

## TP 3

### Exercice 1 : Protection des variables

On peut annuler la signification des caractères spéciaux comme \*, ?, #, |, [], {} en utilisant des caractères d'échappement, qui sont également des caractères génériques.

- \Antislash

L'antislash \, qu'on appelle le caractère d'échappement, annule le sens de tous les caractères génériques, en forçant le shell à les interpréter littéralement.

1. *Essayer le code suivant et comparer les résultats :*

```
echo \$var
```

```
echo "\$var"
```

```
echo $var
```

```
echo “$var”
```

```
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo \$var
$var
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo \"\$var\"
$var
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo $var
$var
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo “$var”
“$var”

Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ]]
```

- **" " Guillemets**

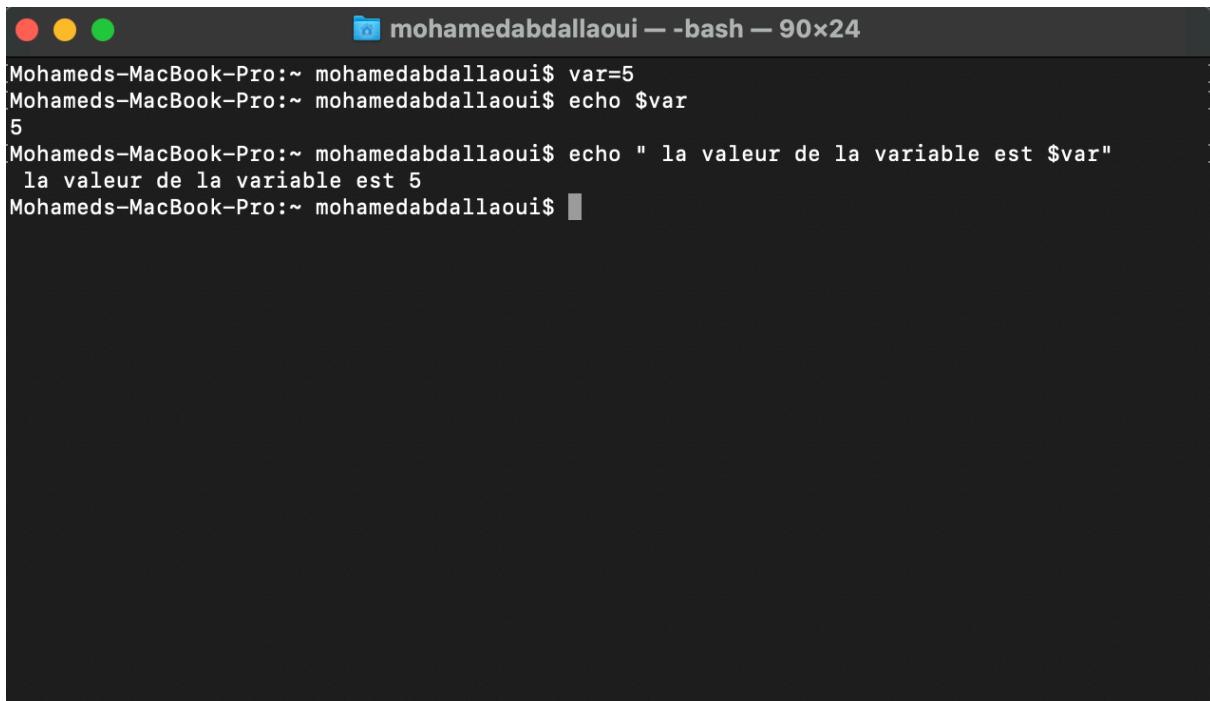
Les guillemets (doubles) " " sont les guillemets faibles mais annulent la plupart des métacaractères entourés à l'exception du tube (), de l'antislash () et des variables (\$var).

**2. Essayer le code suivant :**

```
var=5
```

```
echo la valeur de la variable est $var
```

```
echo "la valeur de la variable est $var"
```



```
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ var=5
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo $var
5
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo " la valeur de la variable est $var"
 la valeur de la variable est 5
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

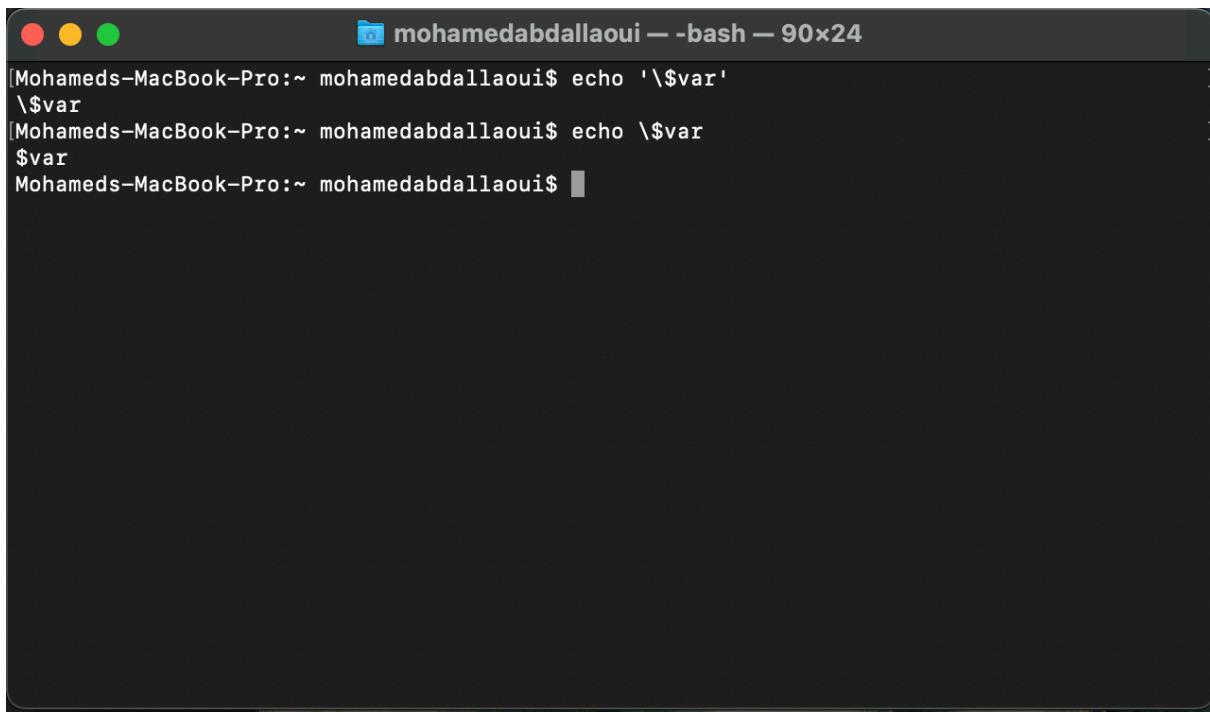
- **Apostrophes**

Les guillemets simples, ou apostrophes (' ') annulent le sens de tous les caractères génériques sauf l'antislash.

**3. Essayer le code suivant et comparer les résultats :**

```
echo '$var'
```

```
echo \$var
```



```
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo '\$var'
\$var
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo \$var
$var
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ]
```

## **Exercice 2 : Interaction utilisateur**

La commande echo pose une question à l'utilisateur.

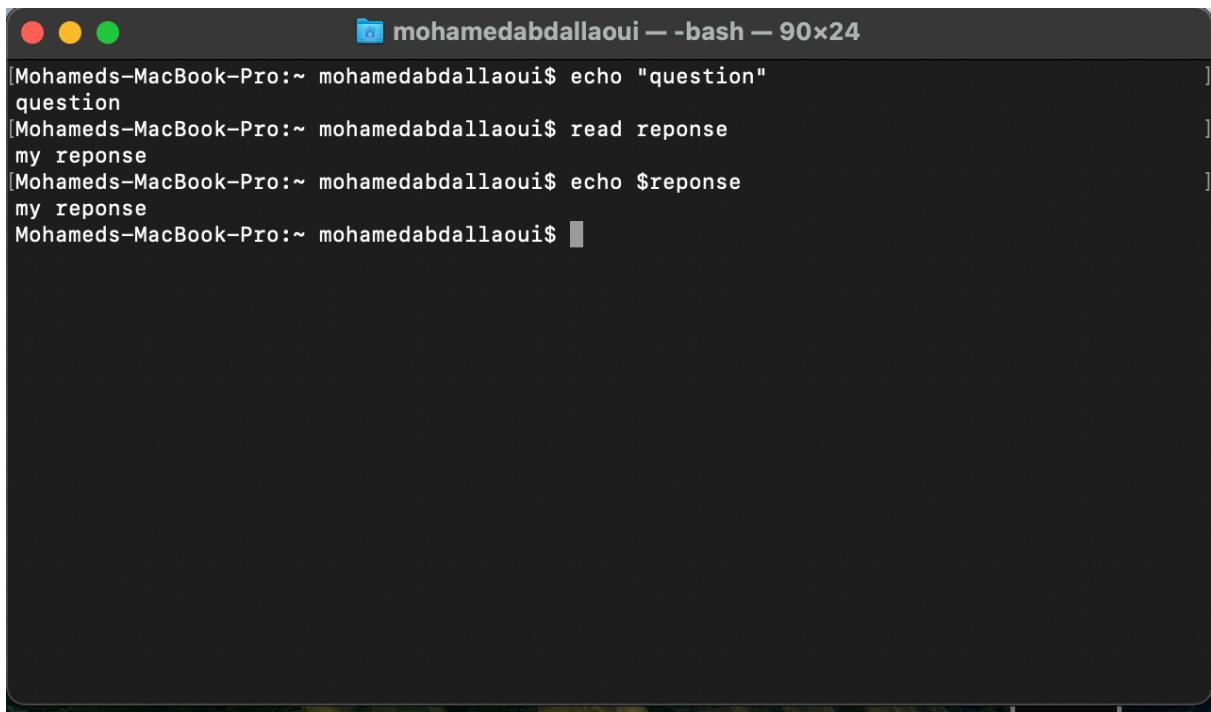
La commande read lit les valeurs entrées au clavier et les stocke dans une variable à réutiliser.

**1. Essayer le code suivant :**

```
echo "question"
```

```
read reponse
```

```
echo $response
```



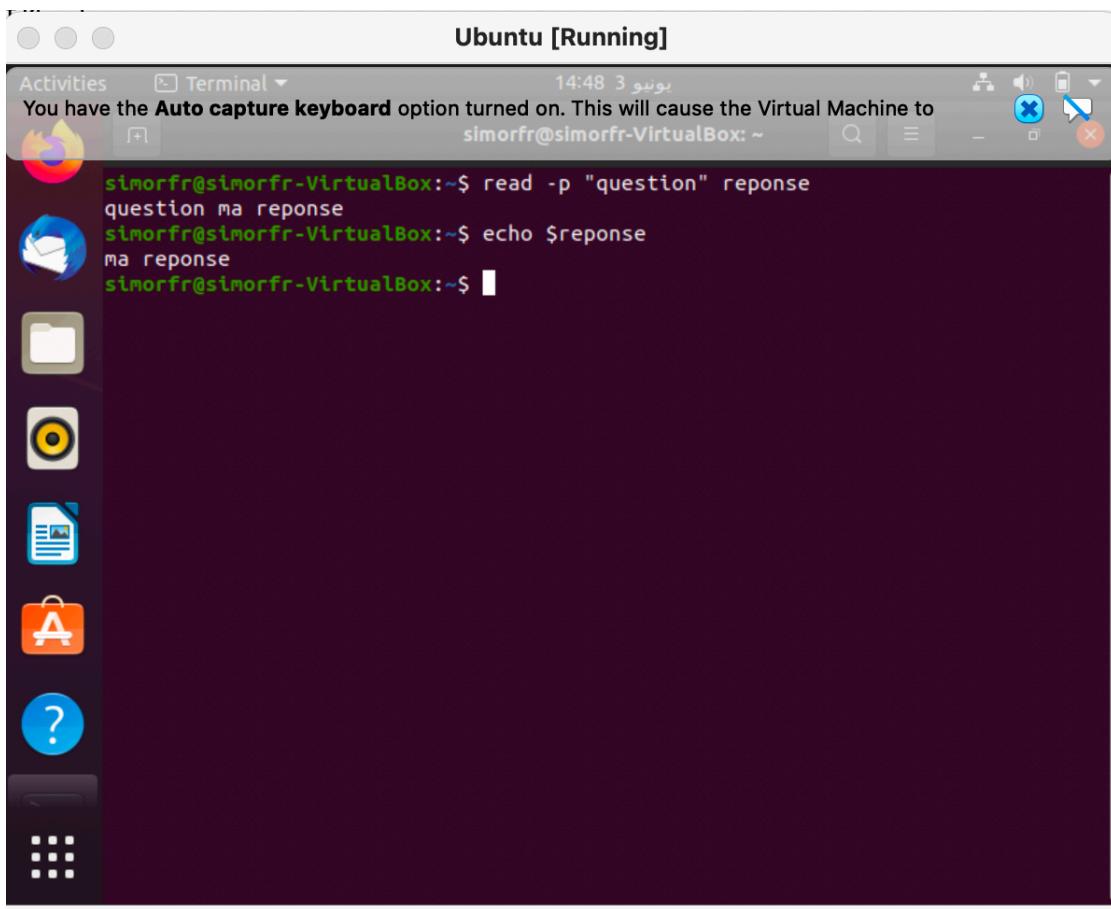
```
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo "question"
question
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ read reponse
my reponse
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ echo $reponse
my reponse
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ]]
```

On peut aller plus vite avec `read -p` qui sort du texte et attend une valeur en entrée :

**2. *Essayer le code suivant :***

```
read -p "question" reponse
```

```
echo $reponse
```



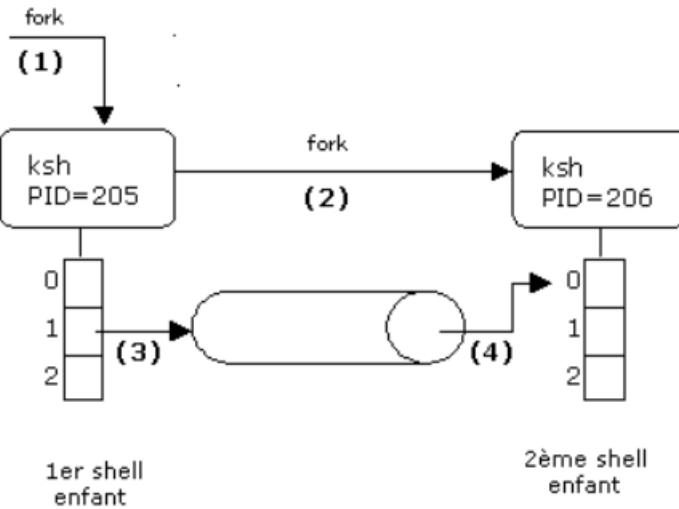
### Exercice 3 : Tubes de communication

Un tube (**pipe** en anglais) permet de faire communiquer deux processus. Le tube est représenté par une barre verticale (touches [Alt Gr] 6 sur un clavier AZERTY) située entre deux commandes Unix. Le résultat de la commande de gauche va partir dans le tube, tandis que la commande de droite va en extraire les données afin de les traiter.

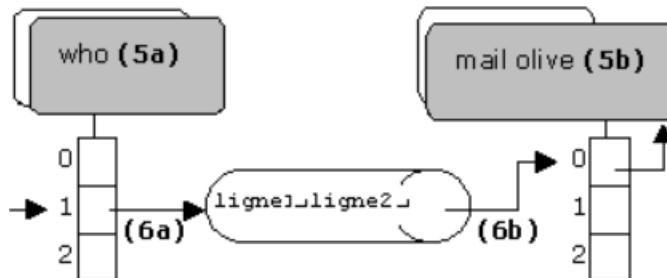
Les figures ci-dessous représentent le mécanisme interne associé au tube de communication.

```
$ who | mail olive
```

1ère étape: Mise en place du tube entre les 2 shells enfant



2ème étape: Chaque shell se remplace avec sa commande



Quelles que soient les commandes présentes de chaque côté du tube, le shell de travail détecte le caractère | sur la ligne de commande et va engendrer un shell enfant (1) qui, à son tour, fait de même (2). Le premier shell enfant (PID=205) dissocie sa sortie standard du terminal et la connecte sur l'entrée du tube (3). Le deuxième shell enfant (PID=206) dissocie son entrée standard du terminal et la connecte sur la sortie du tube (4).

Chaque shell enfant va se remplacer avec sa commande (5a et 5b). Chaque commande commence alors à s'exécuter. Lorsque la commande **who** écrit sur sa sortie standard, les messages partent dans le tube (6a). Parallèlement, la commande **mail** lit son entrée standard (6b), ce qui provoque l'extraction des données ...

### 1. *Essayer le code suivant et noter la différence :*

```
cat /etc/passwd | grep root
```

```

Activities Terminal ١٥:١٢ ٣ بوينو simorfr@simorfr-VirtualBox: ~
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
simorfr:x:1000:1000:mohamed,,,:/home/simorfr:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
simorfr@simorfr-VirtualBox:~$ cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
simorfr@simorfr-VirtualBox:~$ 
```

2. Quelle est la fonctionnalité de « grep »

Q2) Grep est une commande qui permet de faire des recherches de chaînes de caractères dans un flux de texte

3. Que fait la commande de la question 1 ?

Q3) la commande de la question 1 permet d'accéder à passwd et de rechercher la ou il y a "root" et de le détecter.

4. Essayer le code suivant :

```
cat /etc/passwd | grep root | tee /tmp/result
```

Q4) Résultat ci-dessous

5. Quelle est la fonctionnalité de « tee » ?

- Q5) La fonction de tee est qu'elle prend la sortie de PIP et l'utilise ensuite comme entré

```

Activities Terminal 15:31  يونيو 3 simorfr@simorfr-VirtualBox: ~
simorfr@simorfr-VirtualBox: ~$ kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/n/login
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
simorfr:x:1000:1000:mohamed,,,:/home/simorfr:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologin
simorfr@simorfr-VirtualBox:~$ cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/n/login
simorfr@simorfr-VirtualBox:~$ cat /etc/passwd | grep root | tee /tmp/result
bash: tee/tmp/result: No such file or directory
simorfr@simorfr-VirtualBox:~$ cat /etc/passwd | grep root | tee /tmp/result
root:x:0:0:root:/root:/bin/bash
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/n/login
simorfr@simorfr-VirtualBox:~$ cat /tmp/result
root:x:0:0:root:/root:/bin/bash
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/n/login
simorfr@simorfr-VirtualBox:~$ 
```

## 6. Quelle est la différence entre les deux commandes ?

### Exercice 4 : Regroupement de commandes

Le regroupement de commandes peut être utilisé pour :

- rediriger la sortie écran de plusieurs commandes vers un même fichier ou vers un tube ;
- faire exécuter plusieurs commandes dans le même environnement.

#### 1. Essayer les commandes suivantes :

Seule la sortie standard de la deuxième commande est redirigée dans le fichier **resultat**.

date ; ls > resultat

cat resultat

Q1) Résultat ci-dessous

```
mohamedabdallaoui -- -bash -- 80x24
Last login: Wed Jun 9 14:32:10 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ (date;ls)>resultat
[Mohameds-MacBook-Pro:~ mohamedabdallaoui$ cat resultat
Wed Jun 9 14:39:25 +01 2021
04_05_2021classes
1st C program on VS.c
AbdallaouiMohamed.html
Applications
COMEBackprog.c.code-workspace
Creative Cloud Files
Desktop
Documents
Downloads
Fibonaci_CSBS
GAMEfr.c .h>
IdeaProjects
Library
MinijeuBox.c
Movies
Music
```

Les parenthèses ( ) et les accolades { } permettent de regrouper les commandes.

### **Les parenthèses (cmde1 ; cmde2 ; cmde3)**

Avec les parenthèses, un shell enfant est systématiquement créé et c'est ce dernier qui traite la ligne de commande (avec duplications ultérieures si nécessaire).

#### **2. Essayer les commandes suivantes et comparer avec la commande précédente :**

Ici, l'utilisateur se sert des parenthèses pour rediriger la sortie standard de deux commandes :

(date ; ls) > resultat

cat resultat

### **Exercice 5 : Conditions**

- **if-elif-fi**

```
if [ test ]
then
    echo "Le premier test a été vérifié"
elif [ autre_test ]
then
    echo "Le second test a été vérifié"
elif [ encore_autre_test ]
```

then

```
    echo "Le troisième test a été vérifié"
```

else

```
    echo "Aucun des tests précédents n'a été vérifié"
```

fi

**OU**

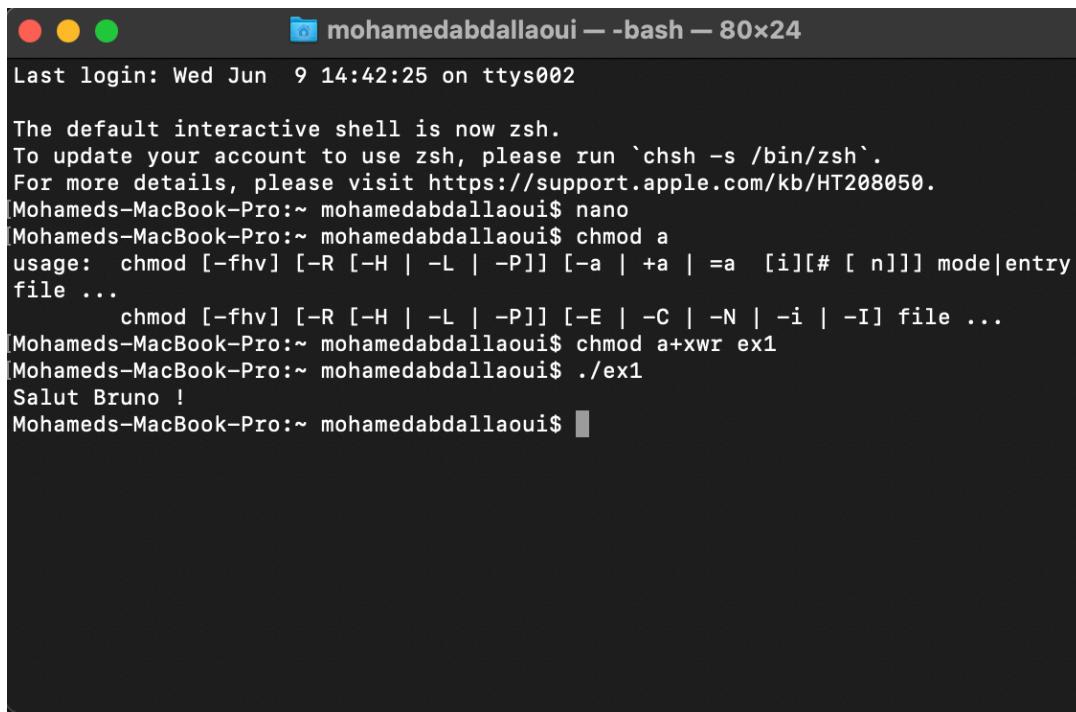
if [ test ]; then

```
    echo "C'est vrai"
```

fi

### **1. Ecrire et exécuter le script suivant :**

```
#!/bin/bash
nom="Bruno"
if [ $nom = "Bruno" ]
then
    echo "Salut Bruno !"
fi
```



A screenshot of a macOS terminal window titled "mohamedabdallaoui — bash — 80x24". The window shows a command-line session. The user has run a script that checks if the variable \$nom is set to "Bruno". Since it is, the script outputs "Salut Bruno !". The terminal also displays system messages about the default shell being zsh and instructions to switch to zsh.

```
Last login: Wed Jun  9 14:42:25 on ttys002

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.

Mohameds-MacBook-Pro:~ mohamedabdallaoui$ nano
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a
usage: chmod [-fhv] [-R [-H | -L | -P]] [-a | +a | =a  [i][# [n]]] mode|entry
      file ...
      chmod [-fhv] [-R [-H | -L | -P]] [-E | -C | -N | -i | -I] file ...
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a+xwr ex1
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./ex1
Salut Bruno !
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

### **2. Ecrire et exécuter le script suivant :**

```
#!/bin/bash
nom1="Bruno"
nom2="Marcel"
```

```
if [ $nom1 = $nom2 ]
then
    echo "Salut les jumeaux !"
fi
```

```
GNU nano 2.0.6          File: j1
#!/bin/bash
nom1="Brun"
nom2="Marcel"
if [ $nom1 = $nom2 ]
then
    echo "Salut les jumeaux !"
fi

[ Read 11 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit  ^J Justify  ^W Where Is  ^V Next Page  ^U UnCut Text  ^T To Spell
```

### 3. Ecrire et exécuter PROPREMENT le script suivant :

```
#!/bin/bash
if [ $1 = "Bruno" ]
then
    echo "Salut Bruno !"
else
    echo "J'te
connais pas, ouste !"
fi
```

```
mohamedabdallaoui -- bash - 80x24
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./q3 moha
J'te connais pas, ouste !
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

Q3 ) resultat ci-dessus

### 4. Ecrire et exécuter PROPREMENT le script suivant :

```
#!/bin/bash
if [ $1 = "Bruno" ]
```

then

```
echo "Salut Bruno !"
elif [ $1 = "Michel" ]
then
    echo "Bien le bonjour Michel"
elif [ $1 = "Jean" ]
then
    echo "Hé Jean, ça va ?"
else
    echo "J'te connais pas, ouste !"
fi
```

Q4) résultat ci-dessous

```
mohamedabdallaoui$ ./q3 Bruno
Salut Bruno !
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

5. Ecrire et exécuter un script qui vérifie si deux paramètres entrés sont identiques ou différents (en affichant un message « les paramètres sont identiques/différents):

```
GNU nano 2.0.6          New Buffer          Modified
#!/bin/bash
$p1="parametre1"
$p2="parametre2"
if [ $p1=$p2 ]
echo "les parametre sont identique"
else
echo "les parametre sont different"
```

**Q5) resultat ci-dessus**

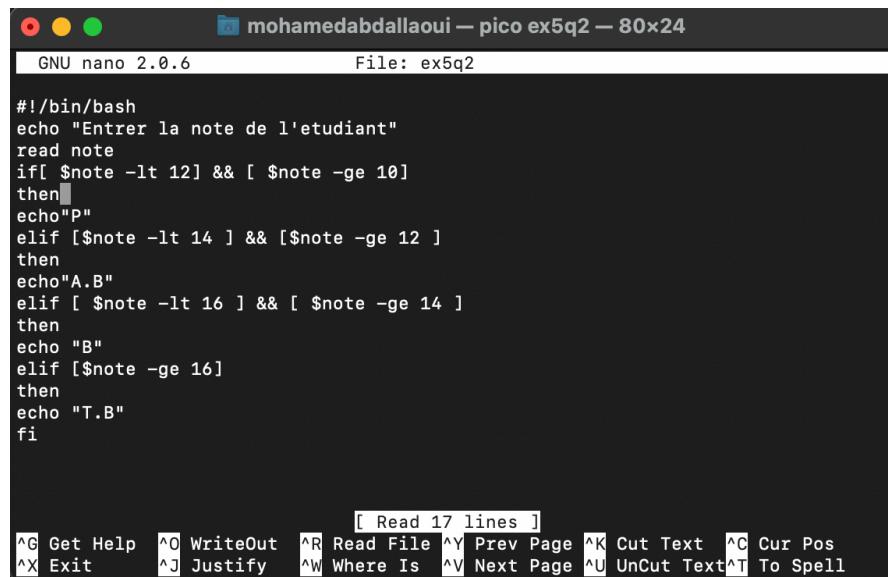
- case-esac

## **1. Ecrire et exécuter le script suivant :**

```
#!/bin/sh
echo "Le contenu du répertoire courant va être affiché."
read -p "Souhaitez-vous afficher aussi les fichiers cachés (oui/non) : " reponse
case $reponse in
    oui)
        clear
        ls -a ;;
    non)
        ls;;
esac
*) echo "Veuillez répondre par oui ou par non." ;;
exit
```

2. Ecrire et exécuter un script pour les mentions d'un étudiant ou la note est entrée par l'utilisateur et la mention est affichée (P, A.B, B, TB)

### **Q2) Resultat ci-dessous**



```
GNU nano 2.0.6          File: ex5q2

#!/bin/bash
echo "Entrer la note de l'étudiant"
read note
if[ $note -lt 12 ] && [ $note -ge 10]
then
echo"P"
elif [ $note -lt 14 ] && [ $note -ge 12 ]
then
echo"A.B"
elif [ $note -lt 16 ] && [ $note -ge 14 ]
then
echo "B"
elif [ $note -ge 16]
then
echo "T.B"
fi

[ Read 17 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

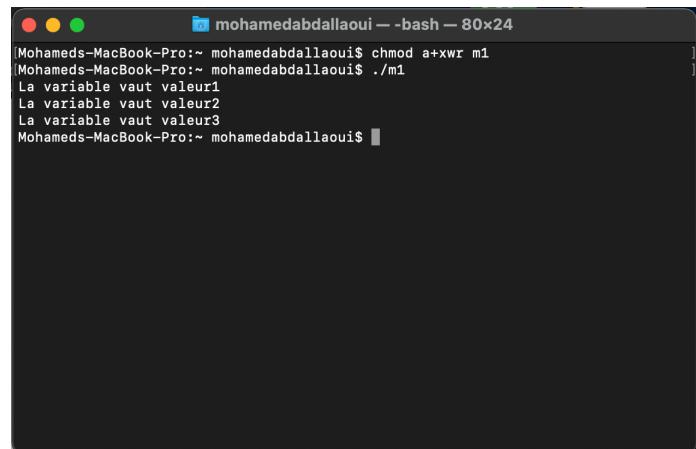
## Exercice 6 : Boucles

- For:

**for** variable in mot1 mot2...  
**do**  
 instructions  
**done**

### 1. *Essayer le script suivant :*

```
#!/bin/bash
for variable in 'valeur1' 'valeur2' 'valeur3'
do
    echo "La variable vaut $variable"
done
```



```
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a+xwr m1
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./m1
La variable vaut valeur1
La variable vaut valeur2
La variable vaut valeur3
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```



```
GNU nano 2.0.6          File: m1

#!/bin/bash
for variable in 'valeur1' 'valeur2' 'valeur3'
do
    echo "La variable vaut $variable"
done

[ Read 6 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

Q1 ) result ci-dessus

**2. Essayer le script suivant :**

```
#!/bin/bash
for animal in 'chien' 'souris' 'moineau'
do
    echo "Animal en cours d'analyse : $animal"
done
```

```
GNU nano 2.0.6          File: m2
#!/bin/bash
for animal in 'chien' 'souris' 'moineau'
do
    echo "Animal en cours d'analyse : $animal"
done
```

```
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico m2
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a+xwr m2
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./m2
Animal en cours d'analyse : chien
Animal en cours d'analyse : souris
Animal en cours d'analyse : moineau
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

**3. Essayer le script suivant :**

```
#!/bin/bash
# Ecriture alternative for ((initial;condition;action))
for ((i=0;i<100;i=i+1)); do
    echo $i
done
exit
```

```
GNU nano 2.0.6          File: m3
#!/bin/bash
# Ecriture alternative for ((initial;condition;action))
for ((i=0;i<100;i=i+1)); do
    echo $i
done
exit
```

```
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./m3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

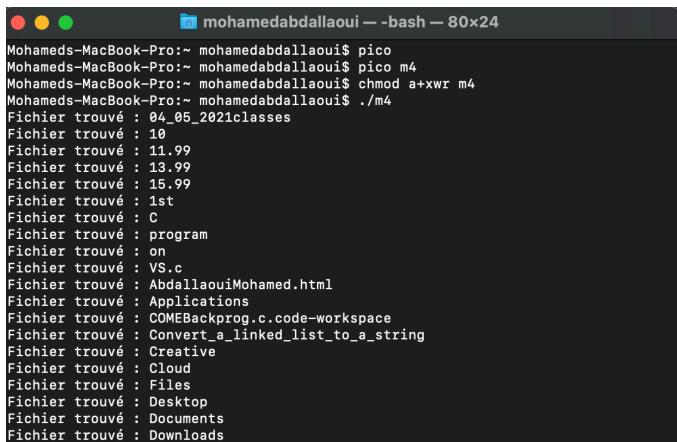
#### 4. Essayer le script suivant :

```
#!/bin/bash
liste_fichiers=`ls`
for fichier in $liste_fichiers
do
    echo "Fichier trouvé : $fichier"
done
```



```
GNU nano 2.0.6          File: m4
#!/bin/bash
liste_fichiers='ls'
for fichier in $liste_fichiers
do
    echo "Fichier trouvé : $fichier"
done

[ Read 7 lines ]
^C Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit     ^J Justify   ^W Where Is  ^V Next Page ^U Uncut Text^T To Spell
```



```
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico m4
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico m4
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a+xwr m4
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./m4
Fichier trouvé : 04_05_2021classes
Fichier trouvé : 10
Fichier trouvé : 11.99
Fichier trouvé : 13.99
Fichier trouvé : 15.99
Fichier trouvé : 1st
Fichier trouvé : C
Fichier trouvé : program
Fichier trouvé : on
Fichier trouvé : VS.c
Fichier trouvé : AbdallaouiMohamed.html
Fichier trouvé : Applications
Fichier trouvé : COMEBackprog.c.code-workspace
Fichier trouvé : Convert_a_linked_list_to_a_string
Fichier trouvé : Creative
Fichier trouvé : Cloud
Fichier trouvé : Files
Fichier trouvé : Desktop
Fichier trouvé : Documents
Fichier trouvé : Downloads
```

- While:

```
while [ test ]
do
    echo 'Action en boucle'
done
Ou

while [ test ]; do
echo 'Action en boucle'
done
```

### 1. Essayer le script suivant :

```
#!/bin/bash
while [ -z $reponse ] || [ $reponse != 'oui' ]
do
    read -p 'Dites oui : ' reponse
done
```

```
mohamedabdallaoui -- pico m5 -- 80x24
GNU nano 2.0.6          File: m5
#!/bin/bash
while [ -z $reponse ] || [ $reponse != 'oui' ]
do
    read -p 'Dites oui : ' reponse
done
```

```
mohamedabdallaoui -- bash -- 80x24
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico m5
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a+xwr m5
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./m5
Dites oui : non
Dites oui : oui
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

### 2. Essayer le script suivant :

```
#!/bin/bash
i=0
while [ $i -lt 100 ] ; do
    echo $i
    i=$((i+1))
done
exit
```

```
mohamedabdallaoui -- pico m6 -- 80x24
GNU nano 2.0.6          File: m6
#!/bin/bash
i=0
while [ $i -lt 100 ] ; do
    echo $i
    i=$((i+1))
done
exit
```

```
mohamedabdallaoui -- bash -- 80x24
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ pico m6
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ chmod a+xwr m6
Mohameds-MacBook-Pro:~ mohamedabdallaoui$ ./m6
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Q2 ) résultat ci-dessus

## Exercice 7 : Fonctions

**Structure :**

function nom {

instructions

}

**ou**

nom () {

instructions

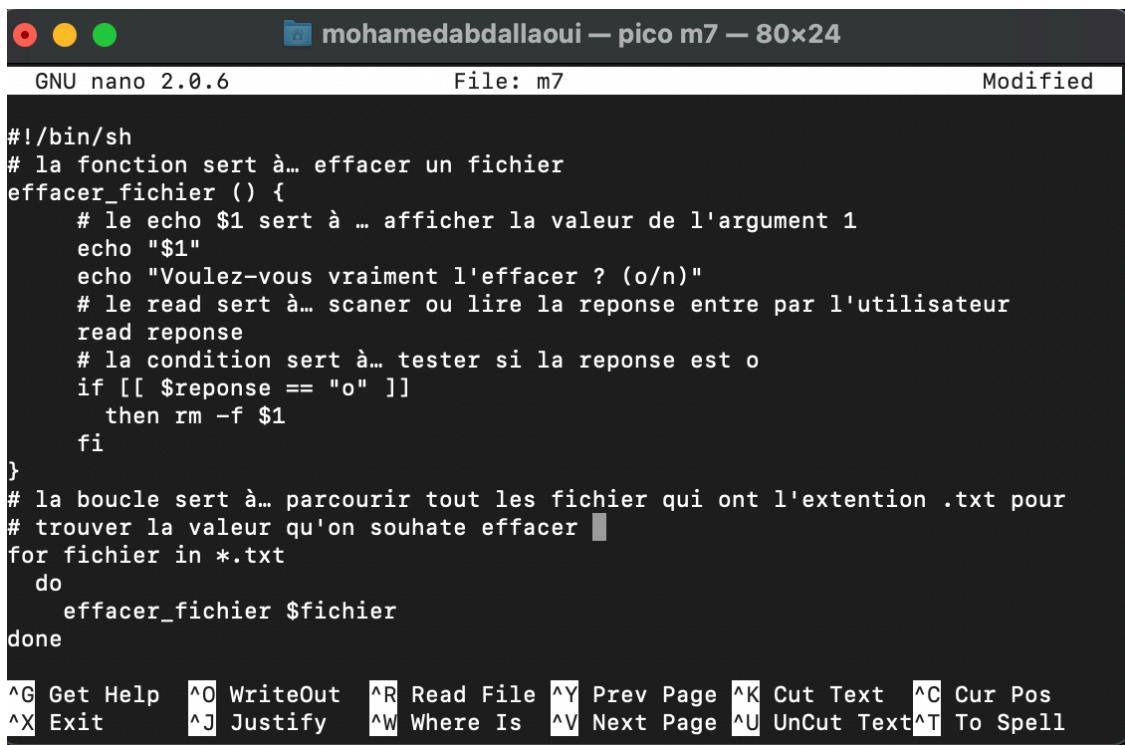
}

**1. Ecrire et exécuter le script suivant :**

```
#!/bin/sh
# la fonction sert à...
effacer_fichier () {
    # le echo $1 sert à ...
    echo "$1"
    echo "Voulez-vous vraiment l'effacer ? (o/n)"
    # le read sert à...
    read reponse
    # la condition sert à...
    if [[ $reponse == "o" ]]
        then rm -f $1
    fi
}
# la boucle sert à...
for fichier in *.txt
do
    effacer_fichier $fichier
done
```

```
mohamedabdallaoui$ ./m7
monreptest.txt
Voulez-vous vraiment l'effacer ? (o/n)
o
test.txt
Voulez-vous vraiment l'effacer ? (o/n)
n
Mohameds-MacBook-Pro:~ mohamedabdallaoui$
```

**2. Essayer d'expliquer le code précédent partie par partie (sous forme de commentaire après les #).**



```
#!/bin/sh
# la fonction sert à... effacer un fichier
effacer_fichier () {
    # le echo $1 sert à ... afficher la valeur de l'argument 1
    echo "$1"
    echo "Voulez-vous vraiment l'effacer ? (o/n)"
    # le read sert à... scanner ou lire la réponse entre par l'utilisateur
    read reponse
    # la condition sert à... tester si la réponse est o
    if [[ $reponse == "o" ]]
        then rm -f $1
    fi
}
# la boucle sert à... parcourir tout les fichier qui ont l'extention .txt pour
# trouver la valeur qu'on souhaite effacer
for fichier in *.txt
do
    effacer_fichier $fichier
done

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U Uncut Text^T To Spell
```