



TP SCRIPTING

EXERCICE 3 :

1- a- Pour la première instruction, il ne devrait pas y avoir d'espace dans l'instruction $i = 1$, la bonne instruction est : **`i=1`**

Pour la deuxième instruction, la variable ne doit pas commencer par un chiffre. La bonne syntaxe pourrait être : **`s=tv`**

Pour la troisième instruction, il devrait pas y avoir d'espace dans la valeur qu'on affecte à la variable unix, la bonne syntaxe pourrait être : **`unix=ritchiethompson`**

b-

Commandes	Durée d'exécution (real)
time mkdir exercice1	0m0, 006s
time rm exercice1	0m0, 005s
time cd /home	0m0, 000s
time echo exercice1	0m0, 000s

c-

* Affichons le contenu de la variable PATH :

```
madeniyou@madeniyou-Inspiron-16-5620: ~  
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* Effaçons le contenu de la variable PATH :

```
madeniyou@madeniyou-Inspiron-16-5620: ~  
madeniyou@madeniyou-Inspiron-16-5620:~$ unset PATH  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* Essayons quelques commandes :

```
madeniyou@madeniyou-Inspiron-16-5620: ~  
madeniyou@madeniyou-Inspiron-16-5620:~$ ls -mkdir  
bash: ls: Aucun fichier ou dossier de ce type  
madeniyou@madeniyou-Inspiron-16-5620:~$ cp essai essai  
bash: cp: Aucun fichier ou dossier de ce type  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

On remarque que les commandes ne s'exécutent pas. Cela est dû au fait que le noyau ne sait pas où chercher la commande, car la variable PATH n'a aucune valeur. Pour y remédier, il faut réaffecter à PATH sa valeur initiale.

```
madeniyou@madeniyou-Inspiron-16-5620: ~  
madeniyou@madeniyou-Inspiron-16-5620:~$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin  
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

d-

* Téléchargeons calcul

```
madeniyou@madeniyou-Inspiron-16-5620:~$ wget https://mandicouba.net/prog/calcul  
--2023-05-22 12:58:29-- https://mandicouba.net/prog/calcul  
Résolution de mandicouba.net (mandicouba.net)... 95.217.105.174  
Connexion à mandicouba.net (mandicouba.net) | 95.217.105.174 | :443... connecté.  
requête HTTP transmise, en attente de la réponse... 200 OK  
Taille : 16712 (16K)  
Enregistre : 'calcul'  
  
calcul 100%[=====>] 16,32K --.-KB/s ds 0s  
2023-05-22 12:58:30 (44,6 MB/s) - 'calcul' enregistré [16712/16712]  
  
madeniyou@madeniyou-Inspiron-16-5620:~$ chmod +x calcul  
madeniyou@madeniyou-Inspiron-16-5620:~$ ls -ld calcul  
-rwxrwxr-x 1 madeniyou madeniyou 16712 me 3 2020 calcul  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* Ajoutons le chemin du fichier dans la variable PATH :

```
madeniyou@madeniyou-Inspiron-16-5620:~$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin:/home/calcul  
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin:/home/calcul  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* Faisons des calculs

```
madeniyou@madeniyou-Inspiron-16-5620:~$ ./calcul 80 + 101  
Resultat: 80 + 101 = 181  
madeniyou@madeniyou-Inspiron-16-5620:~$ ./calcul 800 - 30  
Resultat: 800 - 30 = 770  
madeniyou@madeniyou-Inspiron-16-5620:~$
```

EXERCICE4 : LES VARIABLES

1. Pour répondre aux questions 1a) et 1b) nous avons créés un fichier d'extension .sh. Dans ce fichier nous allons inscrire nos commandes qui seront par la suite exécutés dans le terminal.

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ nano smdm.sh
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ chmod u=x smdm.sh
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

```
prenom='madeniyou'
echo ${prenom}
nom='sall'
echo ${prenom}_${nom}
cat /etc/passwd | grep ${prenom}
```

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ sudo ./smdm.sh
[sudo] Mot de passe de smdm :
madeniyou
madeniyou_sall
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

Lorsqu'on vérifie si l'utilisateur existe, on voit que tel n'est pas le cas parce que cet utilisateur n'a pas été ajouté. Toutefois on pourrait l'ajouter en utilisant la commande adduser.

c) Supprimons ces variables

Pour supprimer ces variables, nous avons tapé la commande unset suivi du nom de la variable qu'on veut supprimer.

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ unset prenom
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ unset nom
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ echo -n /smdm.sh
```

Après suppression, on obtient des espaces vides, cela montre que la suppression a été faite.

2) Nous allons faire un script qui affiche la date :

Pour cela, nous allons créer un autre fichier d'extension sh dans laquelle nous allons éditer ces commandes puis nous allons l'exécuter.

```
echo Nous sommes le `date -u`
```

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ sudo ./date.sh
Nous sommes le mar. 23 mai 2023 13:22:29 UTC
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

3. Modifions le script

Pour cela nous allons entrer à nouveau dans le fichier date.sh pour y apporter ces modifications

```
echo Nous sommes le `date +%d/%m/%y`
```

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ nano date.sh
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ chmod u=rwx date.sh
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ sudo ./date.sh
Nous sommes le 23/05/23
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

4. Faire un script qui affiche la nature de la commande

```
GNU nano 6.2
#!/bin/bash

read -p "veuillez saisir une commande:" commande

type=$(type -t "$commande")

if [[ "$type" == "builtin" ]]; then
    echo "la commande '$commande' est une commande interne."
elif [[ "$type" == "file" ]]; then
    echo "la commande '$commande' est une commande externe."
else
    echo "la commande '$commande' n'est pas reconnue."
fi
```

Maintenant nous allons essayer d'exécuter ce programme

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ ./commandTyp.sh
veuillez saisir une commande:echo
la commande 'echo' est une commande interne.
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ ./commandTyp.sh
veuillez saisir une commande:ls
la commande 'ls' est une commande externe.
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

EXERCICE5 : VARIABLES ET AFFECTATIONS

1.a) Ici on veut montrer que lorsqu'une variable n'a jamais reçu de valeur, alors dans ce cas sa valeur est considérée comme nulle.

Nous allons saisir ces commandes pour confirmer nos propos.

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC: ~  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ echo $inexistante  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ echo $rien  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

La capture ci-dessous montre que maintenant au lieu d'afficher une valeur nulle, le Shell vous affiche un message montrant à l'utilisateur que la variable présumée n'est pas définie

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC: ~  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ set -o nounset  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ echo $rien  
bash: rien : variable sans liaison  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

b)

Nous avons le script donné que nous nous sommes édité dans un fichier d'extension sh. Maintenant nous allons l'exécuter pour voir ce qui est obtenu.

Après exécution, on obtient le résultat ci-dessous.

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC: ~  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ ./esp.sh  
dgi  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

Le Shell essaie d'évaluer la variable \$PREFIXE_ suivie du contenu de la variable \$FICHIER. Cependant, la variable \$PREFIXE_ n'est pas définie, ce qui signifie que le Shell la considère comme une chaîne vide.

Ainsi, la variable NOUVEAU_FICHIER est définie comme une concaténation de la chaîne vide, suivie du contenu de la variable \$FICHIER. Par conséquent, la sortie du script sera la valeur de la variable \$FICHIER, qui est <<dgi>> dans ce cas.

Proposons deux solutions distinctes pour avoir <<esp_dgi>>

- Pour obtenir la sortie esp_dgi, vous devez utiliser des accolades pour délimiter correctement la variable \$PREFIXE et la variable \$FICHIER. Voici la modification du script :

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC: ~  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ PREFIXE=esp  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ FICHIER=dgi  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ NOUVEAU_FICHIER=${PREFIXE}_${FICHIER}  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ NOUVEAU_FICHIER=${PREFIXE}_${FICHIER}  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ echo $NOUVEAU_FICHIER  
esp_dgi  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

En utilisant les accolades \${PREFIXE}, le Shell évaluera correctement la variable PREFIXE avant de la concaténer avec le contenu de la variable \$FICHIER. Ainsi, la variable NOUVEAU_FICHIER sera évaluée comme esp_dgi et cette valeur sera affichée dans la sortie du script.

- Nous avons édité ce script qui donne le même résultat <<esp_dgi>>

```
GNU nano 6.2 dgi.sh *  
#!/bin/bash  
PREFIXE="esp"  
FICHIER="dgi"  
NOUVEAU_FICHIER=$(printf "%s_%s" $PREFIXE $FICHIER)  
echo $NOUVEAU_FICHIER
```

Après exécution, on obtient :

```
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ nano dgi.sh  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$ ./dgi.sh  
esp_dgi  
smdm@smdm-HP-ProBook-440-G8-Notebook-PC:~$
```

EXERCICE 6 :

1- On a la variable suivante :

```
me 22 21:36
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ chaine=ABCDEFGHJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $chaine
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

a- Affichons uniquement FG :

```
me 22 21:41
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaine:5:2}
FG
madeniyou@madeniyou-Inspiron-16-5620:~$
```

b- Affichons uniquement UVWXYZ

```
me 22 21:43
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaine:20:6}
UVWXYZ
madeniyou@madeniyou-Inspiron-16-5620:~$
```

2- Expliquons la sortie des instructions suivantes :

* echo \${chaine#ABC}

Cette syntaxe affiche la valeur de la variable chaine exceptée la chaîne de caractère ABC

```
me 22 21:45
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaine#ABC}
DEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* echo \${chaine#*M}

Cette syntaxe affiche la valeur de la variable chaine exceptée toute la chaîne de caractères venant avant la lettre M, la lettre M comprise

```
me 22 21:46
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaine#*M}
NOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* echo \${chaine#[QRST]}

Cette syntaxe affiche la valeur de la variable chaîne sauf toute chaîne de caractères venant avant la lettre Q ou R ou S ou T, la lettre comprise ; il convient aussi de remarquer que si l'une des 4 lettres apparaît, le reste est ignoré.

```
me 22 21:49
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaîne#[QRST]}
RSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* echo \${chaîne#*}

Cette syntaxe affiche la valeur de la variable chaîne en ne supprimant rien du tout car on n'a rien précisé après le caractère « * »

```
me 22 21:55
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaîne#*}
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

3- Même scénario que la question précédente

a- Avec l'expression **\$variable##motif**

* echo \${chaîne##ABC}

Cette syntaxe affiche la valeur de la variable chaîne exceptée la chaîne de caractère ABC

```
me 22 21:45
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaîne##ABC}
DEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* echo \${chaîne##*M}

Cette syntaxe affiche la valeur de la variable chaîne, en enlevant toute chaîne de caractère venant avant toute lettre M, à chaque fois qu'il y a une lettre M, tout ce qui vient avant ne sera pas affiché y compris la lettre M

```
me 22 22:01
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaîne##*M}
NQRST
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaîne#*M}
NOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

* echo \${chaîne##*[QRST]}

Cette syntaxe affiche la valeur de la variable chaîne en enlevant toute chaîne de caractère venant avant Q, avant R, avant S et avant T, les lettres citées comprises.

```
me 22 22:03
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine##*[QRST]}
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine##*}

Cette syntaxe affiche la valeur de la variable chaine sauf toute chaîne de caractère

```
me 22 22:06
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine##*}
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

b- Avec \$variable%motif

* echo \${chaine%ABC}

Cette syntaxe n'affiche pas une chaîne de caractère se terminant par ABC dans la valeur de chaine

```
me 22 22:11
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%ABC}
ABCDEFGHIJKLMNQRSTUvwxyzABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine%*M}

Cette syntaxe affiche la valeur de chaine telle qu'elle parce que la lettre M n'est pas présente en fin de chaîne

```
me 22 22:27
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%*M}
ABCDEFGHIJKLMNQRSTUvwxyzABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine%*[QRST]}

Cette syntaxe affiche la valeur de chaine en enlevant juste la fin de chaine R, Q, S, ou T

```
me 22 22:29
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%*[QRST]}
ABCDEFGH IJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRS
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine%*}

Cette syntaxe affiche la valeur de chaîne en enlevant toute chaîne de fin

```
me 22 22:31
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%*}
ABCDEFGH IJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

c- Avec \$variable%%motif

* echo \${chaine%%ABC}

Cette syntaxe enlève toute chaîne de caractère se terminant par ABC

```
me 22 22:44
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%%ABC}
ABCDEFGH IJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine%%*M}

Cette syntaxe enlève toute chaîne de caractère se terminant par M

```
me 22 22:46
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%%*M}
ABCDEFGH IJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine%%*[QRST]}

Cette syntaxe enlève toutes les chaînes de caractères se terminant par Q, R, S ou T

```
me 22 22:49
madeniyou@madeniyou-Inspiron-16-5620: ~$ echo ${chaine%%*[QRST]}
madeniyou@madeniyou-Inspiron-16-5620: ~$
```

* echo \${chaine%%*}

Cette syntaxe enlève toute chaîne de caractère se terminant par n'importe quoi

```
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaine%%*}
madeniyou@madeniyou-Inspiron-16-5620:~$
```

4- Conclusion sur le rôle des expressions précédentes :

Ces expressions permettent donc de manipuler et de supprimer des chaînes de valeur d'une variable en fonction des besoins spécifiques.

5- a- Remplaçons R par T dans la valeur de chaîne

```
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${chaine/R/T}
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNQRST
madeniyou@madeniyou-Inspiron-16-5620:~$
```

b- Soit la variable :

```
me 22 23:17
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ variable=dstiladstilbdstilc
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $variable
dstiladstilbdstilc
madeniyou@madeniyou-Inspiron-16-5620:~$
```

Remplaçons toutes les occurrences de i par r

```
me 22 23:17
madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $variable
dstiladstilbdstilc
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${variable/i/r}
dstrladstilbdstilc
madeniyou@madeniyou-Inspiron-16-5620:~$ echo ${variable//i/r}
dstrladstrlbdstrlc
madeniyou@madeniyou-Inspiron-16-5620:~$
```

c- Conclusion sur le rôle des expressions :

Ces deux expressions ont pour rôle de remplacer un caractère par un autre ;
\$variable/motif/remplacement permet de remplacer la première occurrence,
\$variable//motif/remplacement permet de remplacer toutes les occurrences.

EXERCICE 7 :

1- On a la variable :

```
Activités Terminal me 23 08:11
madeniyou@madeniyou-Inspiron-16-5620: ~$
madeniyou@madeniyou-Inspiron-16-5620:~$ infos='aliou/dsti/12-05-1996'
madeniyou@madeniyou-Inspiron-16-5620:~$ echo $infos
aliou/dsti/12-05-1996
madeniyou@madeniyou-Inspiron-16-5620:~$
```

Faisons un script qui affecte les valeurs respectives de la variable infos aux variables user, formation et dnaissance

```
GNU nano 3.2 script.sh
#!/bin/bash
infos='aliou/dsti/12-05-1996'
echo $infos
IFS='/'
read user formation dnaissance <<< "$infos"
echo "user : $user"
echo "formation : $formation"
echo "dnaissance : $dnaissance"
tty
```

```
madeniyou@madeniyou-Inspiron-16-5620:~$ ./script.sh
aliou/dsti/12-05-1996
user : aliou
formation : dsti
dnaissance : 12-05-1996
/dev/pts/2
madeniyou@madeniyou-Inspiron-16-5620:~$
```

2- Modifions le script pour les instructions suivantes :

Réinitialiser IFS et en même temps créer l'utilisateur \$user

```
#!/bin/bash
infos='aliou/dsti/12-05-1996'
echo $infos
IFS='/'
read user formation dnaissance <<< "$infos"
echo "user : $user"
echo "formation : $formation"
echo "dnaissance : $dnaissance"
sudo adduser --group "$formation" --shell /bin/bash --home /home/"$user" --gecos
tty
```

```

madeniyou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ ./script.sh
aliou/dsti/12-05-1996
user : aliou
formation : dsti
dnaissance : 12-05-1996
Ajout du groupe « dsti » (GID 1003)...
Fait.
/dev/pts/2
madeniyou@madeniyou-Inspiron-16-5620:~$ █

```

3- Modifions encore le script pour qu'il puisse lancer le shell bash et se connecter en tant que \$user, créer dans son répertoire un dossier « dos » et afficher le message « mission accomplie »

```

#!/bin/bash
infos='aliou/dsti/12-05-1996'
echo $infos
IFS='/'
read user formation dnaissance <<< "$infos"
echo "user : $user"
echo "formation : $formation"
echo "dnaissance : $dnaissance"
sudo adduser "$user"
su -"$user" -c "/bin/bash"
mkdir dos
echo "Mission accomplie"
tty

```

4- Exécution du script :

```

aliou@madeniyou-Inspiron-16-5620: ~
madeniyou@madeniyou-Inspiron-16-5620:~$ ./script.sh
aliou/dsti/12-05-1996
user : aliou
formation : dsti
dnaissance : 12-05-1996
adduser : L'utilisateur « aliou » existe déjà.
Mot de passe :
bash: impossible de régler le groupe de processus du terminal (-1): Ioctl() inapp
bash: pas de contrôle de tâche dans ce shell
aliou@madeniyou-Inspiron-16-5620:~$

```