnement UNIX/LINUX

Apprendre a se servir de quelques fonctionnalités disponibles dans cet environnement pour la manipulation des textes

L'**évaluation** du cours sera constituée par

Examen écrit

Devoir sur table

Séance 1 : Environnement UNIX/LINUX: présentation, avantages et inconvénients, utilisation, principales commandes.

- Les chemins d'accès aux fichiers
- Quelques commandes utiles : ls, pwd, cd, cat, grep, wc
- Exercice avec ces commandes

Support de cours : Introduction à Linux

Pourquoi une initiation à LINUX ?

LINUX est très puissant pour manipuler et traiter des gros volumes textuels et en linguistique le cas est fréquent.
Les commandes de base utilisées sous un shell (un interpréteur de commande comme DOS sous Windows) sont fortement corrélées à Perl et il est très utile de les manipuler
Connaître un système d'exploitation autre que Windows ne peut pas faire du mal, bien au contraire.

Pourquoi utiliser LINUX
Linux est gratuit. Vous pouvez télécharger une distribution de Linux sur Internet (même sans l'installer sur votre disque dur, **live-cd** : *ubuntu-live, knoppix*)
Linux est très stable
Linux est un système sûr (pas besoin d'antivirus). Vous ne craignez pas tous les virus de Windows
La particularité d'Unix par rapport à Windows est qu'il est muni d'interprètes de commandes (ou shells) fonctionnant en mode texte.

Qu'est-ce que Linux

Linux est un **système d'exploitation** (comme MacOS ou Windows). Le rôle d'un système d'exploitation est de contrôler les composants de votre ordinateur – le processeur, la mémoire, le disque, le clavier, le moniteur, etc. Le système d'exploitation relie tous ces composants d'une part et les applications d'autre part. Quand vous voulez que l'ordinateur fasse quelque chose pour vous – démarrer une application, copier un fichier, ou afficher le contenu d'un répertoire, c'est le système d'exploitation qui accomplit ces tâches.

Un peu d'histoire:

UNIX a été créé en début des années 70, par AT&T. Le but était de construire un système d'exploitation multi-tâche et multi-utilisateurs. Le succès de l'UNIX d'AT&T comme système d'exploitation pour les stations de travail a incité les constructeurs de machines (SUN, IBM, HP, SGI,...) à développer leur propre UNIX (SunOS/Solaris, AIX, HP/UX, IRIX,..) au début des années 80. Par ailleurs, vers la fin des années 70, l'université de Berkley a développé son UNIX : BSD au départ pour palier aux limites de système VAX. En début des années 80, la multitude des systèmes UNIX a poussé les universitaires et les industriels à réfléchir à un standard, le standard POSIX. Les différents systèmes UNIX ont mûri durant les années 80, et sont depuis considérés comme les plus efficaces et sont par conséquent largement utilisés dans l'industrie et les universités. Au début des années 90, un étudiant finlandais Linus Torvalds a commencé à développer avec l'aide d'un groupe de *hackers* un UNIX suivant le modèle GNU (distribué avec la licence GPL) en se basant sur le système Minix, et respectant le standard POSIX. Linux tourne désormais sur plusieurs architectures : ix86 (PC), 68000 (Atari, Amiga, Macintosh) et PowerPC (PowerMac, IBM), SPARC (Sun), ARM (SGI), Alpha (DEC) ... et dans les années 90 plusieurs distributions sont apparues : `Debian, Slackware , RedHat, Suse, Mandrake, Gentoo, Ubuntu`, ... et les interfaces graphiques (Gnome, KDE) ont beaucoup évolué permettant même à des utilisateurs novices d'utiliser le système.

Commandes de base pour l'interpréteur des commandes (le shell)

1. Chemins d'accès

On considère l'arborescence Unix suivante, où seul alex.jpg est un fichier. Tous les autres noms désignent des répertoires. A l'ouverture du shell vous êtes positionnes dans votre **home directory** (en l'occurrence /home/avolansk).

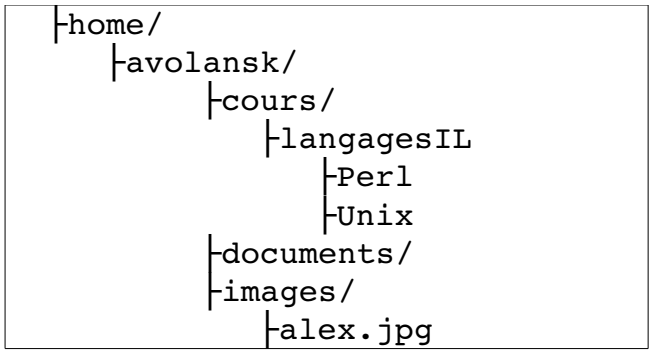

Figure 1

Le chemin absolu à l'un des fichiers ou répertoire de cet arbre, est la suite des noms rencontrés depuis la racine / jusqu'à cet élément, séparés par des /.

* /home/avolansk est un chemin d'accès absolu au *home directory* de Alexandra Volanschi,
* /home/avolansk/images/alex.jpg est un chemin d'accès absolu à alex.jpg

La suite /home/avolansk/images/alex.jpg peut aussi s'interpréter comme une suite de déplacements :

* / : aller à la racine
* home/ : descendre dans home
* avolansk/ : descendre dans avolansk
* images/ : descendre dans images
* logo.jpg : accéder à logo.jpg

Un chemin relatif décrit la suite des déplacements à effectuer à partir du répertoire courant jusqu'à un élément donné. Dans une telle suite, un double-point .. désigne le répertoire immédiatement au dessus du précédent, et le point . désigne le répertoire courant :

* si le répertoire courant est avolansk, images/alex.jpg est un chemin d'accès relatif à alex.jpg.
* si le répertoire courant est documents :
  - .. est un chemin d'accès relatif à /home/avolansk,
  - ../images accède au répertoire /home/avolansk/images.
  - ../images/alex.jpg accède au fichier alex.jpg.

Remarque

En pratique, on identifie souvent un fichier ou un répertoire avec son chemin d'accès (absolu ou relatif). Ainsi, dans la description des commandes ci-dessous, un argument de la forme *répertoire* (resp. *fichier*) pourra être remplacé par n'importe quel chemin d'accès à un

répertoire (resp. à un fichier).
Par exemple, `cd ..` permet de remonter dans le répertoire immédiatement au-dessus du répertoire courant, `cd ../../usr/local` permet de remonter de deux répertoires, puis de descendre successivement dans `usr` et dans `local`, etc...

2. Commandes de base pour le shell

2.1 Fichiers et répertoires

NOTE : à tout moment, pour plus de précisions sur une commande et son fonctionnement, utiliser la commande
$man commande.

1/pwd A tout moment, pour savoir dans quel répertoire on se trouve, on peut utiliser la commande pwd.

2/cd '***change directory***' pour changer de répertoire et aller dans le répertoire spécifié
exemple : `cd ..` permet de remonter d'un niveau
sans arguments, et quelque soit le répertoire courant, `cd` vous permet de revenir au sommet de votre répertoire personnel. (exemple : si vous vous trouvez dans le répertoire images, la commande vous permet de revenir dans /home/avolansk

3/ ls '***list***' permet l'affichage du contenu d'un répertoire :
`ls` *répertoire* : afficher le contenu de *répertoire*.
`ls -l` *répertoire* : afficher toutes les informations disponibles sur le contenu du répertoire spécifié.
Utilisées sans arguments, `ls` et `ls -l` s'appliquent au répertoire courant.

4/ mkdir '*make directory*' *répertoire* = créer *répertoire*.

5/ rmdir '*remove directory*' *répertoire* (remove directory) : détruire *répertoire*. Cette commande ne fonctionne que si le répertoire spécifié est vide.

6/ cp '*copy*' copie de fichiers/répertoires `cp` *fichier_1 fichier_2*

si *fichier_2* n'existe pas, créer une copie de *fichier_1* appelée *fichier_2*. - sinon, remplacer le contenu de *fichier_2* par celui de *fichier_1*.
`cp` *fichier répertoire* : créer dans *répertoire* une copie de *fichier* de même nom.

`cp -r` *répertoire1 répertoire2* (r pour récursif) : copie récursive des contenus du *répertoire1* dans le *répertoire2*

7/ rm '*remove*' destruction des fichiers/répertoires

`rm` *fichier* (remove) : détruire *fichier*. Noter que `rm -r` *répertoire* (`r` pour récursif) détruit le contenu de *répertoire* et de tous ses sous-répertoires.

8/ mv '*move*' : déplacement ou renommage
`mv` *fichier_1 fichier_2* : si *fichier_2* n'existe pas, changer le nom de *fichier_1* en *fichier_2*, sinon remplacer son contenu par celui de *fichier_1*, et détruire *fichier_1*.

`mv` *fichier répertoire* : déplacer *fichier* dans *répertoire*.

`mv` *répertoire_1 répertoire_2* :
- si *répertoire_2* n'existe pas, changer le nom de *répertoire_1* en *répertoire_2*. - sinon, déplacer *répertoire_1* dans *répertoire_2*.

NOTE : Destructions ou déplacements multiples : le caractère wildcard `*`

On peut déplacer ou détruire plusieurs fichiers ou répertoires simultanément, à l'aide du caractère `*`. Par convention, ce caractère ne fait jamais partie d'un nom de fichier ou de répertoire. Il est interprété par le Shell de la manière suivante:

mot`*` désigne tous les fichiers ou répertoires commençant par *mot* (qui peut être le mot vide).

Le premier argument des commandes `mv, cp` (ou même de `ls`, ainsi que de `cat` et `more`, voir ci-dessous) peut être de cette forme. La commande s'applique alors à tous les fichiers commençant par *mot*:

si `/home/avolansk/langagesIL/Unix` contient les fichiers `test_1.txt, test.txt, testtest`, la commande

`rm /home/avolansk/langagesIL/Unix/test*` effacera ces trois fichiers.

`rm *` efface le contenu du répertoire courant (faire attention a cette commande : ça a provoque de grandes catastrophes dans l'histoire).

`cp */test* /home/avolansk/documents` copie dans `/home/avolansk/documents` tous les fichiers commençant par `test` contenus dans les sous-répertoires immédiats du répertoire courant.

2.2 Manipulations de base sur les fichiers texte

Affichage : `cat, more,head,tail, echo`

9/ cat `'concatenate' 'cat` *fichier* : afficher à l'écran le contenu de *fichier*. Noter que l'on peut afficher à la suite les contenus de plusieurs fichiers, par un appel de la forme `cat` *fichier_1 ...fichier_n*, ou même `cat` *nom*`*`.

10/ more `more` *fichier* : afficher à l'écran le contenu de *fichier*, en s'arrêtant à chaque fois que l'affichage remplit une page écran. Pour continuer l'affichage, appuyer sur la barre d'espace.

11/ head `head -n 20` *fichier* : affiche les 20 premières lignes de *fichier*

12/ tail `tail -n 20` *fichier* : affiche les 20 dernières lignes de *fichier*

13/ echo

`echo` *chaîne*. Afficher *chaîne* (une suite de caractères quelconque) à l'écran, suivi d'un retour-chariot (caractère invisible, provoquant le retour à la ligne).

`echo` *chaîne* > *fichier*: au lieu d'afficher *chaîne* à l'écran, créer *fichier*, et stocker la sortie de `echo` *chaîne* (c'est-à-dire *chaîne*, suivi d'un retour-chariot) dans *fichier*. Noter que si *fichier* existe, son contenu est écrasé. Ainsi, la séquence de commandes

```
echo toto > test.txt
cat test.txt
echo tutu > test.txt
cat test.txt
```

affichera à l''ecran successivement `toto` suvi d'un retour-chariot (premier contenu de `test.txt`) puis `tutu` suvi d'un retour-chariot (second contenu).

Redirection : > et >>

Plus généralement, l'opérateur > permet de rediriger la sortie de n'importe quelle commande vers un fichier, en écrasant son contenu sicelui-ci existe déjà. Ainsi,

`cat` *fichier_1 fichier_2* affiche à l'écran les contenus de *fichier_1* et *fichier_2*.

`cat` *fichier_1 fichier_2* > *fichier_3* stocke à la suite dans *fichier_3* les contenus de *fichier_1* et *fichier_2*. La séquence de commandes

```
echo toto > test_1.txt
echo tutu > test_2.txt
cat test_1.txt test_2.txt > test_3.txt
cat test_3.txt
```

affichera:

```
toto
tutu
```

L'opérateur >> redirige de la même manière la sortie d'une commande vers un fichier, mais, dans le cas où celui-ci existe déjà, en stockant celle-ci à la fin du fichier. Ainsi, dans l'exemple précédent,

```
cat test_1.txt test_2.txt > test_3.txt
```

est équivalent à:

```
cat test_1.txt > test_3.txt (stocker toto dans test_3.txt)
cat test_2.txt >> test_3.txt  (stocker tutu après toto).
```

3 Exercices

1. Répertoires : `pwd, ls, cd, mkdir, mv`

Afficher la position du répertoire dans lequel vous vous trouvez. Afficher son contenu. Revenir, si vous n'y êtes pas déjà, à votre répertoire privé (utiliser la commande `cd` sans arguments).

1. Créer à partir de votre répertoire privé l'arborescence dans la Figure1. Commencer avec les sous-répertoires *`Images`*, *`Documents`* et *`Cours`*. Descendre dans `Cours`:

   créer un sous-répertoire `langagesIL`.

   sans changer de répertoire, créer un sous-répertoire de `langagesIL` appelé `Perl`.

   toujours sans changer de répertoire, créer un nouveau sous-répertoire de `langagesIL` appelé `Unix`.

2. Descendre, en une seule étape, dans `Perl`. Sans changer de répertoire, créer un sous-répertoire d'`Unix` appelé `Exos`.

3. Remonter, en une seule étape, dans `Documents`.

afficher le contenu de ce répertoire.

en une seule étape, copier le répertoire `Exos` dans `Perl`.

sans changer de répertoire, afficher le contenu de `Perl` à sa nouvelle position.

toujours sans changer de répertoire, changer le nom de `Unix` en `Linux`, sans changer sa position.

4. Aller dans `Linux` en une seule étape, puis afficher son contenu.

5. Dessinez sur un papier cette arborescence après ces étapes.

Si vous êtes perdus, vous pouvez bien sûr à tout moment

1. afficher la position du répertoire courant,
2. afficher son contenu,
3. afficher le contenu de ses sous-répertoires.
4. effacer un répertoire créé par erreur, à l'aide de `rmdir` (à condition qu'il soit vide, sinon avec rm -r).