

# Correction : TP3-Structures conditionnelles et itératives

---

## Objectif :

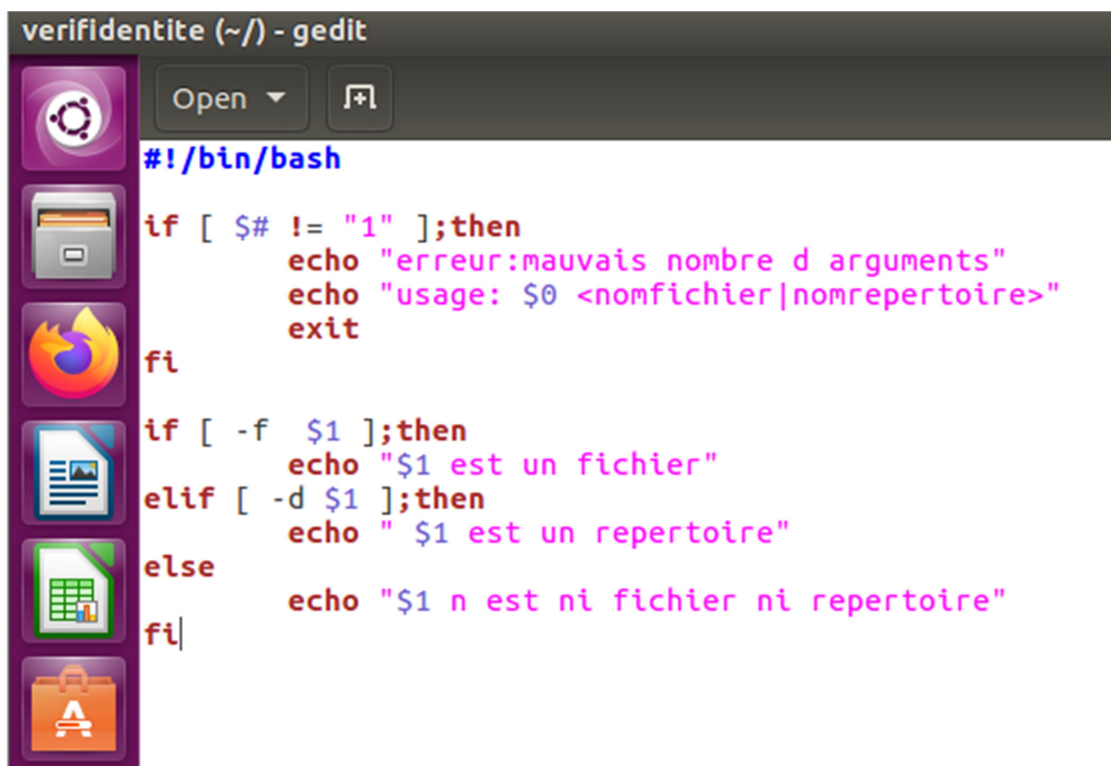
L'objectif visé de ce TP est de :

- Manipuler les structures conditionnelles et itératives en langage Shell.

## Travail demandé :

### Exercice 1

Ecrire un programme verifidentite qui détermine si le paramètre passé en argument est un fichier ou un répertoire.

A screenshot of a terminal window titled 'verifidentite (~/) - gedit'. The window shows a shell script for 'verifidentite'. The script starts with a shebang line '#!/bin/bash'. It then has a first conditional block: 'if [ \$# != "1" ];then' followed by 'echo "erreur:mauvais nombre d arguments"', 'echo "usage: \$0 <nomfichier|nomrepertoire>"', and 'exit'. This is followed by 'fi'. Then a second conditional block: 'if [ -f \$1 ];then' followed by 'echo "\$1 est un fichier"', 'elif [ -d \$1 ];then' followed by 'echo " \$1 est un repertoire"', and 'else' followed by 'echo "\$1 n est ni fichier ni repertoire"'. The script ends with 'fi|'. On the left side of the terminal window, there is a vertical sidebar with icons for various applications: a gear, a folder, a web browser, a document, a spreadsheet, and a presentation.

### Exercice 2

Ecrire un programme copieArchiveHome qui prend en argument le dossier personnel de l'utilisateur et le copie dans un dossier nommé ARCHIVES-NOMUSER.

```
copieArchiveHome (~/) - gedit
#!/bin/bash

if [ $# -ne "1" ];then
    echo "erreur de nombre d arguments"
    exit
fi

if [ -d $1 ];then
    mkdir "/tmp/ARCHIVE-$USER"
    cp -R $1 "/tmp/ARCHIVE-$USER"
else
    echo "$1 n est pas un repertoire"
fi
```

### Exercice 3

Ecrire un programme effacerFichier qui détruit tous les fichiers passés en paramètre, mais, avant de l'effacer, le programme montre les 5 premières lignes et demande une confirmation pour effacement.

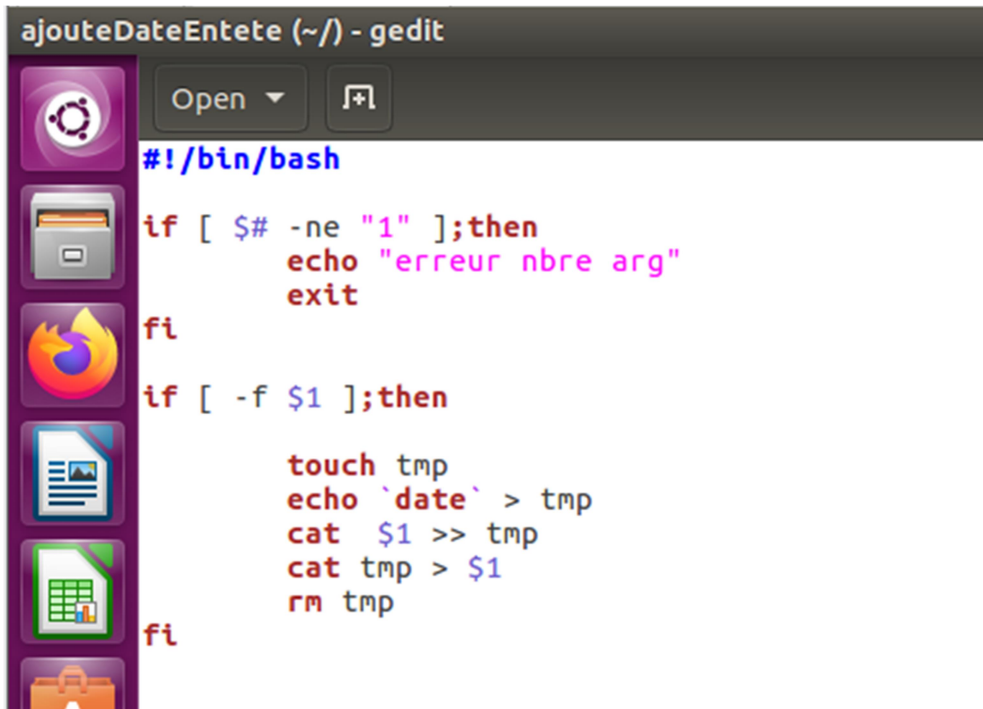
```
effacerFichier (~/) - gedit
#!/bin/bash

if [ $# -eq "0" ];then
    echo "erreur d arguments"
    echo "usage: $0 {fichier}"
    exit
fi

for i in $*
do
    if [ -f $i ];then
        head -5 $i
        rm -i $i
    fi
done
```

### Exercice 4

Ecrire un programme ajouteDateEntete qui ajoute à l'entête du fichier la date actuelle.



```
#!/bin/bash

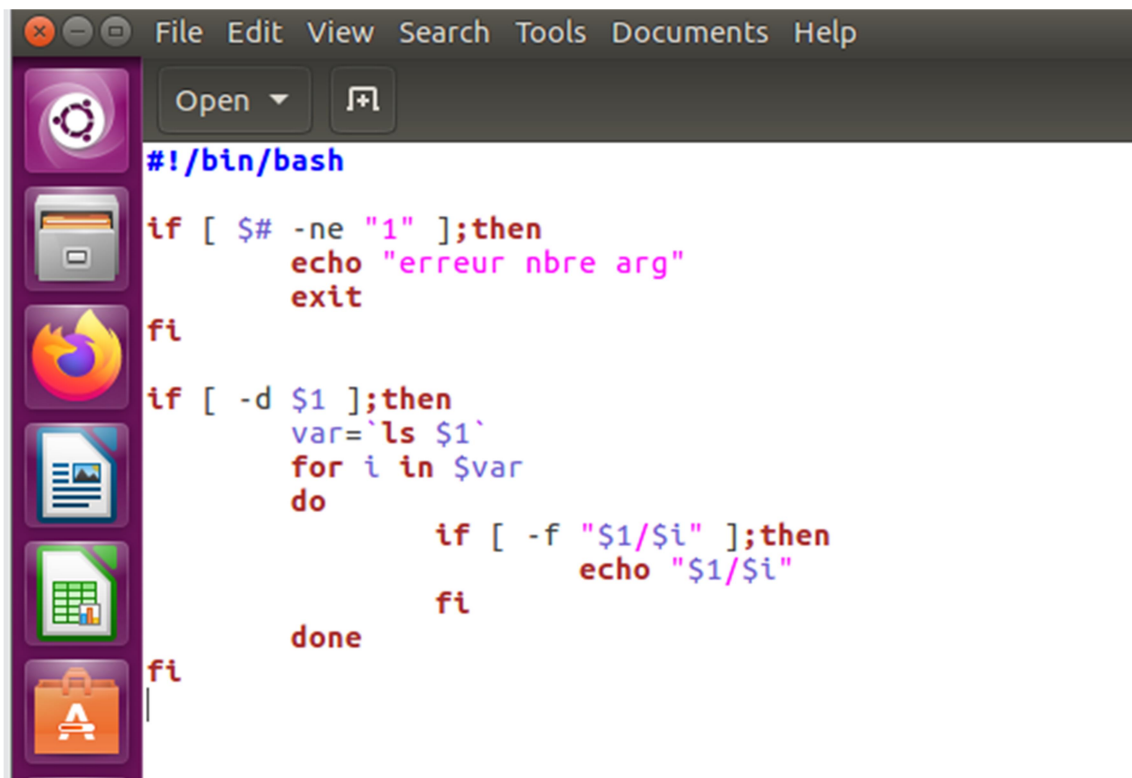
if [ $# -ne "1" ];then
    echo "erreur nbre arg"
    exit
fi

if [ -f $1 ];then

    touch tmp
    echo `date` > tmp
    cat $1 >> tmp
    cat tmp > $1
    rm tmp
fi
```

### Exercice 5

Ecrire un programme fichRep qui prend en argument le nom d'un répertoire et qui liste les fichiers de ce répertoire.



```
#!/bin/bash

if [ $# -ne "1" ];then
    echo "erreur nbre arg"
    exit
fi

if [ -d $1 ];then
    var=`ls $1`
    for i in $var
    do
        if [ -f "$1/$i" ];then
            echo "$1/$i"
        fi
    done
fi
```

### Exercice 6

Ecrire un programme nbreDossFich prenant en argument un répertoire quelconque et qui affiche le nombre de dossiers et fichiers qui existent dans ce dernier.



```
#!/bin/bash

if [ $# -ne "1" ];then
    echo "erreur nbre arg"
    exit
fi

if [ -d $1 ];then
    var=`ls $1`
    nbreDoss=0
    nbreFich=0
    for i in $var
    do
        if [ -f "$1/$i" ];then
            let nbreFich=nbreFich+1
        elif [ -d "$1/$i" ];then
            let nbreDoss=nbreDoss+1
        fi
    done
fi
echo "nbre fichiers= $nbreFich"
echo "nbre dossiers= $nbreDoss"
```

### Exercice 7

Ecrire un programme observations prenant en argument une série de moyennes de matières et attribue pour chaque note une observation :

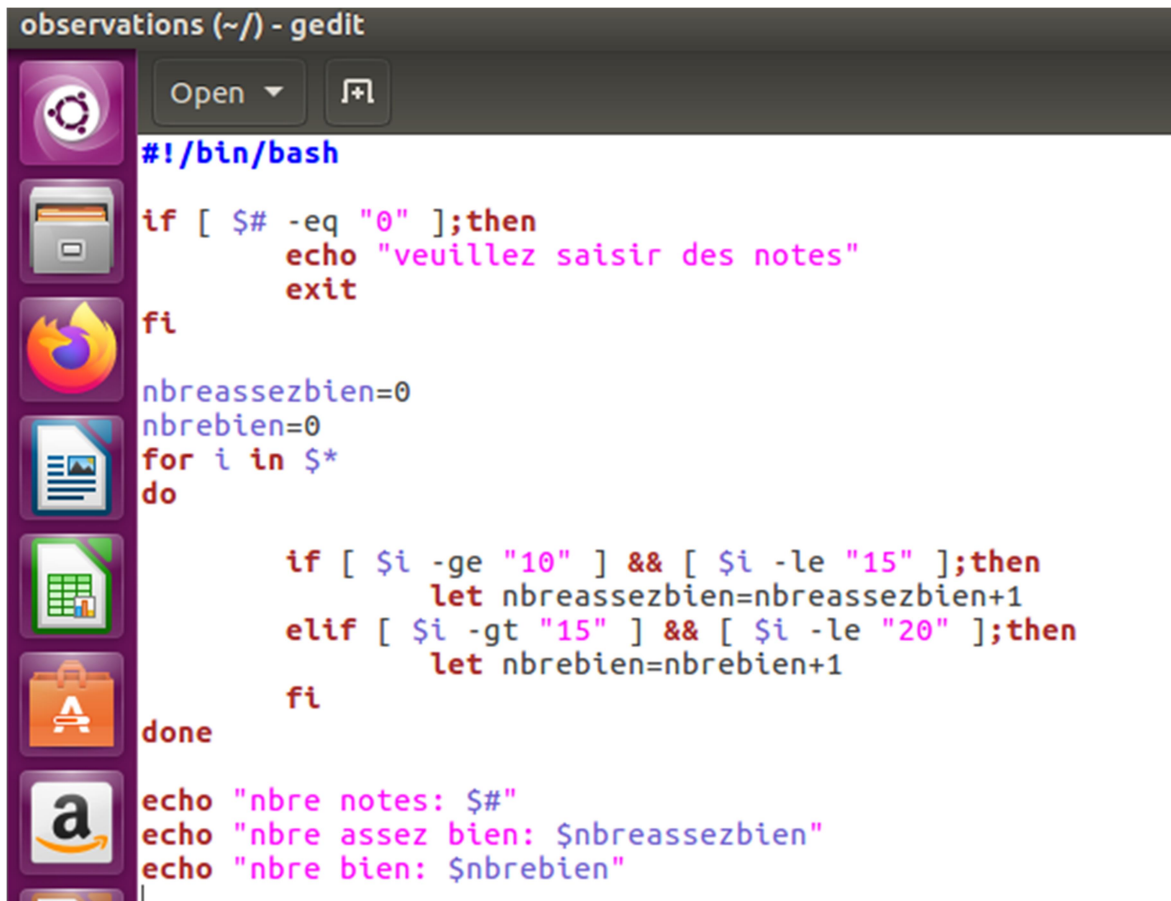
Exemple, le programme doit afficher le texte suivant :

Il y a nbre moyennes qui sont introduites.

Il y a nbre notes qui sont : assez bien.

Il y a nbre notes qui sont : bien.

etc ...



```
#!/bin/bash

if [ $# -eq "0" ];then
    echo "veuillez saisir des notes"
    exit
fi

nbreassezbien=0
nbrebien=0
for i in $*
do
    if [ $i -ge "10" ] && [ $i -le "15" ];then
        let nbreassezbien=nbreassezbien+1
    elif [ $i -gt "15" ] && [ $i -le "20" ];then
        let nbrebien=nbrebien+1
    fi
done

echo "nbre notes: $# "
echo "nbre assez bien: $nbreassezbien"
echo "nbre bien: $nbrebien"
```

....

### Exercice 8

Ecrire le script shell qui permet de faire le traitement ci-dessous: Ce script permet de déterminer le type de quatre arguments saisis par l'utilisateur et faire ainsi des traitements spécifiques. Au départ le script affiche le texte suivant : « Bonjour, test du xxx » (xxx étant la date lors de la saisie) suivi du message : «Total arg est nbr» (nbr étant le nombre d'arguments saisis). Si le nombre d'arguments n'est pas adéquat, le script affiche un message d'erreur. Le script détermine par la suite le type de chaque argument. Dans le cas de:

- Un répertoire : il affiche «arg est un répertoire» puis liste son contenu dans le fichier /tmp/contenu.
- Un fichier ordinaire : il affiche « arg est un fichier» puis le nombre de ligne de ce fichier (la commande pour déterminer le nombre de ligne du fichier est wc -l fichier. Selon le nombre de ligne de fichier un traitement sera fait:

✓ si le nombre de ligne est = 0, il affiche « fichier vide ».

- ✓ si le nombre de ligne est >20, il affiche les 10 premières lignes du fichier (commande head).
- ✓ si le nombre de ligne est <20, il affiche tout le fichier.
- ✓ si le type n'est ni fichier ni répertoire : « arg est de type autre »

NB: arg c'est l'argument tel qu'il a été saisi par l'utilisateur



```
#!/bin/bash
echo "Bonjour, test `date`"
echo "Total arg est $#"
```

```
if [ $# -ne 4 ]
then
echo "Error"
else
for i in $*
do
if [ -d $i ]
then
echo "$i est un répertoire"
ls $i >> /tmp/contenu
elif [ -f $i ]
then
echo "$i est un fichier"
line=`wc -l $i | cut -d" " -f 1`
echo "Le nombre de lignes="$line
if [ $line -eq 0 ]
then
echo "Fichier vide"
fi
if [ $line -ge 20 ]
then
head $i
fi
if [ $line -lt 20 ]
then
cat $i
fi
else
echo "$i est de type autre"
fi
done
fi
```