

IF 110 : Système d'exploitation - TP2

Claire Capdevielle - Rémi Giraud
ccapdevi@labri.fr, remi.giraud@labri.fr

2015-2016

1 Rappels

1.1 L'interpréteur de commandes

Le shell est un interpréteur de commandes qui :

- initialise l'environnement, notamment en exécutant le script `~/ .bashrc` (voir la seconde partie du TP)
- génère le prompt.

Quand une commande est validée, le shell :

1. effectue les substitutions de variables,
2. interprète les métacaractères
3. gère les redirections et les pipes,
4. effectue les substitutions de commandes,
5. exécute la commande.

C'est le mécanisme d'évaluation du shell. **Ce n'est pas ce qui est saisi qui sera exécuté mais le résultat de l'évaluation de l'expression.**

1.2 Commandes

Une commande est constituée d'un nom suivi d'arguments.

`$ commande [options] [arguments]`

On différencie deux types de commandes :

- Externes au shell : ce sont des exécutables placés dans `/bin`, `/sbin`, `/usr/bin`, `/user/sbin`, etc. Elles sont exécutées dans un sous processus. **Comme les exécutables peuvent se trouver dans différents répertoires, le shell doit savoir où les chercher. C'est la variable "PATH" qui définit la liste des répertoires à scruter et l'ordre dans lequel doit faire sa recherche.**
- Internes au shell : elles font partie intégrante de l'interpréteur de commandes, elles ne sont pas exécutées dans un processus fils.

Remarque: Toutes les commandes Unix distinguent les majuscules des minuscules. Par exemple `"ls"`, `"Ls"`, `"IS"` et `"LS"` sont quatre mots différents au niveau de l'interpréteur de commandes.

2 Execution d'une commande

2.1 Notion de variable

Sous votre shell, il est possible de définir des variables. Par exemple sous votre environnement est défini une variable `DISPLAY` contenant le nom de votre terminal.

Question 1 Exécutez les commandes `echo DISPLAY` et `echo $DISPLAY`. Qu'en déduisez-vous ? Comment accède-t-on au contenu d'une variable ?

Question 2 Affichez le contenu de la variable `FOO`.

Question 3 Exécutez la commande `FOO=DISPLAY`. Affichez le contenu de la variable `FOO`.

Question 4 Exécutez la commande `FOO=$DISPLAY`. Affichez le contenu de la variable `FOO`.

2.2 Recherche et lancement d'une commande

Question 5 Affichez le contenu de la variable `PATH`. Sauvegardez cette valeur dans une variable `PATHSVG` et vérifiez la valeur sauvegardée. Mettre la chaîne vide dans `PATH` et essayez les commandes `cd`, `pwd`, `ls`. Que peut-on en conclure ? Comment lancer `ls` sans modifier `PATH` ?

Modifiez la variable `HOME` en lui affectant le chemin absolu vers l'un de vos répertoires (autre que votre répertoire personnel). Observez ce que produit la commande `cd` sans argument. **Restaurer les valeurs de `PATH` et de `HOME`.**

3 Personnalisation de Bash

A chaque lancement d'un nouveau shell Bash, le script `~/.bashrc` est lancé. Celui-ci permet de configurer l'environnement Bash.

Lisez le fichier `~/.bashrc`.

Vous verrez qu'il exécute les scripts `.bash_alias` et `.bash_export`.

3.1 Alias

Il est possible de définir des **alias** sous Unix permettant de personnaliser vos appels de commandes les plus utilisées. Les commandes à utiliser ici sont `alias` et `unalias`. Lancer `man alias`. Attention, ces commandes ne sont valables seulement pour le shell courant. Si vous voulez que vos alias soient permanents, insérez les dans `.bashrc` ou `.bash_alias`.

Question 6 Affichez la liste des alias définis dans le shell courant.

Question 7 Définissez un alias dont le nom est `date` sur la commande `ls`.

Question 8 Détruisez l'alias précédent et rappelez la commande `date` afin d'en vérifier son bon fonctionnement.

Question 9 Ouvrez le fichier `.bash_alias` se trouvant dans votre répertoire d'accueil. Rajoutez les alias suivants à l'intérieur de ce fichier, à la suite des alias déjà existant :

```
alias rm='rm -i'
alias ll='ls -la'
```

Question 10 Ajoutez également un alias permettant d'avoir le man en anglais directement.

Pour que ces modifications soient prises en compte par bash dès maintenant, tapez la commande `source .bash_alias`.

3.2 Variables d'environnement

Les systèmes Unix utilisent des variables réservées pour la configuration de l'environnement.

Pour modifier une variable d'environnement, utilisez la syntaxe suivante : `export VAR=valeur`. La variable `PATH` donne la liste des répertoires où aller chercher les commandes. Vous pouvez voir son contenu avec la commande : `echo $PATH`

Si vous voulez, par exemple, que les fichiers qui sont dans le répertoire `bin` de votre répertoire d'accueil soient directement accessibles, ajoutez le chemin dans la variable `PATH` comme suit : `export PATH=$PATH:$HOME/bin`

Vous pouvez personnaliser l'affichage du prompt (en y indiquant le chemin du dossier courant par exemple) en editant la valeur `PS1` dans votre `.profile` ou `.bashrc` (ou `.bash_profile`). Un prompt secondaire Prompt String 2 (`PS2`) est utilisé lorsque la commande en cours a besoin de plusieurs ligne.

Question 11

- Exécutez `cat abc \`. A quoi sert le `\` après une commande ?
- Éditez ou ajoutez la ligne suivante dans votre `.bashrc`
`PS2="Anything else to write?"`
- Exécutez `cat abc \`

4 Mécanismes de substitutions

!!!!Attention!!!!

Le caractère backquote ' (au clavier touches `altGR +7`) est différent du caractère quote ' (au clavier touche 4)

Interprétez et commentez le résultat des commandes suivantes.

- | | |
|---|--|
| 1. Substitution des accolades | <code>\$ echo \$(ls ~/Textes)</code> |
| <code>\$ echo a{ab,c,edf}f</code> | |
| <code>\$ mkdir ~/Textes/a{1,2,5}</code> | 5. Substitution arithmétique |
| | <code>\$ echo p\$((3+4))q</code> |
| 2. Substitution du caractère tilde | 6. Substitutions des métacaractères |
| <code>\$ echo ~/bar</code> | <code>\$ cd</code> |
| <code>\$ echo ~ccapdevielle</code> | <code>\$ echo */*</code> |
| 3. Substitution de variables | <code>\$ echo ../?[r-u]*</code> |
| <code>\$ ab=totot</code> | <code>\$ echo ../?[x-z]?</code> |
| <code>\$ echo \$ab</code> | <code>\$ echo ../?[~x-z]*?</code> |
| <code>\$ echo \${ab}ababab</code> | 7. Substitutions et exécution de commandes |
| <code>\$ echo \$abababab</code> | <code>\$ x=p; y=\${\${x}wd}; echo cwd=\$y</code> |
| <code>\$ y=p ; \${y}wd</code> | <code>\$ x=p; y='\${x}wd'; echo cwd=\$y</code> |
| 4. Substitution de commandes | |
| <code>\$ echo 'ls -l'</code> | |

4.1 Quotations

1. D  duire le r  le des quotes et doubles quotes
2. Expliquer le comportement des commandes

```
$ x=p; y='${x}wd'; echo $y
$ echo "*/" *
$ echo "*" $(pwd)"
```

```
$ A=$(echo *)
$ echo '$A'
$ echo "$A"
$ echo $A
$ $A
$ "$A"
```

Question 12 Faites afficher les cha  nes suivantes :

```
***_BONJOUR_***
_Le_caractere_'*'
/Fichier_d'entree\
*_Mon_Jan_20_17:59:48_EET_1992_* (la date doit   tre le r  sultat d'une ex  cution de
date)
$_Mon_Jan_20_18:07:15_EET_1992_$$_ (idem)
$6248$ (o   le nombre entre $ est le pid du shell)
(_bin/cat_bin/tar_) c'est-  dire la liste de fichiers dans le r  pertoire /bin dont le
nom est    trois lettres et dont la deuxi  me lettre est une de lettres a-d (utiliser les
m  tacaract  res).
```

Question 13

Dans votre r  pertoire courant, cr  ez    l'aide d'une seule commande les fichiers suivants : "fich1, fich2, fich11, fich12, fich1a, ficha1, fich33, .fich1, .fich2, toto, afich". Donnez les commandes permettant de lister les fichiers :

1. dont les noms commencent par fich,
2. dont les noms commencent par fich suivi d'un seul caract  re,
3. dont les noms commencent par fich suivi d'un chiffre,
4. dont les noms commencent par .,
5. dont les noms ne commencent pas par f,
6. dont les noms contiennent fich,
7. dont les noms ne contiennent pas les chiffres 2 et la lettre a.