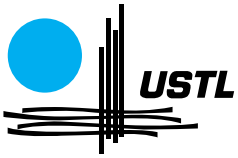

Licence d'informatique
Module de C/Unix

Travaux Pratiques d'initiation à la programmation en shell

Ce TP a pour but de vous familiariser avec la programmation en shell sur système Unix.

Les shells (ou langages de commandes) assurent l'interface entre les utilisateurs et le système. Sous Unix, il s'agit de véritables langages de programmation interprétés. On peut distinguer les commandes externes, qui entraînent la création d'un nouveau processus correspondant à l'exécution d'un programme binaire et les commandes internes, qui sont des commandes propres à l'interpréteur de commande.

L'interpréteur de commandes Bourne-shell Le Bourne-shell est le premier langage de commandes qui ait été développé sous UNIX. Vous utilisez actuellement le C-shell (`/bin/csh`). Le C-shell est particulièrement intéressant pour son mode interactif. Le Bourne-shell est plus adapté que le C-shell pour la programmation et reste le shell le plus utilisé pour programmer. Référez vous au chapitre correspondant du polycopié de cours. Pour réaliser la suite du TP, il faut lancer un Bourne-shell par la commande `sh`.

Commande `echo` Essayez les commandes suivantes :

```
echo Bonjour
echo Bonjour a tous !
```

Visualisation des valeurs des variables de l'interpréteur Essayez les commandes suivantes :

```
set
echo $HOME
echo $PS2
```

Affectation de valeur à une variable Essayez les commandes suivantes :

```
x=a
echo $x
set
x='123 4567 89'
echo $x
echo $y
```

```
echo ${x}y
date
d=`date`
echo $d
liste= `ls`
echo $liste
```

Expansion des chaînes de caractères Essayez les commandes :

```
echo 'dddd $HOME dd'
echo "dddd $HOME dd"
```

Exportation des variables Vérifiez qu'une variable n'est pas automatiquement exportée dans un sous-interpréteur de commandes. Pour l'exporter, il faut utiliser la commande `export` :

```
sh
set                                  # pas de variable x
CTRL-D                               # sortie du sous interpreteur
export x
sh
set                                  # la variable x est bien exportee
CTRL-D
```

Id : tpshell.tex,v 1.6 2000/11/28 18 :34 :55 marquet Exp

Boucle `for` Essayez les commandes :

```
for i in a ee dx; do echo Hi! ; done
```

Vérifiez qu'en écrivant la commande `for...done` sur plusieurs lignes, l'interpréteur attend la suite de la commande ; ce qui est materialisé par ¡¡ l'invite secondaire ¿¿ > (voir la variable `$PS2`). Vous devez obtenir quelque chose comme :

```
$ for i in aa ee dx
>do echo Hi!
>done
Hi!
Hi!
Hi!
$
```

Fichier de commandes Faites de la commande `for...done` précédente un fichier de texte, par exemple `cmd.sh`. Essayez d'exécuter `cmd.sh` :

```
cmd.sh
```

C'est impossible ; vous ne possédez pas le droit d'exécution sur le fichier de commandes `cmd.sh` (vérifiez cela par un `ls -l cmd.sh`). Deux solutions s'offrent à vous.

1. Passer votre fichier de commandes en paramètre au Bourne-shell

   ```
   sh cmd.sh
   ```

2. Changer les droits du fichier de commandes :

   ```
   chmod 755 cmd.sh
   ```

   ou

   ```
   chmod +x cmd.sh
   ```

   Vérifiez le résultat par les commandes :

   ```
   ls -l cmd.sh
   cmd.sh
   ```

Les fichiers de commandes peuvent être exécutés à partir du C-shell. Comme le C-shell possède également un langage de commandes, il faut indiquer à la première ligne du fichier le shell à utiliser :

```
#! /bin/sh
```

Le caractère # sert habituellement à introduire un commentaire.

Retour sur la boucle `for` Remarquez l'usage possible en lecture de variable d'itération d'une commande `for`, par exemple :

```
for i in 1 2 'a b' 4
do
echo $i
done
```

ou

```
for i in *
do
echo $i
done
```

Commande `read` Essayez la commande `read`, en constituant un fichier de commandes du style :

```
echo '?'
read REP
echo "reponse $REP"
```

Construction `case` Ecrivez un fichier de commandes `choix.sh`, qui demande interactivement à l'usager s'il veut afficher la date ou le répertoire de travail, analyse la réponse et exécute l'action demandée. Exemple :

```
$ choix.sh
(d)ate/(r)epertoire courant ? x
????
$ choix.sh
(d)ate/(r)epertoire courant ? r
/home/licence/duchmol/unix/tp2
$ choix.sh
(d)ate/(r)epertoire courant ?d
Mon Sep 24 18:21:29 MEST 2001
```

Arguments d'un fichier de commandes Utilisez les variables `$#` et `$*` et une boucle `for` pour développer une commande `arg.sh` qui écrit sur la sortie standard :

- le nombre d'arguments qui a été passé à la commande ;
- le nom de chemin du programme (c'est-à-dire le nom de la commande) ;
- puis chacun des arguments à raison d'un par ligne.

Exemple :

```
$arg.sh 1 arg $HOME
Nombre de chaines: 3
arg.sh
1
arg
/home/licence/duchmol
$
```