Système d'exploitation

Jean-Sébastien Coron

Université du Luxembourg

October 3, 2011

Plan du cours

- Contrôle des processus.
- Les variables shell
- Scripts simples.

Le shell

- Un shell est un programme qui lit et exécute les commandes de l'utilisateur.
- Un shell permet aussi :
  - De controler les processus.
  - De rediriger l'entrée et/ou la sortie.
  - D'écrire des scripts.
- Script shell:
  - Fichier texte contenant une série de commandes.
  - Comme fichier .bat sous MS-DOS.
- De nombreux shell existent:
  - sh, bash, csh, tcsh.

Contrôle des processus

- Le shell permet de contrôler les processus qui s'exécutent.
- Un processus s'exécute soit en foreground, soit en background.
- Foreground.
  - Le processus reçoit les commandes du clavier et renvoie sa sortie à l'écran.
  - Un seul processus en foreground par shell.
  - Peut être suspendu avec Ctrl-Z.
  - Relancé en foreground par `fg`, en background par `bg`.
  - Définitivement interrompu en tapant Ctrl-C.

Contrôle des processus

- Exécution d'un processus en background:
  - On utilise le caractère &.
    ```
    $ find -name hello.txt &
    [1] 2812
    $
    ```
  - Cherche un fichier nommé hello.txt dans le répertoire courant.
  - Le processus s'exécute en tache de fond.
  - 1 représente le numéro du processus s'exécutant en background. 2812 représente son PID.
- `jobs` permet d'avoir la liste de tous les processus en background.

Affichage des processus

- La commande `ps` permet d'avoir la liste des processus qui s'exécutent sur la machine.

```
$ ps
  PID   TTY      TIME     CMD
 2653   con      00:01:03 bash
 3672   0        00:00:05 find
 3687   0        00:00:00 ps
```

- Pour stopper un process, on utilise la commande `kill`.
  - `kill 3672` stoppe l'exécution de la commande `find`.

Les variables shell

- Une variable est une donnée identifiée par un nom.
- On accède au contenu d'une variable en ajoutant $.

```
$  var='hello world'
$  echo $var
   hello world
```

- Pour qu'une variable soit visible en dehors du shell, il faut l'exporter vers l'environnement.
  - `export var`.
  - Exemple: la variable PAGER est utilisée par `man` pour déterminer comment afficher le texte.
    - `export PAGER=cat`: tout s'affiche à la fois.
    - `export PAGER=less`: page par page.

La variable PATH

- Variable d'environnement qui donne la liste des répertoires que le shell recherche pour exécuter des commandes.
  - Si `PATH` contient `/bin/:/usr/bin:/usr/local/bin:.`
  - alors pour la commande `cat`, le shell va chercher `/bin/cat`, `/usr/bin/cat`,...
- Si le `PATH` contient ".", alors il cherche aussi dans le répertoire courant.
  - On peut taper `macommande` au lieu de `./macommande`.
  - `PATH=$PATH:.`

Evaluation des variables

- Le shell substitue les valeurs des variables dans un texte avant son évaluation.
- Pour modifier l'évaluation, on peut utiliser "" ou ''.
- Les guillemets "":
  - Permet de grouper des mots, supprime le remplacement des méta-caractères, sans supprimer le remplacement des variables.
  - `x=hello world`
    `world: command not found`.
  - `x="hello world"` est correct.
  - `y="phrase=$x"; echo $y`
    `phrase=hello world`.

Les quotes ’ ’

- Les quotes ’ ’
  - Groupent les mots et suppriment toute évaluation.
- Exemple:
  -
```
$ y=hello
$ ls
fich1 fich2
$ echo le caractère * $y
le caractère fich1 fich2 hello
$ echo "le caractère * $y"
le caractère * hello
$ echo ’le caractère * $y’
le caractère * $y
```

Evaluation des commandes

- Les back-quotes (‘) permettent de substituer le résultat d’une commande.
  - `$ echo the date is ‘date‘`
    `the date is Tue Nov 17 15:01:44 2004`
- On peut stocker le résultat d’une commande dans une variable.
  - `$ x=‘ls‘`
    `$ echo $x`
    `fich1 fich2`
  - Equivalent: `$ x=$(ls)`

Découpage des chemins

- Les commandes `dirname` et `basename` sont utiles pour découper un chemin en répertoire/nom de fichier.
- `$ dirname /vers/mon/rep/fich.txt`
  `/vers/mon/rep`
- `$ basename /vers/mon/rep/fich.txt`
  `fich.txt`

Les scripts shell

- Un script shell est un fichier texte contenant des commandes du shell.
- Permet l'automatisation des taches.

Un script simple

- Soit le script suivant dans le fichier `simple`

```
#!/bin/bash
# commentaire
echo "Le nombre d'argument est $#"
echo "Les arguments sont $*"
echo "Le premier est $1"
echo "Le numéro du process est $$"
echo "Entrez un nombre:"
read nombre
echo "Le nombre entré est $nombre"
```

- Pour exécuter le script, il faut rendre le fichier `simple` exécutable (`chmod`).