Aix*Marseille université
Initiative d'excellence
Année 2018-19

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 1 : TP 4

Systèmes d'exploitation 1 : programmation Shell

1 Scripts de démarrage

1.1 Commande personnelle

1.1.1 Remonter au dossier parent

Pour remonter dans l'aborescence des dossiers, on utilise la commande

```
$ cd ..
```

L'espace permet de distinguer la commande et le paramètre. Il est donc nécessaire. Cependant, les habitués du DOS, oublient souvent de le mettre car sous DOS, si la commande est la même, l'espace n'est pas nécessaire. Il arrive donc fréquemment qu'on l'oublie, ce qui entraine une erreur :

```
$ cd..
cd.. : commande introuvable
```

1.1.2 Exercice

- Ajouter un alias personnel `cd..` de la commande `cd ..`
- Ouvrir un nouveau terminal et tester la nouvelle commande
- Passer en superutilisateur et tester à nouveau la commande :

```
$ sudo su
[sudo] Mot de passe :
$ cd..
```

- Que se passe-t'il ? Faites en sorte que tous les utilisateurs puissent utiliser cet alias de commande.

1.2 Personnalisation de son invite de commande

1.2.1 Exercice

- Personnaliser votre invite de commande en ajoutant les lignes suivantes à votre fichier `.bashrc`

```
# \n pour pouvoir continuer la commande sur la ligne suivante
# $( ..) pour obtenir le résultat de l'exécution de la commande incluse
salut="Bonjour $USER, \n
Nous sommes le $(date) \n
Bon travail !"
# -e option indispensable pour interpréter les \n
echo -e $salut
```

- Ouvrir un nouveau terminal et observer.

2 Informations sur le système

2.1 Exercices

— Saisir la commande suivante. Quelles indications donne-t-elle ?

```
$ uname -a
```

Le dossier `/proc` contient une image du système en fonctionnement. Il contient notamment les informations relatives au processeur.

— Executer la commande suivante :

```
$ cat /proc/cpuinfo
```

— Quelle est le modèle de votre processeur ? Quelle est le format des adresses mémoire physiques et virtuelles ?

— Exécuter la commande suivante :

```
$ cat /proc/cpuinfo | grep "model name"
```

— Expliquer son fonctionnement.

Le pseudo-système de fichiers sysfs, introduit par le noyau Linux 2.6, est un système de fichiers virtuel monté dans le dossier `/sys`. Il n'occupe donc pas d'espace disque et sa taille est de 0 Ko :

```
$ ls -ld /sys
drwxr-xr-x 13 root root 0 août   3 10:26 /sys
```

Sysfs a été conçu pour exporter depuis l'espace noyau vers l'espace utilisateur des informations sur les périphériques et leurs pilotes.

La température du ou des coeurs de votre processeur est accessible dans le fichier `/sys/class/thermal/thermal_zone?/temp` ou dans les fichiers `/sys/devices/platform/coretemp.?/hwmon/hwmon?/temp?_input`. Pour connaitre quels sont les fichiers qui contiennent la température des coeurs, il faut lire les fichiers `temp?_label` :

```
$ grep "" /sys/devices/platform/coretemp.?/hwmon/hwmon?/temp?_label
/sys/devices/platform/coretemp.0/hwmon/hwmon0/temp1_label:Package id 0
/sys/devices/platform/coretemp.0/hwmon/hwmon0/temp2_label:Core 0
/sys/devices/platform/coretemp.1/hwmon/hwmon1/temp1_label:Package id 1
/sys/devices/platform/coretemp.1/hwmon/hwmon1/temp2_label:Core 0
```

On y trouve la température des coeurs et de chaque processeur. Mon ordinateur est virtualisé et j'ai deux processeurs, donc deux coeurs. Votre ordinateur fournira sans doute une sortie différente.

— Quelle la température du ou des coeurs de votre processeur ?

3 Périphériques matériels

Parmi les périphériques matériels les plus couramment utilisés, on compte les terminaux séries. Leurs pseudo-fichiers sont situés dans le dossier `/dev` et portent un nom qui commence par `ttyS` pour les voies séries matérielles (`ttyS0` = `COM1` sous Windows), `ttyUSBx` ou `ttyACMx` pour les périphériques sur voie série via une liaison USB. Pour mettre au point un programme qui devra communiquer par voie série avec un périphérique, il est parfois utile d'utiliser des ports séries virtuels. L'utilitaire `socat` permet de créer deux ports séries virtuels connectés entre eux.

3.1 Communication par voie série

3.1.1 Exercices

— Installer l'utilitaire `socat` et exécuter le :

```
$ sudo apt-get install socat
$ socat -d -d pty,rawer pty,rawer
```

— D'après sa documentation, que signifie les options qu'on lui a passé ? `http://www.dest-unreach.org/socat/doc/socat.html`
— Identifier les noms des pseudo-fichiers qui correspondent aux deux ports virtuels.
— Ouvrer un nouveau terminal et "écouter" un des deux ports
— Ouvrer un nouveau terminal et "envoyer" une chaine de caractères par le second port :

Terminal 1 :

```
$ cat /dev/pts/0
```

Terminal 2 :

```
$ echo "Des trucs à envoyer..." > /dev/pts/1
```

— Expliquer la commande exécutée dans le terminal 2.
— Pour arrêter l'écoute dans le terminal 1, utiliser la combinaison de touches `CTRL+C`.
— Créer un dossier nommé `logs` dans dossier `Documents`
— Relancer l'écoute et rediriger les données qui arrivent sur le terminal 1 vers un fichier texte nommé `pts0.log` et qui devra être situé dans le dossier `logs`.
— Depuis le terminal 2, continuer à envoyer du texte. Observer ce qui se passe dans le terminal 1, puis consulter le contenu du fichier `pts0.log`.

3.1.2 Et avec python ?

Après tout, ce ne sont que des fichiers texte !
— Ecrire un script python qui envoie des données au terminal 1.
— Ecrire un script python qui reçoit les données envoyées depuis le terminal 2 jusqu'à ce que le caractère "q" soit reçu.

4 Ecriture de scripts

Dans l'exercice précédent, le terminal qui reçoit les données est bloqué en attente de données pour les afficher ou pour les redirigés dans le fichier log. On souhaiterait faire de cette opération une tâche qui pourrait s'exécuter en arrière plan. Par la même occasion, nous allons ajouter un champs d'horodatage et remplacer la donnée en entrée par la température moyenne du processeur.

4.1 Programmation shell

4.1.1 Les expressions arithmétiques :

Elle sont enfermées dans des doubles parenthèses :

```
$ echo $((2+7))
9
$ variable=1 # attention : pas d'espace !
$ ((variable++))
echo $variable
2
```

4.1.2 Structure d'une boucle :

```
while expression
do
    commandes
done
```

où `expression` est un booléen

4.1.3 Exercices :

— Ecrire un script shell qui envoi la température moyenne des coeurs de votre processeur sur un des ports série toutes les secondes en vous aidant du script ci-dessous à compléter.

```
#!/bin/bash`)
while true
do
    temp1=<température du coeur 1>
    temp2=<température du coeur 2>
    temp3=<température du coeur 3>
    temp4=<température du coeur 4>
    temp=<calcul de la température moyenne>
    echo "$temp °C" > <port pour envoi>
    sleep 1
done
```

— Ecrire un script shell qui ajoute au fichier `pts0.log` l'horodatage et la donnée arrivée à partir de l'extrait ci-dessous :

```
#!/bin/bash`)
while true
do
    read recu < <le port écouté ici>
    echo "<la date ici> ; <la ligne reçue ici>" >> <le fichier log ici>
done
```

— Exécuter les scripts. Consulter le contenu du fichier `pts0.log`
— Pour libérer les terminaux qui exécutent les scripts, utiliser la combinaison de touches `CTRL+C`
— Supprimer le fichier `pts0.log`.
— Pour exécuter les scripts en arrière plan dans le même terminal, faire suivre les commandes par le caractère `&`

```
$ ./script &
```

Le terminal retourne le `PID` du processus qui correspond à l'exécution du script.

— Renouveler l'expérience. Observer le fichier `pts0.log` se créer et se remplir.
— Pour stopper une tâche utiliser la commande `kill` suivie du `PID` du processus que l'on veut stopper.
— Relancer les sripts en arrière plan dans deux terminaux différents, puis fermer le terminal de réception. Le fichier `pts0.log` continue-t-il de se remplir ?

Le processus correspondant à notre script est un enfant du processus qui exécutait le terminal que l'on a fermé. La fin du processus père entraîne la fin de tous ses processus enfants.

4.1.4 Tester l'existence du dossier logs

Les données s'écrivent dans le fichier `pts0.log` dans le dossier `logs` parce qu'il existe. Mais que se passerait-il si ce dossier n'existait pas ?

— Relancer les deux processus puis supprimer le dossier `logs`.
— Que se passe-t-il ?

Nous allons tester l'existence du dossier et le créer s'il n'existe pas.

Structure de test :

```
$ test expression
```

ou

```
$ [ expression ]
```

Le résultat est contenu dans une variable spéciale `$?`. S'il est égal à 0, l'expression est `VRAI` sinon, elle est `FAUSSE`.

```
$ a=10
$ test $a -eq 10
$ echo $?
0
```

L'option `-eq` est l'égalité entre deux valeurs numériques.

Test sur les valeurs numériques :

option	signification
-eq	égal
-ne	différent
-lt	strictement inférieur
-gt	strictement supérieur
-le	inférieur ou égal
-ge	supérieur ou égal

Test sur les chaines de caractères :

option	signification
-z	la chaine est vide
-n	n'est pas vide
=	les chaines comparées sont identiques
!=	les chaines comparées sont différentes

Test sur les fichiers :

option	signification
-e	il existe
-f	c'est un fichier normal
-d	c'est un répertoire
-r	il est lisible
-w	il est modifiable
-x	il est exécutable
-s	il n'est pas vide

Structures conditionnelles

```
if expression
then
    # séquence exécutée si suite-de-commandes rend une valeur 0
    bloc-instruction1
else
    # séquence exécutée sinon
    bloc-instruction2
fi
```

4.1.5 Exercices

- Modifier le script de réception pour tester l'existence du dossier `logs` et le créer s'il n'existe pas.
- Modifier le script de réception pour créer un nouveau fichier toutes les minutes (chaque fichier devrait contenir 60 lignes). Le nom du fichier doit être de la forme : `pts0-201922062144.log` où `201922062144` représente l'année, le mois, le jour, l'heure et les minutes.

Indices :

```
$ wc -l < nom_fichier   # nombre de lignes dans un fichier

$ t=$(date +%Y%m%d%H%M)# formater une date
$ echo $t
```