Programmation C
L3 ESA 2008–2009
hoenen@icps.u-strasbg.fr

Université de Strasbourg
Département d'informatique
24 mars 2009

TP1 : introduction UNIX et environnement de développement

Introduction

Pour interagir avec un système UNIX, l'utilisateur rentre des commandes dans un interpréteur : le shell. Vous pouvez lancer cet interpréteur à l'aide d'un programme qui peut s'appeler console ou terminal (ou un nom équivalent, dépendant du système installé). Dans cet interpréteur, la présence du *prompt* en début de ligne (généralement le caractère `$`, qui peut parfois être précédé du nom de l'utilisateur, de la machine ainsi que d'un chemin dans le système de fichier, par exemple : `nom@machine:~$`), signifie que l'interpréteur est prêt à recevoir une commande. Taper cette commande au clavier puis appuyer sur la touche `<RET>` (retour chariot, ou touche "entrée") permet d'exécuter une commande.

Les commandes UNIX suivent généralement le format suivant `commande -options cible`, ou `command` est le nom de la commande à exécuter, `options` est une liste d'options (souvent une suite de lettres ou des mots qui suivent un tiret) pour cette commande, et `cible` est l'entité sur laquelle la commande sera exécutée.

Attention : UNIX respecte la casse, autrement dit le système fait la différence entre les lettres en majuscule et en minuscule. La commande `ls` permet de lister les fichiers d'un répertoire alors que la commande `LS` n'existe pas.

Commencez par vous familiariser avec les différents menus de votre espace de travail : trouvez notamment comment ouvrir un shell (console, terminal, xterm, etc...), un navigateur internet et un éditeur de texte.

Trouver de l'aide

On va maintenant voir un ensemble de commandes qui pourra vous être utile pour utiliser correctement un système UNIX. La liste des commandes présentée ici est loin d'être exhaustive étant donné le nombre de commandes existantes. De même, seules certaines options seront présentées ici. Nous allons maintenant voir comment consulter la liste des commandes disponibles, et la description de ces commandes ainsi que leurs options.

Autocomplétion Dans de nombreux shell, la touche tabulation (`tab`) du clavier est associée au mécanisme d'autocomplétion. Ce mécanisme permet de compléter automatiquement tout ou une partie de la commande que l'utilisateur est en train de taper :

- si le mot est complétée et suivi d'un espace, une unique commande a été associé au début du mot tapé au clavier,
- si il n'y a pas d'espace ajouté après le mot (qu'il soit complété ou pas), alors il existe plusieurs commandes commençant par le même préfixe,
- si il y a conflit entre plusieurs commandes, appuyer une deuxième fois sur la touche tabulation permet d'afficher l'ensemble des commandes partageant le même préfixe.

Ne rien taper après le prompt et appuyer deux fois sur `tab` permet d'afficher l'ensemble des commandes disponibles sur le système.

Le manuel Pour savoir à quoi servent les différentes commandes, on peut consulter le manuel en ligne par la commande : `man commande`. Les pages de ce manuel permettent de décrire la commande ainsi que chaque option de cette commande. Les touches espace, entrée et les flèches permettent de se déplacer dans le man, la touche `/` suivie d'un mot permet de retrouver ce mot si il est présent dans cette page de man.

Liste de commandes

Apprivoiser l'arborescence de fichiers Pour rappel, dans un système UNIX, *tout est fichier* ! Les répertoires sont des fichiers qui peuvent contenir d'autres fichiers et donc d'autres répertoires, ce qui engendre une arborescence de répertoires que l'on appelle le système de fichiers. Au début de cette arborescence se trouve un répertoire unique, la racine, notée `/`. Depuis cette racine, un répertoire contenu et séparé du répertoire contenant (ou répertoire parent) par le caractère `/` (c'est le seul caractère qui n'est pas utilisable dans un nom de fichier).

Chaque fichier et répertoire d'un système UNIX possède un chemin absolu depuis la racine qui correspond à la liste de ses répertoires parents jusqu'à la racine du système : un chemin absolu débute donc toujours par un `/`. Un chemin peut également être relatif, c'est à dire entre deux répertoires du système. Les raccourcis suivants concernent l'arborescence de fichiers :

`.`	représente le répertoire courant
`..`	représente le répertoire parent
`~`	représente le répertoire de login de l'utilisateur

Les commandes suivantes sont utiles pour gérer les fichiers et se déplacer dans l'arborescence du système de fichiers :

`ls`	afficher le contenu de répertoires `ls -l` : afficher sous forme de liste détaillée `ls chemin` : afficher le contenu d'un répertoire autre que le répertoire courant
`cd`	changer de répertoire de travail (ou répertoire courant) `cd ..` pour aller dans le répertoire parent `cd documents` pour aller dans le répertoire `documents` présent dans le répertoire courant
`pwd`	afficher le nom complet du répertoire courant
`mkdir`	créer des répertoires `mkdir rep` : créer un répertoire de nom `rep` dans le répertoire courant
`rmdir`	supprimer des répertoires vides `rmdir nom` : supprimer le répertoire `nom` du répertoire courant (doit être vide)

`cp`	copier des fichiers et des répertoires
	`cp source destination` pour copier le fichier `source` dans un fichier `destination`
	`cp -r src dest` pour copier le répertoire `src` et tout ce qu'il contient dans `dest`

`mv`	déplacer ou renommer des fichiers `mv src dest` renomme le fichier `src` en `dest` (éventuellement en le déplaçant)
`rm`	effacer des fichiers ou des répertoires `rm fich` pour supprimer le fichier `fich` **Attention : le contenu du fichier sera définitivement perdu !**

Contenu des fichiers Dans cette partie on pourra avoir recours aux redirections à l'aide des *pipes* `|`, `>` et `>>`. Utiliser `> fich` après une commande permet de rediriger le résultat de cette commande vers le nouveau fichier `fich` (écrase le contenu du fichier si il existe déjà), et `>> fich` permet de rediriger le résultat à la fin d'un fichier `fich` existant sans en écraser le contenu. Pour lier la sortie d'une commande à l'entrée d'une autre commande, utiliser `com1 | com2`.

Les commandes suivantes sont utiles pour lire ou effectuer des opérations depuis le contenu de fichiers :

`cat`	concaténer des fichiers et les afficher sur la sortie standard
	`cat f1` pour afficher le contenu d'un fichier `f1`
	`cat f1 f2 > f3` pour mettre le contenu des fichiers `f1` et `f2` dans un nouveau fichier `f3`

`diff`	comparer des fichiers ligne par ligne
`grep`	afficher les lignes d'un fichier contenant un motif donné `grep "toto" fich` pour afficher toutes les lignes du fichier `fich` qui contiennent le mot `toto`
`echo`	afficher une ligne de texte `echo "bonjour" > fich` pour créer un fichier `fich` qui contient le texte `bonjour`

`more` afficher le contenu d'un fichier page après page
(appuyer sur `q` pour quitter et espace la page suivante)

`du` évaluer l'espace disque occupé par des fichiers

`wc` afficher le nombre de lignes, de mots et d'octets d'un fichier

`chmod` modifier les autorisations d'accès à un fichier
`chmod u+x f1` pour ajouter (`+`) à l'utilisateur (`u`) les droits d'exécution (`x`) sur un fichier `f1`

Gestion des processus et du système Lorsqu'une commande est lancée dans l'interpréteur, ce dernier n'est plus accessible tant que la commande n'est pas terminée, on dit que la commande *ne rend pas la main*. Pour arrêter un processus (un programme ou une commande en exécution), on peut lui envoyé plusieurs types de signaux. Par exemple appuyer simultanément sur les touches `ctrl` et `c` du clavier (ou `ctrl+c`) permet de forcer un processus à terminer. La combinaison `ctrl+z` permet de suspendre le processus et de redonner la main au shell. On peut alors utiliser la commande `bg` pour faire reprendre le processus suspendu en tâche de fond (ne bloque plus le shell). On peut également préciser dès le lancement de la commande que l'on souhaite que celle-ci s'exécute en tâche de fond : pour cela il suffit de faire suivre la commande du caractère `&` avant d'appuyer sur la touche entrée.

Les commandes suivantes permettent d'effectuer des opérations concernant les processus ou le système :

`ps` afficher un instantanné des processus courants
(permet d'associer chaque programme/processus avec son identifiant unique, le pid)

`top` afficher de manière interactive les tâches qui s'exécutent

`kill` envoyer un signal à un processus
`kill pid` pour tuer un processus identifié par son `pid`

`sleep` endormir un processus pour une durée déterminée

`date` afficher ou configurer la date et l'heure du système

`time` exécuter un programme et afficher un résumé de l'usage des ressources système

`clear` effacer l'écran du terminal

`history` afficher l'historique des commandes

`uname` afficher des informations sur le système

`finger` afficher des informations sur un utilisateur

`passwd` modifier le mot de passe d'un utilisateur
Attention : pour être sécurisé, un mot de passe doit faire au moins 7 ou 8 caractères et mélanger lettres, chiffres et si possible ponctuation (sauf caractères accentués)

Outils Il existe de nombreuses commandes et outils dans un système UNIX/Linux, nous ne présentons dans la suite que quelques commandes

`find` rechercher des fichiers dans une hiérarchie de répertoires
`find ~ -name "fi*"` pour retrouver tous les fichier dont le nom débute par `fi` depuis `~`

`tar` utilitaire de gestion d'archives
`tar -xvzf arch.tgz` pour décompresser et désarchiver le contenu de l'archive `arch.tgz`

`gzip` compresser un fichier

`gunzip` décompresser un fichier

`cal` afficher un calendrier

`bc` utilitaire calculatrice
`bc -l` pour utiliser les calculs flottants et la fonctions mathématiques

`ssh` client de connexion sécurisée à distance
`ssh login@ordi` pour se connecter comme utilisateur `login` sur la machine `ordi`

`xclock` horloge analogique/digitale pour X

Pratique des commandes et du `man`

Dans un nouveau répertoire `TP1` :

1. Trouver la date du prochain vendredi 13, et le jour de la semaine où vous êtes né.
2. Afficher une fenêtre avec l'heure sous la forme "hh :mm AM/PM".
3. Compter le nombre de fichiers/répertoires de votre `~` qui ont été accédés/modifiés aujourd'hui.
4. Trouver les options de la commande `tar` pour :
   - créer une archive compressée en listant les fichiers ajoutés à l'archive,
   - décompresser et désarchiver un fichier portant l'extension `.tar.gz` (ou `.tgz`).
5. Tester la commande précédente sur un répertoire `test` contenant les fichiers :
   - `test1`, qui contient la liste des fichiers/répertoires de `~`,
   - `test2`, qui contient la liste des fichiers/répertoires de `/users/externe`,
   - `test3`, qui contient la liste des fichiers/répertoires de la racine.
6. Trouver et afficher dans un format facilement compréhensible par un humain la taille des fichiers suivants :
   - l'archive compressée issue de la question précédente,
   - la même archive mais cette fois non compressée,
   - l'ensemble du contenu de `~`.
7. Afficher alors uniquement les fichiers dont la taille est supérieure à 1Mo, puis ceux dont la taille est inférieure.

Environnement de développement

On va maintenant faire un premier petit programme `C` qui affiche le classique `Hello world` à l'écran. Voici le code de ce programme :

```
#include <stdio.h>
int main(int argc, char **argv)
{
  printf("Hello world\n");
  return 0;
}
```

1. La première chose à faire est de choisir un éditeur de texte. Il en existe de nombreux, plus ou moins puissants mais aussi plus ou moins complexe. Parmi les plus simples, vous pouvez essayer `nedit`, `gedit`, `kedit`, `kwrite` (attention, tous ne sont pas forcément installés). Pour un éditeur plus puissant, vous pouvez essayer `kate`, `emacs` ou même `vi`.
2. Ouvrez un fichier `helloworld.c` avec votre éditeur et écrivez le code donné plus haut. Vérifiez que votre éditeur gère la coloration syntaxique (la mise en couleur des mots clés du langage de programmation) et l'indentation automatique (les tabulations pour aligner les instructions d'un même bloc).
3. Ensuite compilez le programme par à l'aide du compilateur GNU C :
   `gcc -Wall hellowolrd.c -o helloworld`
   Que signifient les options `-Wall` et `-o` ?
4. Vérifiez qu'un fichier exécutable `helloworld` a été créé et testez-le en lançant la commande `./helloworld` dans le shell.
5. Modifiez le programme pour qu'il affiche `Hello world` un nombre de fois passé en paramètre du programme.