Un peu plus loin dans le fonctionnement du SHELL

Septembre 2015

Objectifs
Utiliser des nouvelles commandes.
Utiliser les redirections.
Manipuler des variables d'environnement.
Une petite auto-évaluation.

1 D'autres commandes utiles

1.1 Compression

Comme l'espace disque n'est pas infini, nous vous conseillons de régulièrement archiver et compresser vos TP. La commande à utiliser est `tar`.
Un exemple abstrait d'utilisation pour la compression et archivage :
`tar czf nom-de-l'archive répertoire1 fichier1 fichier2`
Les options utiles sont :
c (création)
x (extraction)
t (listage contenu)
v (verbose),
f pour préciser le nom du fichier d'archive, pensez à inclure les « extensions » dans le nom .tar .tar.gz .tgz
z pour compresser

▷ **Exercice 1 :**

1. *Listez le contenu de l'archive* `Arbo/Prog/TP01.tar`. *Puis décompressez la.*
2. *Créez dans* `Prog` *une archive nommée* `vecteur.tgz` *qui contiendra tous les fichiers d'un même TP de C sur les vecteurs, c-a-d le fichier* `makefile` *et ceux dont le nom contient* `VectInd`. *Pensez à utiliser les méta-cartactères pour cette question. Vérifiez votre opération en listant le contenu de votre archive.*

1.2 La commande grep

La commande `grep` permet de chercher du texte dans un fichier texte. Elle est très puissante. Nous l'utiliserons de la façon suivante : `grep ''texte recherché'' fichier1 fichier2`

▷ **Exercice 2 :**

1. *Le répertoire* `TP01` *contient le sujet d'un Tp en langage pascal. Il y a une fonction qui s'appelle inverse définie dans le fichier* `inverse.pas`.
   Est ce que celle-ci est utilisée dans le fichier `test_dbc.pas` *?*
2. *Cherchez les fichiers dans le répertoire* `Prog` *qui contiennent le mot* `stdio`. *C'est le nom d'une librairie.*

1.3 La commande find

La commande `find` peut être utilisée pour trouver un fichier suivant certains critères : son nom, sa taille, sa date, son type,... Cette commande est très riche, nous n'aborderons qu'une utilisation élémentaire, pour découvrir son potentiel, tapez `man find`.
Rappel : le système fait la différence entre minuscule et majuscule. Faites attention lorsque vous nommez vos fichiers. Évitez de donner des noms avec des espaces.
La syntaxe de la commande est la suivante :
`find répertoire_de_départ caractéristique_recherchée`

▷ **Exercice 3 :**

1. *Cherchez tous les fichiers dont le nom est* `TP01.tar`.
2. *Comment chercher tous les fichiers dont les noms commencent par un «T» majuscule ou un «t» minuscule ?*
3. *Est ce que tous vos fichiers de langage C sont dans le même répertoire ?*
4. *Quelle différence faites-vous entre les commandes* `whereis` *et* `find` *?*

2 Les redirections

Les commandes Unix et les programmes ont, sauf exception, un fichier d'entrée, un fichier de sortie et un fichier d'erreurs. Par défaut, le fichier d'entrée est le clavier, le fichier de sortie est l'écran.
Il est possible de les rediriger à l'aide de caractères interprétés par le shell.

— le caractère > signifie « rediriger la sortie vers ». Dans l'exemple `commande > res.txt`, le résultat de la commande n'apparaîtra pas à l'écran mais sera copié dans un fichier de nom `res.txt` (dont le contenu précédent, s'il existe, sera effacé).

— les caractères >> signifient ajouter la sortie au fichier. `commande >> res.txt` copiera la sortie de la commande à la fin du fichier `res.txt`

— le caractère < signifie « lire à partir d'un fichier ». Si le programme `prog` lit ses données à partir du clavier, la commande `prog < exp1.dat` lui fera lire ses données dans le fichier `exp1.dat`

— le caractère `|`, appelé pipe, est un « tube » qui fait communiquer deux programmes. Dans l'exemple `ls | more`, la sortie de la commande `ls` est l'entrée de la commande `more`

— rediriger les erreurs dans un fichier : `prog >\& res-err.txt` envoie la sortie et les erreurs dans le même fichier `res-err.txt`

— rediriger la sortie dans un fichier et les erreurs dans un autre : `(prog > res.txt) >\& err.txt`

Remarque :
— pour ne pas visualiser ou mémoriser une sortie, redirigez vers `/dev/null`.
— pour rediriger vers la sortie standard , utilisez `/dev/tty`

▷ **Exercice 4 :**
Nous utiliserons la commande `getent passwd` *pour interroger l'annuaire des comptes informatiques.*

1. *Comment lister page à page l'annuaire des comptes ?*
2. *Chercher la ligne qui contient votre nom de famille ?*

3. *Combien de personnes ont le même prénom que vous dans notre annuaire ?*

3 Connexion à distance

Les ordinateurs sous Linux offrent la possibilité de se connecter à distance (avec un environnement graphique) sur un autre poste.
Par exemple, la commande `ssh nom_station` établie une connexion sécurisée avec un autre ordinateur dont le nom est `nom_station` puis exécute un SHELL sur cet ordinateur.
La station à laquelle on désire se connecter peut avoir un nom de login différent. Aussi, il faudra le spécifier lors de la connexion : `ssh login@nom_station`.
Enfin, il est possible aussi de « déporter » l'affichage, c'est-à-dire que l'affichage graphique d'une application lancée sur l'ordinateur auquel on s'est connecté se fera sur notre ordinateur :
`ssh -X nom_station`.

▷ **Exercice 5 :**
Connectez-vous à la station Leia.
Exécutez Iceweasel sur cette station.
Que se passe-t-il ?
Comment remédier au problème ?

4 Variables d'environnement

4.1 Définition

Une variable d'environnement a un NOM en majuscule et un contenu.
Les variables d'environnement sont transmises par le système d'exploitation à tous les processus.
Lors de leur démarrage, les shells exécutent des fichiers de configuration, qui peuvent contenir des commandes pour définir des variables d'environnement.

Quelques variables d'environnement

Nom	Description
USER	Le nom d'utilisateur de la personne actuellement attachée au système
PATH	La liste des répertoires, séparés par deux points, pour la recherche des programmes
SHELL	Le nom de l'interpréteur de commandes actuellement utilisé
TERM	Le nom du terminal de l'utilisateur. Utilisé pour déterminer les capacités du terminal
OSTYPE	Type du système d'exploitation
MACHTYPE	L'architecture du CPU sur lequel tourne actuellement le système
MANPATH	La liste des répertoires, séparés par deux points, pour la recherche des pages de manuel

Voici des éléments de syntaxe pour la manipulation des variables :

- Affichage du contenu d'une variable : `echo $NOM`
- Visualisation des variables : `env`, `printenv` ou `setenv`
- Définition variable `setenv NOM valeur`
- Ajout à une variable `setenv NOM ${NOM}:valeur1:valeur2`

4.2 Visualisation de variables

Nous allons faire l'état des lieux du système avant de faire des modifications.

▷ **Exercice 6 :**
Tapez `setenv | more`
Remarquez que le contenu de la plupart de ces variables sont des chemins.
Affichez le contenu des variables d'environnement suivantes : PRINTER et HOME.

4.3 Le path

Une des variables les plus importante est le `PATH`. Son contenu est une chaîne composée de chemins de répertoires séparés par deux-points.
Affichez son contenu en saisissant : `echo $PATH`
Lorsqu'on entre une ligne de commande sous un shell, ce dernier demande au système d'exécuter la commande, après avoir éventuellement substitué les méta-caractères et remplacé les alias. A chaque commande doit correspondre un fichier exécutable. Il est recherché par le système dans une liste de répertoires contenus dans la variable `PATH`. La recherche se fait dans l'ordre.

▷ **Exercice 7 :**

1. *Lister les commandes commençant par c ?*
   Il y a beaucoup de commande qui commencent par c.
2. *Mémoriser le contenu de* `PATH` *dans la variable* `OLDPAtH`.
   Modifier la variable `PATH` *à* `/usr/local/bin`.
3. *Vérifier que la création de* `OLDPAtH` *et la modification de* `PATH` *s'est bien passée.*
4. *Recommencez à cherchez les commandes qui commencent par c. Qu'observez vous ?*
5. *Remettez votre* `PATH` *comme il faut en tapant source* `.login`. **Attention pas de .login**

5 Éléments indispensables à connaître

— Connaître l'arboresence du système de fichier
— Savoir se situer et se déplacer à l'aide des chemins absolus et relatif
— Comprendre à quoi servent les droits sur les fichiers et les répertoires
— Maîtriser la visualisation et la modification de ces droits
— Avoir une idée du fonctionnement de l'interpréteur de commande (alias, méta-caractères, path ...)
— Pouvoir lire un man pour trouver la syntaxe correcte d'une commande
— Visualiser les processus

6 Petite auto-évaluation

1. Placez-vous dans votre home. Indiquer quelle commande vous utiliser.
2. Expliquez la commande suivante : `ls [tT1]* > toto.lst`
3. Créez depuis votre home, un fichier vide `test1` qui sera à l'emplacement suivant :
   `/home/login/1TR/AccSys/TP3/test1`
   Que devez vous faire au préalable ?
4. Placez-vous dans le répertoire `TP3`. Expliquez la commande `cp test1 ../test2/`
5. Quelle commande affiche ce résultat :
   `-rwx------ login Domain User 40 test1`

6. Quel est le résultat de la commande suivante : `chmod 745 test1` ?
7. Quelle est la difference entre `cd /home/login/1TR` et `cd home/login/1TR` ?
8. `test1` et `./test1` retourne le message suivant : `test1: command not found`
   Que fait le shell avec ces lignes. Quelle est la démarche à suivre pour pouvoir exécuter `test1` ?
9. Que font ces commandes ?
   ```
   find . -user login |wc -l
   cp test* Validation
   ```

7 Annexes

Pour être très rapide au clavier et gagner du temps lors des TP, voici une liste de combinaisons de touche utiles :

Touche	Effet
`Ctrl c`	Caractère `intr` tue le processus en cours attaché au terminal. Envoie le signal `SIGINT`.
`Ctrl z`	Caractère `susp` suspend l'exécution du processus en cours attaché au terminal. Envoie le signal `SIGTSTP`.
`Ctrl s`	Caractère stop. Arrête le défilement de l'écran
`Ctrl q`	Caractère start. Reprend le défilement de l'écran
`Ctrl l`	Efface l'écran en mettant la ligne courante en haut
`Ctrl j`	Passe à la ligne suivante
`Ctrl d`	Si le curseur est en début de ligne, envoie le caractère `EOF` et tue le shell
`Ctrl \`	Envoie le signal `SIGQUIT`. Arrête le processus en cours attaché au terminal et crée un fichier core, image mémoire du programme (si ulimit l'autorise).

Frappes de contrôle

Commande	Description	Exemple
`echo`	affichage de ce qui suit	`echo "nom machine est : " $HOST`
`set`	affectation ou affichage de toutes	`set var1="contenu"`
`setenv`	affectation ou affichage var environnement	`setenv HOST machine`
`alias`	définit un alias	`alias`
`unalias`	supprime un alias	
`which`		
`bindkey`	affiche les frappes de contrôle du shell	

Quelques commandes internes au shell

Touche	Effet
`Ctrl h`	effacement du caractère précédent le curseur
`Ctrl d`	effacement du caractère sous le curseur. Si le curseur est en début de ligne, tue le shell.
`Ctrl u`	effacement de toute la ligne
`Ctrl k`	effacement à partir du curseur jusqu'à la fin de la ligne
`Ctrl @`	définit une marque de région (set-mark-command)
`Ctrl w`	effacement de la région, par défaut des caractères à gauche du curseur jusqu'au début de ligne (kill region)
`Ctrl y`	recopie (yank) à partir du curseur ce qui a été effacé
`Ctrl a`	positionnement en début de ligne
`Ctrl e`	positionnement en fin de ligne
`Ctrl t`	échange le caractère sous le curseur avec le précédent transpose
`Ctrl b`	déplacement arrière d'un caractère (backward)
`Ctrl f`	déplacement avant d'un caractère (forward)
`Ctrl i`	complète le mot (complete word)
`Maj b`	déplacement arrière d'un mot (backward)
`Maj f`	déplacement avant d'un mot (forward)
`Maj d`	effacement du mot à droite
`Maj u`	Convertit le mot à droite du curseur en majuscules
`Maj l`	Convertit le mot à droite du curseur en minuscules
`Maj c`	Met la première lettre du mot à droite du curseur en majuscules

Correction d'une ligne en shell tcsh