Système d'exploitation

Jean-Sébastien Coron

Université du Luxembourg

October 10, 2009

Plan du cours

- Traitement des chaînes sous UNIX.
- Scripts shell.

Découpage des chaines

- Les caractères ## permettent d'éliminer la plus grande chaîne en correspondance avec le motif, en partant du début.
  - ```
    $ x='babar'
    $ echo ${x##*ba}
    r
    ```
  - ```
    $ y='hello.txt'
    $ echo ${y##*.}
    txt
    ```
- Le caractère # élimine la plus courte.
  - ```
    $ x='babar'
    $ echo ${x#*ba}
    bar
    ```
    ```

Découpage des chaînes

- % élimine la plus courte correspondance en partant de la fin, et %% élimine la plus longue.
  - ```
    $ y='hello.tar.gz'
    echo ${y%.*}
    hello.tar
    ```
  - ```
    $ y='hello.tar.gz'
    echo ${y%%.*}
    hello
    ```

Découpage des chaînes

- `${string:position}`
  - Extrait la sous-chaîne de `string` à partir de `$position`.
  - ```
    $ y='bonjour'
    echo ${y:3}
    jour
    ```
- `${string:position:length}`
  - Extrait `$length` caractères de la sous-chaîne de `string` à partir de `$position`.
  - ```
    $ y='bonjour'
    echo ${y:3:2}
    jo
    ```

La commande `cut`

- La commande `cut`
  - Permet de sélectionner certaines parties des lignes d'un fichier ou de l'entrée standard (si aucun fichier n'est précisé).
  - `cut [-c] [-f] list [-n] [-d delim] [-s] [file]`
- Options:
  - `-c list`: spécifie les caractères à sélectionner.
    - `-c2-5` sélectionne les caractères 2 à 5 de chaque ligne.

La commande `cut`

- Options:
  - `-f list`: sélectionne pour chaque ligne les champs spécifiés, les champs étant délimités par un caractère délimiteur.
    - `-f1,5` sélectionne les champs 1 et 5.
  - `-d delim`: spécifie le caractère délimiteur.
  - `-s`: supprime les lignes sans caractère délimiteur.
- `list`:
  - Liste de nombres séparés par une virgule, avec - pour indiquer un intervalle.
  - `1,2,3,5` ou `1-3,5`

Exemples

- Exemples:
  - `$ echo "hello" | cut -c 2-4`
    `ell`
  - `$ echo "he ll o wo" | cut -f2,3 -d' '`
    `ll o`

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`).

Execution conditionnelle

- Il est possible d'exécuter une suite d'instruction sous une certaine condition.
  - ```
    if [ test ]
    then
      commande-si-vrai
    else
      commande-si-faux
    fi
    ```
- Le test peut être une condition sur un fichier, ou une comparaison de chaine ou de valeur.
  - Ne pas oublier les espaces dans `if [ test ] then`
    ```

Conditions sur les fichiers

- `[ -s file ]`: vrai si le fichier existe et est non vide.
- `[ -f file ]`: vrai si c'est un fichier ordinaire.
- `[ -d file ]`: vrai si c'est un répertoire.
- `[ -r file ]`: vrai si le fichier est libre en lecture.
- `[ -w file ]`: vrai si le fichier est libre en écriture.
- `[ -x file ]`: vrai si le fichier est exécutable.

Example

- Afficher le contenu d'un fichier s'il existe.
  - ```
    #!/bin/bash
    if [ -f $1 ]
    then
      cat $1
    fi
    ```
- Créer un répertoire `toto` s'il n'existe pas déjà.
  - ```
    #!/bin/bash
    if [ !  -d toto ]
    then
      mkdir toto
    fi
    ```