

1 Scripts de démarrage

1.1 Commande personnelle

1.1.1 Remonter au dossier parent

1.1.2 Exercice

- Ajouter un alias personnel `cd..` de la commande `cd ..`

```
$ nano ~/.bashrc # ou vi ~/.bashrc ou gedit ~/.bashrc
...
# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
# Ajouter ici :
# My personal cd alias like DOS
alias cd..='cd ..'
...
```

- Ouvrir un nouveau terminal et tester la nouvelle commande

```
~$ cd..
/home$
```

- Passer en superutilisateur et tester à nouveau la commande :

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

- Que se passe-t'il ? Faites en sorte que tous les utilisateurs puissent utiliser cet alias de commande.
 - pour être accessible à tous les utilisateurs, l'alias doit être situé dans le fichier `/etc/bash.bashrc` :

```
$ nano /etc/bash.bashrc # ou vi /etc/bash.bashrc ou gedit /etc/bash.bashrc
...
alias cd..='cd ..'
```

1.2 Personnalisation de son invite de commande

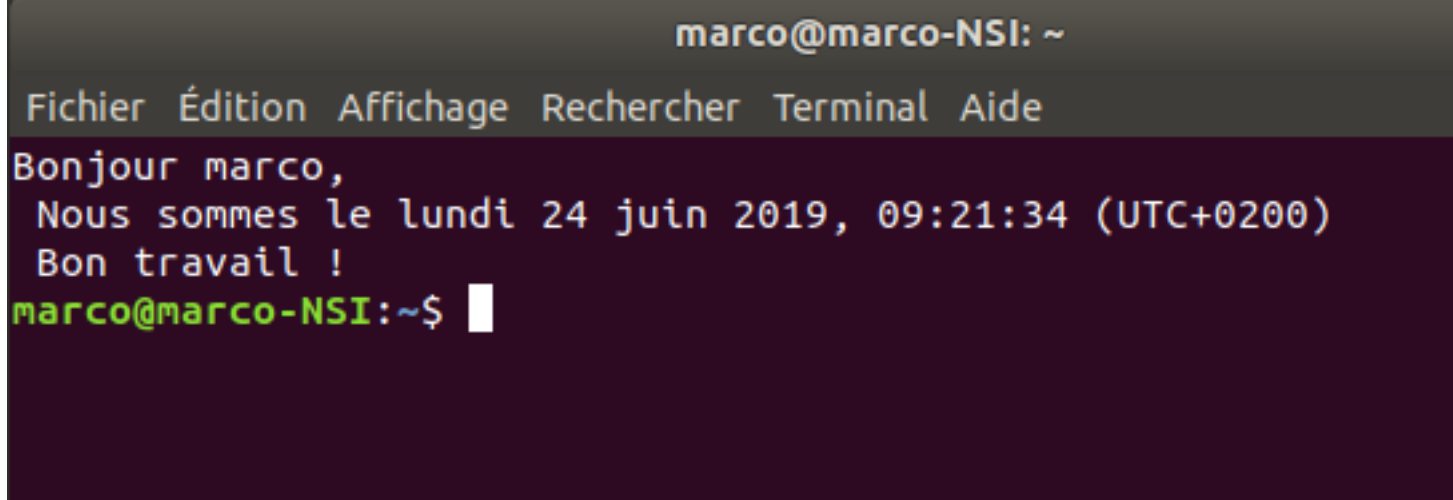
1.2.1 Exercice

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

```
$ nano ~/.bashrc # ou vi ~/.bashrc ou gedit ~/.bashrc
```

```
...  
# Ajouter en fin de fichier :  
# \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.



```
marco@marco-NSI: ~  
Fichier Édition Affichage Rechercher Terminal Aide  
Bonjour marco,  
Nous sommes le lundi 24 juin 2019, 09:21:34 (UTC+0200)  
Bon travail !  
marco@marco-NSI:~$
```

2 Informations sur le système

2.1 Exercices

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

```
$ uname -a  
Linux marco-NSI 4.18.0-22-generic #23~18.04.1-Ubuntu SMP Thu Jun 6 08:37:25 UTC 2019  
x86_64 x86_64 x86_64 GNU/Linux
```

```
$ uname --help # ou man uname
```

- La commande `uname -a` permet d'obtenir toutes les informations à propos du noyau :
 - le nom du noyau : `Linux`
 - le nom du nœud réseau (hostname) : `marco-NSI`
 - la version du noyau : `4.18.0-22-generic`
 - la version de la distribution : `#23~18.04.1-Ubuntu SMP Thu Jun 6 08:37:25 UTC 2019`
 - afficher le nom de matériel de la machine : `x86_64`
 - le système d'exploitation : `GNU/Linux`

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

— Exécuter la commande suivante :

```
$ cat /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 58
model name    : Intel(R) Core(TM) i7-3720QM CPU @ 2.60GHz
stepping      : 9
microcode     : 0x20
cpu MHz       : 2594.159
...
```

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

```
model name    : Intel(R) Core(TM) i7-3720QM CPU @ 2.60GHz
address sizes  : 42 bits physical, 48 bits virtual
```

- Exécuter la commande suivante :

```
$ cat /proc/cpuinfo | grep "model name"
model name    : Intel(R) Core(TM) i7-3720QM CPU @ 2.60GHz
```

- Expliquer son fonctionnement.
 - le résultat de la commande `cat /proc/cpuinfo` est passé en argument de la commande `grep "model name"`
 - On a ici utilisé un pipe (tube) entre deux processus qui correspondent à l'exécution des commandes.
 - Le résultat est la recherche de la chaîne "model name" dans le contenu de le fichier `cpuinfo`

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`

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

```
$ cat /sys/class/thermal/thermal_zone0/temp
44915
$ cat /sys/class/thermal/thermal_zone1/temp
45255
```

La température est donnée en millièrne de degrés Celcius

3 Périphériques matériels

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>
- `-d -d` : Prints fatal, error, warning, and notice messages.
- `pty` : Establishes communication with the sub process using a pseudo terminal instead of a socket pair.
- `rawer` : Makes terminal rawer than raw option.
- Identifier les noms des pseudo-fichiers qui correspondent aux deux ports virtuels.

```
$ socat -d -d pty,rawer pty,rawer
2019/06/24 10:22:16 socat[5048] N PTY is /dev/pts/0
2019/06/24 10:22:16 socat[5048] N PTY is /dev/pts/1
2019/06/24 10:22:16 socat[5048] N starting data transfer loop with FDs [5,5] and [7,7]
```

Les ports virtuels sont ici `/dev/pts/0` et `/dev/pts/1`

- Ouvrir un nouveau terminal et "écouter" un des deux ports
- Ouvrir un nouveau terminal et "envoyer" une chaîne 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.
- la chaîne de caractère est redirigé du flux standard de sortie vers le fichier `/dev/pts/1`
- Pour arrêter l'écoute dans le terminal 1, utiliser la combinaison de touches `CTRL+C`.
- Créer un dossier nommé `logs` dans dossier `Documents`

```
$ mkdir ~/Documents/logs
```

- 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`.

```
$ cat /dev/pts/0 > ~/Documents/logs/pts0.log
```

- 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`.
- Dans le terminal 1, plus rien ne s'affiche.
- On ouvre le fichier `pts0.log` qui s'est créé dans `~/Documents/logs` et on trouve ce que l'on a envoyé.
- Fermer puis ré-ouvrir ou recharger le fichier pour voir les nouvelles données envoyées

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.

```
f = open('/dev/pts/1', "w")
f.write("Des trucs envoyés depuis un script python")
f.close()
```

- Ecrire un script python qui reçoit les données envoyées depuis le terminal 2 jusqu'à ce que le caractère "q" soit reçu.

```
port = "/dev/pts/0"
f = open(port, "r")
recu = ""
while(recu!="q"):
    recu = f.readline()[:-1]
    print(f'reçu sur {port} : {recu}')
f.close()
```

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 :

4.1.2 Structure d'une boucle :

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=$(cat /sys/class/thermal/thermal_zone0/temp)
    temp2=$(cat /sys/class/thermal/thermal_zone1/temp)
    temp3=$(cat /sys/class/thermal/thermal_zone2/temp)
    temp4=$(cat /sys/class/thermal/thermal_zone3/temp)
    temp=$((temp1+temp2+temp3+temp4)/4000)
    echo "$temp °C" > /dev/pts/1
    sleep 1
done
```

Pour l'exécuter, le nom du script peut être précédé de la commande interpréteur ou il doit posséder les droits d'exécution :

```
$ bash nom_script
```

ou

```
$ chmod +x nom_script
$ ./nom_script
```

- 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`
logdir="$HOME/Documents/logs"
logfile="pts0"
logext=".log"
file=$logdir/"$logfile$logext"

while true
do
    read recu < /dev/pts/1
    echo "$(date);$recu" >> $file
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.

```
$ ./listenpts0.0 &
[2] 11696
$ kill 11696
$
[2]+  Complété          ./listenpts0.0
```

- Relancer les scripts en arrière plan dans deux terminaux différents, puis fermer le terminal de réception. Le fichier `pts0.log` continue-t-il de se remplir ?
 - Les données sont toujours envoyées depuis le terminal d'envoi mais le fichier `pts0.log` ne se remplit plus. La fermeture du terminal a mis fin à l'exécution du script qui tournait en arrière plan.

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 ?

- Stopper l'exécution du script d'envoi puis supprimer le dossier `logs`.
- Que se passe-t-il ?

```
listenpts0.0: ligne 11: /home/marco/Documents/logs/pts0.log: Aucun fichier ou dossier de ce type
```

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

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.

```
#!/bin/bash
logdir="$HOME/Documents/logs"
logfile="pts0-"
logdate=$(date +%Y%m%d%H%M)
logext=".log"
file=$logdir"/"$logfile$logdate$logext
if [ ! -e $logdir ]
then
    mkdir $logdir
    echo > $file
fi

while true
do
    nbligne=$(wc -l < $file)
    echo $nbligne
    if [ $nbligne -gt 60 ]
    then
        logdate=$(date +%Y%m%d%H%M)
        file=$logdir"/"$logfile$logdate$logext
    fi
    read recu < /dev/pts/0
    echo "$(date);$recu" >> $file
done
```