Aide-mémoire Shell

Commandes de base

pwd permet de connaître le chemin absolu du répertoire courant (celui dans lequel on se trouve).

cd [chemin] positionne le répertoire courant à "chemin". Si aucun chemin n'est précisé, on se retrouve dans le "home_directory", répertoire initial de l'utilisateur.

ls [opt] [sel] permet de visualiser la liste des fichiers contenus dans le répertoire. Un certain nombre d'options peuvent être indiquées (opt). En voici quelques unes :

- **-t** ordonne les fichiers en fonction de leur date.
- **-l** affiche un certain nombre d'informations relatives au fichier (sa taille, sa date...).
- **-a** visualise tous les fichiers, y compris les fichiers cachés (sous Unix, les fichiers cachés commencent par le caractère ".").

on peut, à l'aide des méta-caractères (*, ?), sélectionner un sous-ensemble des fichiers contenus dans le répertoire en précisant les critères que leur nom doit vérifier (sel). Par exemple, "w*xx" (fichier commençant par w et terminant par xx).

Redirection des E/S

> redirige tout ce qui est affiché par une commande dans un fichier. "ma_commande > xx" inscrira dans xx tout ce que ma_commande a affiché.

>> même chose que ">" à ceci près que, si le fichier indiqué existait déjà le résultat de la commande est inscrit *à la fin* du fichier dans l'état précédent la commande.

< prend le contenu d'un fichier et le considère comme provenant du clavier (redirection des entrées). "cmd < xx" revient à faire en sorte que la commande cmd, si elle pose des questions à l'usager, prenne les réponses dans le fichier xx.

Meta-caractères

***** signifie "n'importe quelle séquence de caractères".

? signifie "n'importe quel caractère" non vide.

Caractères spéciaux

" une chaîne entre guillemets est "protégée" les espaces sont donc considérés comme un caractère et non comme un séparateur. Seul les variables ainsi que les méta-caractères ***** et **?** sont interprétés.

' une chaîne entre apostrophes est "constante". La protection est la même que pour les chaînes protégées par des guillemets mais les variables, ainsi que les caractères * et ? ne sont pas interprétés.

` une chaîne entre "backquotes" est considérée comme une commande shell qui est interprétée. C'est le résultat de cette commande qui est affecté à la variable.

| c'est un "tube"de communication entre deux processus.

\ il s'agit du caractère de protection de n'importe quel caractère. Par exemple, "*" est interprété comme le caractère "*" et non comme le méta-caractère "*". "\\" correspond au caractère "\".

& à la fin d'une commande, permet son exécution en "arrière plan"

Variables spéciales dans un script shell

$ # indique le nombre de paramètres de la ligne de commande.

$ n avec n compris entre 0 et 9, indique la valeur du nième paramètre. 0 correspond au nom de la commande invoquée. $n correspond à la chaîne vide s'il n'y a pas n paramètres.

$ * donne l'ensemble des âramètres

shift [n] décale les paramètres d'un cran (pas de valeur précisée) ou bien de "n crans" (si n est précisé). Attention: comportement non standard si le nombre de paramètres n'est pas suffisant.

Structures de contrôle dans un script shell

tests la syntaxe d'une commande de test est la suivante (les mots clefs sont indiqués en gras) :

```
if [ un_test ]
then
    sequence d'instructions
elif [ un_test ]
then
    sequence d'instructions
else
    sequence d'instructions
fi
```

Les tests suivants sont possibles (e1 et e2 sont des expressions numériques, ch1 et ch2 des expressions alphanumériques) :

e1 < e2	e1 **-lt** e2
e1 > e2	e1 **-gt** e2
e1 ≤ e2	e1 **-le** e2

e1 ≥ e2	e1 **-ge** e2
e1 = e2	e1 **-eq** e2
e1 ≠ e2	e1 **-ne** e2
ch1 = ch2	ch1 **=** ch2
ch1 ≠ ch2	ch1 **!=** ch2
ch1 vide	**-z** ch1
ch1 est le nm d'un fichier	**-f** ch1
ch1 est le nom d'un répertoire	**-d** ch1
négation de *condition*	! *condition*
ou	[condition1] **||** [condition2]
et	[condition1] **&&** [condition2]

ATTENTION: attention, vous devez avoir un espace entre chaque élément. Par exemple, [$x=$y] ne marchera pas alors que [$x = $y] est correct. Je conseille même de mettre les références à des variables entre guillemets car, si elle n'a pas de valeur, "" est interprété comme la chaîne vide par shell (qui la reconnaît). Si la variable n'est pas entre guillemets, ça risque de planter.

En clair, écrivez plutôt des choses du style ["$x" = "$y"].

boucle "for" la syntaxe d'une boucle "for" est la suivante :

for nom_var **in** liste de valeurs
do
    sequence d'instructions
done

boucle "while" la syntaxe d'une boucle "while" est la suivante (les tests se font de la même manière que pour l'instruction "if") :

while [un_test **]**
do
    sequence d'instructions
done

boucle "repeat" la syntaxe d'une boucle "repeat" est la suivante (les tests se font de la même manière que pour l'instruction "if") :

until [un_test **]**
do
    sequence d'instructions
done

"case" la syntaxe d'une série d'alternatives de type "case" est la suivante:

case var in
    profil) sequence d'instructions**;;**
esac

ou "profil" est une séquence de caractères pouvant contenir des méta-caractères comme "*3" ou "?". La séquence d'instructions qui suit doit forcément se terminer par les caractères "; ;".

Quelques commandes shell

echo	provoque l'affichage des arguments à l'écran
cat fich	affiche à l'écran le contenu du fichier "fich".
read var	effectue une saisie d'une ligne au clavier et range le résultat dans la variable var.
wc [opt] fich	permet de compter des entités de <fich>. Si opt vaut "-l", le nombre de lignes du fichier sera rendu. Si opt vaut "-c", le nombre de caractères dans le fichier sera rendu.
grep str fich	vérifie que la séquence de caractères str est présente dans le fichier fich. Si tel est le cas, elle rend la (ou les) ligne(s) contenant l'expression. Dans les expressions ainsi définies, on peut utiliser le caractère spécial ^qui signifie "début de ligne" et $ qui signifie "fin de ligne". On peut également, en insérant l'argument "-v" avant l'argument str, afficher les lignes ne contenant pas le profil recherché.
expr expression	calcule l'expression arithmétique indiquée en argument. Attention, ne travaille qu'en arithmétique entière.
cp f1 f2	effectue la copie du fichier "f1" dans le fichier "f2". Si "f2" existait déjà, il est détruit.
mv f1 f2	déplace le fichier "f1" dans le fichier "f2". Si "f2" existait déjà, il est détruit. Après l'opération, "f1" n'existe plus.
mkdir dir	crée le répertoire "dir" dans le répertoire courant.
rm f1	détruit le fichier "f1".
rmdir dir	détruit le répertoire "dir".
exit	interrompre l'exécution d'un script.
tr seq1 seq2	lit depuis stdin et retranscrit dans stdout en fonction de seq1 et de seq2.
touch fichiers	change la date de modification des fichiers donnés en paramètre. Si les fichiers n'existent pas, les crée.
set -x	toute commande suivant cette instruction sera tracée dans stderr.