TP 16 et 17: Scripts shell Linux

6. TP16 – Introduction aux scripts shell.

6.1. L’exécution de scripts

Téléchargement de vim

```
root@DS1: ~#apt-get install vim
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  libgpm2 libsodium23 vim-runtime
Paquets suggérés :
  gpm ctags vim-doc vim-scripts
Les NOUVEAUX paquets suivants seront installés :
```

▪ Création d’un script avec l’éditeur vim :

```
root@DS1: ~#vim un_script.sh_
```

```
#!/bin/bash
date
uptime
uname -a_
~
```

```
root@DS1: ~#ls -l
total 44
-rw-r--r-- 1 root root 146 13 déc.  16:09 avecdoublons
-rw-r--r-- 1 root root   0 13 déc.  16:14 eleves.txt
-rw-r--r-- 1 root root  65 13 déc.  16:14 erreurs.log
-rw-r--r-- 1 root root  73 12 déc.  11:42 etudiants.txt
-rw-r--r-- 1 root root 211 13 déc.  15:53 notes.csv
-rw-r--r-- 1 root root  73 13 déc.  16:11 pasdedoublons
-rw-r--r-- 1 root root 146 13 déc.  15:29 prenom
-rw-r--r-- 1 root root  73 12 déc.  11:46 prenoms_tries
-rw-r--r-- 1 root root  73 13 déc.  16:25 prenoms_tries.txt
-rw-r--r-- 1 root root  73 13 déc.  16:10 sansdoublons
-rw-r--r-- 1 root root 130 13 déc.  16:17 sio1.txt
-rw-r--r-- 1 root root  33 30 janv. 15:18 un_script.sh
root@DS1: ~#_
```

▪ Exécution via le shell standard

```
root@DS1: ~#sh un_script.sh
jeu. 30 janv. 2025 15:19:19 CET
 15:19:19 up 18 min,  1 user,  load average: 0,05, 0,04, 0,03
Linux DS1 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64 GNU/Linux
root@DS1: ~#_
```

▪ Exécution du script en affichant la phase d’interprétation et la trace des commandes :

```
root@DS1: ~#bash -x un_script.sh
+ date
jeu. 30 janv. 2025 15:20:44 CET
+ uptime
 15:20:44 up 19 min,  1 user,  load average: 0,01, 0,03, 0,02
+ uname -a
Linux DS1 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64 GNU/Linux
root@DS1: ~#
```

▪ Exécution du script sous forme d'une commande à partir du répertoire courant :

```
root@DS1: ~#./un_script.sh
-bash: ./un_script.sh: Permission non accordée
root@DS1: ~#chmod +x un_script.sh
root@DS1: ~#ls -l
total 44
-rw-r--r-- 1 root root 146 13 déc.  16:09 avecdoublons
-rw-r--r-- 1 root root   0 13 déc.  16:14 eleves.txt
-rw-r--r-- 1 root root  65 13 déc.  16:14 erreurs.log
-rw-r--r-- 1 root root  73 12 déc.  11:42 etudiants.txt
-rw-r--r-- 1 root root 211 13 déc.  15:53 notes.csv
-rw-r--r-- 1 root root  73 13 déc.  16:11 pasdedoublons
-rw-r--r-- 1 root root 146 13 déc.  15:29 prenom
-rw-r--r-- 1 root root  73 12 déc.  11:46 prenoms_tries
-rw-r--r-- 1 root root  73 13 déc.  16:25 prenoms_tries.txt
-rw-r--r-- 1 root root  73 13 déc.  16:10 sansdoublons
-rw-r--r-- 1 root root 130 13 déc.  16:17 sio1.txt
-rwxr-xr-x 1 root root  33 30 janv. 15:18 un_script.sh
root@DS1: ~#./un_script.sh
jeu. 30 janv. 2025 15:22:03 CET
 15:22:03 up 20 min,  1 user,  load average: 0,00, 0,02, 0,01
Linux DS1 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64 GNU/Linux
root@DS1: ~#un_script.sh
-bash: un_script.sh : commande introuvable
root@DS1: ~#
```

▪ Affichez le contenu de la variable PATH :

```
root@DS1: ~#echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
root@DS1: ~#_
```

▪ Modification di contenu de la variable PATH à partir du fichier /etc/bash.bashrc pour pouvoir exécuter le script à partir d'un répertoire quelconque :

```
# if the command-not-found package is installed, use it
if [ -x /usr/lib/command-not-found -o -x /usr/share/command-not-found/command-not-found ]; then
        function command_not_found_handle {
                # check because c-n-f could've been removed in the meantime
                if [ -x /usr/lib/command-not-found ]; then
                   /usr/lib/command-not-found -- "$1"
                   return $?
                elif [ -x /usr/share/command-not-found/command-not-found ]; then
                   /usr/share/command-not-found/command-not-found -- "$1"
                   return $?
                else
                   printf "%s: command not found\n" "$1" >&2
                   return 127
                fi
        }
fi
export PATH=$PATH:/root
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 30 15:01:34 CET 2025 on tty1
root@DS1: ~#echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root
root@DS1: ~#_
```

▪ Execution du script sans indiquer son chemin :

```
root@DS1: ~#un_script.sh
jeu. 30 janv. 2025 15:29:18 CET
 15:29:18 up 2 min,  1 user,  load average: 0,31, 0,27, 0,11
Linux DS1 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64 GNU/Linux
root@DS1: ~#_
```

6.2. Les variables

▪ Création d'une variable par la suite, pour l'utiliser :

```
root@DS1: ~#CONFIG_SHELL=/etc/profile
root@DS1: ~#ls -l $CONFIG_SHELL
-rw-r--r-- 1 root root 769 10 avril  2021 /etc/profile
root@DS1: ~#_
```

▪ Création d'une variable, affichage de celle-ci, puis affichage et destruction de chacune des variables:

```
root@DS1: ~#A="bonjour Mr"
root@DS1: ~#echo $A
bonjour Mr
root@DS1: ~#set | head
A='bonjour Mr'
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_fignore:globasciiranges:globs
sub_replacement:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_VERSINFO=([0]="2" [1]="11")
BASH_LINENO=()
BASH_LOADABLES_PATH=/usr/local/lib/bash:/usr/lib/bash:/opt/local/lib/bash:/usr/pkg/lib/bash:/opt/pkg/lib/bash:.
root@DS1: ~#unset A
root@DS1: ~#set | grep A=
root@DS1: ~#
```

▪ Création d'un script interactif qui saisit une variable et l'affiche :

```
root@DS1: ~#vim bonjour.sh_
```

```
#!/bin/bash
printf "Votre nom ? "
read nom
echo "Bonjour Mr $nom"_
~
```

```
root@DS1: ~#sh bonjour.sh
Votre nom ? Metreau
Bonjour Mr Metreau
root@DS1: ~#_
```

▪ Création d'un script affichant les paramètres :

```
root@DS1: ~#vim param.sh
```

```
echo "Le nom du script          : $0"
echo "Le 1er parametre          : $1"
echo "Le 2eme parametre         : $2"
echo "Tous les parametres       : $*"
echo "Le nombre de parametres   : $#"
```

```
root@DS1: ~#chmod +x param.sh
root@DS1: ~#./param.sh un deux trois
Le nom du script                : ./param.sh
Le 1er parametre                : un
Le 2eme parametre               : deux
Tous les parametres     : un deux trois
Le nombre de parametres : 3
root@DS1: ~#
```

6.3. Les structures de contrôle

6.3.1. Structure conditionnelle

▪ Ecriture d'un script utilisant une alternative :

```
root@DS1: ~#vim creer_rep.sh
```

```
#!/bin/bash
echo "Nom du répertoire à créer ? "
read nom_rep
if mkdir $nom_rep 2> /dev/null
then
        echo "Opéraion réussie"
else
        echo "Echec"
fi
```

```
root@DS1: ~#sh creer_rep.sh
Nom du répertoire à créer ?
sauve
Opéraion réussie
root@DS1: ~#sh creer_rep.sh
Nom du répertoire à créer ?
sauve
Echec
root@DS1: ~#ls
avecdoublons  creer_rep.sh  erreurs.log    notes.csv  pasdedoublons  prenoms_tries      sansdoublons  sio1.txt
bonjour.sh    eleves.txt    etudiants.txt  param.sh   prenom         prenoms_tries.txt  sauve         un_script.sh
root@DS1: ~#
```

▪ Mise en œuvre d'une alternative avec utilisation de la commande test :

```
root@DS1: ~#vim heureux.sh
```

```
printf "Êtes-vous heureux ? "
read reponse
if [ "$reponse" = "oui" ]
then
        echo "Bravo"
else
        echo "Mangez du chocolat"
fi
```

```
root@DS1: ~#sh heureux.sh
Êtes-vous heureux ? oui
Bravo
root@DS1: ~#sh heureux.sh
Êtes-vous heureux ? n
Mangez du chocolat
root@DS1: ~#
```

6.3.2. Boucle while.

▪ Création d'un script qui teste l'existence du fichier « flag » toutes les dix secondes.

```
root@DS1: ~#vim flag_existe.sh
```

```
while [ ! -f flag ]
do
sleep 10
done
echo "Le fichier flag existe"
```

```
root@DS1: ~#rm flag
root@DS1: ~#sh flag_existe.sh &
[1] 1485
root@DS1: ~#touch flag
root@DS1: ~#Le fichier flag existe

[1]+  Fini                    sh flag_existe.sh
root@DS1: ~#
```

▪ Autre version du programme :

```
root@DS1: ~#vim flag_existebis.sh
```

```
while :
do
        sleep 10
        if test -f flag ;then
                break
        fi
done
echo "Le fichier flag existe"
```

```
"flag_existebis.sh" 8L, 91B écrit(s)
root@DS1: ~#rm flag
root@DS1: ~#sh flag_existebis.sh &
[2] 1581
root@DS1: ~#touch flag
root@DS1: ~#Le fichier flag existe

[2]+  Fini                    sh flag_existebis.sh
root@DS1: ~#
```

6.3.3. Boucle for

▪Un script permettant d'effectuer un décompte de secondes à l'aide d'une boucle "for" :

```
root@DS1: ~#vim boucle_for.sh
```

```
for i in 5 4 3 2 1
do
        echo "=====> $i"
        sleep 1
done
echo "FEU!"
```

```
root@DS1: ~#sh boucle_for.sh
=====> 5
=====> 4
=====> 3
=====> 2
=====> 1
FEU!
root@DS1: ~#
```

6.3.4. Choix multiple

▪ Ecriture d'un script "menu" permettant d'écrire un menu avec l'instruction case :

```
root@DS1: ~#vim menu.sh
```

```
#!/bin/bash

echo "1 - Afficher la date et l'heure"
echo "2 - Afficher la charge système"
echo "3 - Aficher la version du système"

echo -n "votre choix ? "
read choix

case "$choix" in
1)
        date
        ;;
2)
        uptime
        ;;
3)
        uname -a
        ;;
*)
        echo "choix incorrect"
        ;;
esac
```

```
root@DS1: ~#sh menu.sh
1 - Afficher la date et l'heure
2 - Afficher la charge système
3 - Aficher la version du système
votre choix ? 3
Linux DS1 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64 GNU/Linux
root@DS1: ~#
```

6.3.5. La commande let

▪ Ecriture d'un script "+1" utilisant une boucle while et mettant en œuvre l'incrémentation d'une variable via la commande let :

```
root@DS1: ~#vim plusun.sh
```

```
#!/bin/bash
i=0     # ou let i=0
while ((i<5)) # ou while let 'i<5'
do
        echo Bonjour
        let i=i+1       # ou ((i+1))
done
```

```
root@DS1: ~#bash plusun.sh
Bonjour
Bonjour
Bonjour
Bonjour
Bonjour
```

ou

```
root@DS1: ~#chmod +x plusun.sh
root@DS1: ~#plusun.sh
Bonjour
Bonjour
Bonjour
Bonjour
Bonjour
root@DS1: ~#_
```

6.3.6. Lecture d'un fichier et commande set

▪ Création d'un fichier users.txt contenant des lignes au format : login, mot-de-passe nom d'utilisateur groupes secondaires, avec les champs séparés par un espace et sans espaces internes.

```
kbeatini watson kevin 2SIO,SISR
acanonne navarone axel 2SIO,SISR
cmartinez malaucoccyx cedric 2SIO,SISR
vgoss toutafait victor 2SIO,SISR
jpichon pelican jonathan 2SIO,SISR
rfumey weed raphael 2SIO,SISR
vamnceau dakota vincent 2SIO,SISR_
```

▪ Rédaction d'un script lecture.sh qui lit le fichier users.txt ligne par ligne, extrait les champs dans des variables et les affiche.

```
root@DS1: ~#vim lecture.sh_
```

```
#!/bin/bash
cat users.txt | while true
do
        read ligne
        if [ "$ligne" = "" ] ; then break ; fi
        echo "lecture de la ligne --->" $ligne
        set $ligne_
        login=$1
        passwd=$2
        nom=$3
        groupe=$4
        echo login=$login, mdp=$passwd, groupe=$groupe, nom=$nom
done
```

```
root@DS1: ~#sh lecture.sh
lecture de la ligne ---> kbeatini watson kevin 2SIO,SISR
login=kbeatini, mdp=watson, groupe=2SIO,SISR, nom=kevin
lecture de la ligne ---> acanonne navarone axel 2SIO,SISR
login=acanonne, mdp=navarone, groupe=2SIO,SISR, nom=axel
lecture de la ligne ---> cmartinez malaucoccyx cedric 2SIO,SISR
login=cmartinez, mdp=malaucoccyx, groupe=2SIO,SISR, nom=cedric
lecture de la ligne ---> vgoss toutafait victor 2SIO,SISR
login=vgoss, mdp=toutafait, groupe=2SIO,SISR, nom=victor
lecture de la ligne ---> jpichon pelican jonathan 2SIO,SISR
login=jpichon, mdp=pelican, groupe=2SIO,SISR, nom=jonathan
lecture de la ligne ---> rfumey weed raphael 2SIO,SISR
login=rfumey, mdp=weed, groupe=2SIO,SISR, nom=raphael
lecture de la ligne ---> vamnceau dakota vincent 2SIO,SISR
login=vamnceau, mdp=dakota, groupe=2SIO,SISR, nom=vincent
root@DS1: ~#
```

6.3.7. Commande shift et boucle for

Écriture d'un script qui reçoit un répertoire en premier argument, le sauvegarde dans une variable, et traite ensuite les fichiers passés en arguments, en les récupérant dans la variable $* après utilisation de la commande shift pour exclure le répertoire.

```
root@DS1: ~#vim decal.sh
```

```
# Affichage des variables paramètres de position avant le décalage shift
echo -e "Affichage avant shift\n"
echo "1er argument \$1 : $1"
echo "2eme argument \$2 : $2"
echo "3eme argument \$3 : $3"
echo "4eme argument \$4 : $4"
echo "Tous les arguments \$* : $*"
echo -e "Nombre d'arguments \$# : $#\n"
#Sauvegarde du 1er argument et décalage des arguments avec la commande shift
rep=$1
shift
# Affichage des variables après éxécution de la commande shift
echo -e "Affichage après utilisation de la commande shift\n"
echo "1er argument \$1 : $1"
echo "2eme argument \$2 : $2"
echo "3eme argument \$3 : $3"
echo "4eme argument \$4 : $4"
echo "Tous les arguments \$* : $*"
echo -e "Nombre d'aruments \$# : $#\n"
#Création du répertoire et déplacement dans le répertoire créé
mkdir $rep
cd $rep
```

```
# Création des fichiers dans le répertoire créé
for fichier in $*
do
        touch $fichier
        echo "Fichier $fichier créé"
done
```

▪ Accorde le droit d'exécution :

```
root@DS1: ~#chmod u+x decal.sh
root@DS1: ~#
```

▪ Exécution du script :

```
root@DS1: ~#./decal.sh /test f1 f2 f3 f4 f5 f6
Affichage avant shift

1er argument $1 : /test
2eme argument $2 : f1
3eme argument $3 : f2
4eme argument $4 : f3
Tous les arguments $* : /test f1 f2 f3 f4 f5 f6
Nombre d'arguments $# : 7

Affichage après utilisation de la commande shift

1er argument $1 : f1
2eme argument $2 : f2
3eme argument $3 : f3
4eme argument $4 : f4
Tous les arguments $* : f1 f2 f3 f4 f5 f6
Nombre d'arguments $# : 6

mkdir: impossible de créer le répertoire « /test »: Le fichier existe
Fichier f1 créé
Fichier f2 créé
Fichier f3 créé
Fichier f4 créé
Fichier f5 créé
Fichier f6 créé
```

▪ Vérification de la création du répertoire et des fichiers :

```
root@DS1: ~#ls -l /test
total 0
-rw-r--r-- 1 root root 0 31 janv. 15:36 f1
-rw-r--r-- 1 root root 0 31 janv. 15:36 f2
-rw-r--r-- 1 root root 0 31 janv. 15:36 f3
-rw-r--r-- 1 root root 0 31 janv. 15:36 f4
-rw-r--r-- 1 root root 0 31 janv. 15:36 f5
-rw-r--r-- 1 root root 0 31 janv. 15:36 f6
root@DS1: ~#
```

6.4. Les sous-programmes

▪ Ecriture d'un script utilisant une fonction de présentation avec passage d'arguments à la suite du nom de la fonction :

```
root@DS1: ~#vim affiche.sh
```

```
presentation ()
{
        for i
        do
                echo "==== $i"
        done
        echo
}

# Debut du programme

presentation "Bonjour" "Ce matin" "nous etudions" "les structures de contrôle" "ainsi que" "les fonctions"
date
presentation "Soyez attentif à la syntaxe" "Bon courage"
~
```

```
root@DS1: ~#sh affiche.sh
==== Bonjour
==== Ce matin
==== nous etudions
==== les structures de contrôle
==== ainsi que
==== les fonctions

ven. 31 janv. 2025 15:56:45 CET
==== Soyez attentif à la syntaxe
==== Bon courage

root@DS1: ~#
```

▪ Écriture d'un script qui affiche un menu. La sortie du programme est contrôlée par une fonction qui demande une confirmation.

```
#!/bin/bash

confirmation ()
{
        print "Etes-vous sur $* (y/n) ? "
        read reponse
        if [ $reponse = "y" ] ;then return 0 ; else return 1 ; fi
}

while :
do
        clear
        echo "1 - Afficher la date et l'heure"
        echo "2 - Afficher la charge systeme"
        echo "3 - Afficher la version du systeme"
        echo "99 - FIN"
        printf "Votre choix ? "; read choix
        case "$choix" in
        1) date ;;
        2) uptime ;;
        3) uname -a ;;
        99)
                if confirmation "de vouloir quitter" ;then
                        break
                fi
                ;;
        *) echo "Choix incorrect" ;;
        esac
        sleep 3
done
exit 0
```

```
1 - Afficher la date et l'heure
2 - Afficher la charge systeme
3 - Afficher la version du systeme
99 - FIN
Votre choix ? 99
Warning: unknown mime-type for "Etes-vous sur de vouloir quitter (y/n) ? " -- using "application/octet-stream"
Error: no such file "Etes-vous sur de vouloir quitter (y/n) ? "
y
root@DS1: ~#
```

```
1 - Afficher la date et l'heure
2 - Afficher la charge systeme
3 - Afficher la version du systeme
99 - FIN
Votre choix ? 9
Choix incorrect
^C
root@DS1: ~#
```

7. TP17 – Introduction aux scripts shell : autoformation.

7.1. Saisie et exécution du script

Exemple :
Saisie du script bonjourbis.sh :

```
root@DS1: ~#vim bonjourbis.sh
```

```
#!/bin/bash
echo Bonjour $USER
echo -n "Nous sommes le " ; date
echo "Votre numéro d'utilisateur est"_$(grep "^$USER" /etc/passwd | cut -d: -f3)
~
```

chmod a+x bonjourbis.sh

```
root@DS1: ~#chmod a+x bonjourbis.sh
root@DS1: ~#
```

Lancement de l'exécution du script, tapez ./bonjourbis.sh

```
root@DS1: ~#./bonjourbis.sh
Bonjour root
Nous sommes le ven. 31 janv. 2025 16:27:15 CET
Votre numéro d'utilisateur est 0
root@DS1: ~#
```

Exemple :
→ Saisie du script bonjourter.sh

```
root@DS1: ~#vim bonjourter.sh
```

```
#!/bin/bash
if [ $# = 2 ]
then
echo "Bonjour $2 $1 et bonne journée !"
else
echo "Syntaxe : $0 nom prénom"
fi
```

→ Appel du script sans et avec arguments :

```
root@DS1: ~#sh bonjourter.sh
Syntaxe : bonjourter.sh nom prénom
root@DS1: ~#sh bonjourter.sh P Boucly
Bonjour Boucly P et bonne journée !
root@DS1: ~#
```

7.2. Entrées-Sorties

• echo “Bonjour à tous !” :

```
root@DS1: ~#echo "Bonjour à tous !"
Bonjour à tous !
root@DS1: ~#
```

Exemple :

```
root@DS1: ~#echo -e "Bonjour \nà tous !"
Bonjour
à tous !
root@DS1: ~#echo -e "Bonjour \nà toutes \net à tous ! \c"
Bonjour
à toutes
et à tous ! root@DS1: ~#
```

Exemple :

→ Saisie du script saisie_clavier.sh.

```
#!/bin/bash
echo "Donnez votre prénom et votre nom :"
read prenom nom
echo "Bonjour $prenom $nom"
```

→ Appel du script saisie_clavier.sh.

```
root@DS1: ~#sh saisie_clavier.sh
Donnez votre prénom et votre nom :
Nicolas M
Bonjour Nicolas M
root@DS1: ~#
```

7.3. Les variables BASH

Exemple :

```
root@DS1: ~#n=123
root@DS1: ~#echo "La variable \$n vaut $n"
La variable $n vaut 123
root@DS1: ~#salut="bonjou à tous !"
root@DS1: ~#echo "Alors moi je dis : $salut"
Alors moi je dis : bonjou à tous !
```

Exemple (à saisir dans la console) :

```
root@DS1: ~#echo 'Alors moi je dis : $salut'
Alors moi je dis : $salut
root@DS1: ~#echo "Alors moi je dis : \"$salut\""
Alors moi je dis : ""
root@DS1: ~#salut="bonjour à tous !"
root@DS1: ~#echo "Alors moi je dis : \"$salut\""
Alors moi je dis : "bonjour à tous !"
root@DS1: ~#readonly salut
root@DS1: ~#salut="Bonjour à tous sauf à toto"
-bash: salut : variable en lecture seule
root@DS1: ~#echo "Alors moi je dis : $salut"
Alors moi je dis : bonjour à tous !
root@DS1: ~#
```

Exemple :

```
root@DS1: ~#user="/home/stagiaire"
root@DS1: ~#echo $user
/home/stagiaire
```

```
root@DS1: ~#u1=$user1
root@DS1: ~#echo $u1

root@DS1: ~#u1=${user}1
root@DS1: ~#echo $u1
/home/stagiaire1
root@DS1: ~#_
```

7.4. La commande test

Exemples :

```
root@DS1: ~#[ -e ./fichier ]
root@DS1: ~#echo $?
0
root@DS1: ~#touch fichier
root@DS1: ~#[ -e ./fichier ]
root@DS1: ~#echo $?
0
root@DS1: ~#[ -s ./fichier ]
root@DS1: ~#echo $?
0
root@DS1: ~#date > fichier
root@DS1: ~#[ -s ./fichier ]
root@DS1: ~#echo $?
0
root@DS1: ~#
```

```
root@DS1: ~#[ -r "/etc/passwd" ]
root@DS1: ~#echo $?
0
root@DS1: ~#[ -r "/etc/shadow" ]
root@DS1: ~#echo $?
0
```

```
root@DS1: ~#su - nicolas
nicolas@DS1:~$ [ -r "/etc/shadow" ]
nicolas@DS1:~$ echo $?
1
nicolas@DS1:~$ [ -r "/etc/shadow" ] || echo "lecture du fichier interdite"
lecture du fichier interdite
nicolas@DS1:~$ [ -r "/etc/passwd" ]
nicolas@DS1:~$ echo $?
0
nicolas@DS1:~$
```

Test d'une chaine
Exemples :

```
root@DS1: ~#[ -z "est-ce que la chaine est vide ?" ] ; echo $?
1
root@DS1: ~#ch="Bonjour" ; [ "$ch" = "bonjour" ] ; echo $?
1
root@DS1: ~#_
```

```
root@DS1: ~#[ $USER != "root" ] && echo "l'utilisateur n'est pas \"root\" !" || echo "l'utilisateur est le \"root\" !"
l'utilisateur est le "root" !
root@DS1: ~#su - nicolas
nicolas@DS1:~$ [ $USER !="root" ] && echo l'utilisateur n'est pas \"root\" !" || echo "l'utilisateur est le \"root\" !"
> ^C
nicolas@DS1:~$ [ $USER != "root" ] && echo "l'utilisateur n'est pas \"root\" !" || echo "l'utilisateur est le \"root\"
l'utilisateur n'est pas "root" !
nicolas@DS1:~$ _
```

Test d'un nombre
Exemples :

```
root@DS1: ~#a=15 ; [ $a -lt 15 ] ; echo $?
1
root@DS1: ~#a=15 ; [ $a -le 15 ] ; echo $?
0
root@DS1: ~#
```

Opérations dans une commande test
Exemples :

```
root@DS1: ~#ls -ld /root
drwx------ 5 root root 4096  7 févr. 16:59 /root
root@DS1: ~#_
```

```
root@DS1: ~#note=9 ; [ $note -lt 8 -o $note -ge 10 ] && echo "tu n'es pas convoqué à l'oral"
root@DS1: ~#note=7 ; [ $note -lt 8 -o $note -ge 10 ] && echo "tu n'es pas convoqué à l'oral"
tu n'es pas convoqué à l'oral
root@DS1: ~#_
```

7.5. Structures conditionnelles

7.5.1. Conditionnelle simple

Exemples :

```
root@DS1: ~#vim compte.sh_
```

```
#!/bin/bash
if grep "^nicolas" /etc/passwd
then
echo "nicolas a déjà un compte"
fi
```

```
root@DS1: ~#sh compte.sh
nicolas:x:1000:1000::/home/nicolas:/bin/bash
nicolas a déjà un compte
root@DS1: ~#_
```

```
#!/bin/bash
if grep "^nicolas" /etc/passwd > /dev/null_
then
echo "nicolas a déjà un compte"
fi
```

```
root@DS1: ~#sh compte.sh
nicolas a déjà un compte
root@DS1: ~#
```

```
root@DS1: ~#vim note.sh_
```

```
#!/bin/bash
echo "Saisissez votre note :"
read note
if [ $note -gt 16 ]
then echo "C'est tres bien !"
fi
```

```
root@DS1: ~#sh note.sh
Saisissez votre note :
17
C'est tres bien !
root@DS1: ~#
```

7.5.2. Conditionnelles imbriquées

Exemples :

```
root@DS1: ~#vim examen.sh
```

```
#!/bin/bash
echo "Saisissez votre moyenne : "
read note
if [ $note -lt 8 ]
then
echo "Vous êtes recalé"
elif [ $note -lt 10 ]
then
echo "Vous êtes convoqué à l'oral de rattrapage"
else
echo "Vous êtes admis"
fi
```

```
root@DS1: ~#sh examen.sh
Saisissez votre moyenne :
20
examen.sh: 5: [: missing ]
Vous êtes admis
root@DS1: ~#
```

```
root@DS1: ~#vim devoir.sh
```

```
#!/bin/bash
Fichier=/home/nicolas/devoir1.txt
if [ -f $Fichier -a -r $Fichier ]
then
echo "je vais verifier votre devoir"
elif [ ! -e $Fichier ]
then
echo "ton devoir n'existe pas ! "
else
echo "je ne peux pas lire  !"
fi
```

```
root@DS1: ~#sh devoir.sh
ton devoir n'existe pas !
root@DS1: ~#su - nicolas
nicolas@DS1:~$ touch devoir1.txt
nicolas@DS1:~$ exit
```

```
root@DS1: ~#sh devoir.sh
je vais verifier votre devoir
root@DS1: ~#_
```

→ Supposons que le script exige la présence de deux paramètres. Il faut tester la valeur de $#. Est-elle nulle ?

```
root@DS1: ~#vim atgument.sh
```

SI $# est égal à 0 alors la valeur sera nulle, cela signifie que l'utilisateur n'a passé aucun argument au script lors de son exécution.

```
#!/bin/bash
if [ $# = 0 ]
then
echo "Erreur, la commande exige deux arguments"
elif [ $# = 1 ]
then
echo "Donnez le second argument :"
read arg2
fi
```

```
root@DS1: ~#sh argument.sh
Erreur, la commande exige deux arguments
root@DS1: ~#sh argument.sh arg1
Donnez le second argument :

root@DS1: ~#
```

7.5.3. Choix multiples

Exemples :

```
root@DS1: ~#vim cas.sh_
```

```
#!/bin/bash
case $USER in
root)
        echo "Mes respects Monsieur le $USER" ;;
guest | nicolas?)
        echo "Salut $USER" ;;
*)
        echo "Bonjour $USER" ;;
esac
```

```
root@DS1: ~#chmod 755 cas.sh
root@DS1: ~#cas.sh
Mes respects Monsieur le root
root@DS1: ~#_
```

```
root@DS1: ~#cp /root/cas.sh /bin/
root@DS1: ~#su - nicolas
nicolas@DS1:~$ cas.sh
Bonjour nicolas
nicolas@DS1:~$
```

→ Le script attend une réponse oui/non de l'utilisateur

```
root@DS1: ~#vim poursuite.sh_
```

```
#!/bin/bash
echo "Voulez-vous vraiment éxécuter le script ?"
read reponse
case $reponse in
[nN]*)
        echo "Script interrompu"
        exit 0
        ;;
[yYoO]*)
        echo "Attention pour le décompte final"
        for i in 10 9 8 7 6 5 4 3 2 1
        do
        echo "===>$i"
        sleep 1
        done
        echo "Allumage Vulcain"
        sleep 2
        echo "Allumage des 2 EAP"
        sleep 2
        echo "Décollage"
        sleep 2
        echo "Tous les paramètres à bord sont normaux"
        sleep 2
        echo "Victor est en orbite"
        ;;
esac
```

```
root@DS1: ~#sh poursuite.sh
Voulez-vous vraiment éxécuter le script ?
oui
Attention pour le décompte final
===>10
===>9
===>8
===>7
===>6
===>5
===>4
===>3
===>2
===>1
Allumage Vulcain
Allumage des 2 EAP
Décollage
Tous les paramètres à bord sont normaux
Victor est en orbite
root@DS1: ~#
```

7.6. Structures itératives

7.6.1. Boucle for

Exemples :
→ La liste peut être explicite :

```
#!/bin/bash
for nom in Benoit Nicolas Pierre
do
echo "$nom, bonjour"
done
```

```
root@DS1: ~#sh liste.sh
Benoit, bonjour
Nicolas, bonjour
Pierre, bonjour
root@DS1: ~#
```

→ La liste peut être calculée à partir d'une expression modèle

```
#!/bin/bash
if [ ! -e /tmp/nicolas ]
then
mkdir /tmp/nicolas
fi
for fich in /home/nicolas/*
do
        cp $fich /tmp/nicolas
done
```

```
root@DS1: ~#sh copie.sh
root@DS1: ~#ls -l /tmp/nicolas
total 0
-rw-r--r-- 1 root root 0  8 févr. 12:26 devoir1.txt
root@DS1: ~#
```

→ Si aucune liste n'est précisée, les valeurs sont prises dans la variable système $@, c'est-à-dire en parcourant la liste des paramètres positionnels courants.

```
#!/bin/bash
cd /tmp/nicolas ; set *
for nom in $@
do echo $nom
done
```

```
root@DS1: ~#sh sanslisteprecise.sh
devoir1.txt
root@DS1: ~#
```

7.6.2. Boucle while

Exemples :
→ Dire bonjour toutes les secondes (arrêt par CTRL-C) :

```
root@DS1: ~#vim true.sh
```

```
#!/bin/bash
while true
do
echo "Bonjour Mr $USER"
sleep 1
done
```

```
root@DS1: ~#sh true.sh
Bonjour Mr root
Bonjour Mr root
Bonjour Mr root
Bonjour Mr root
Bonjour Mr root
Bonjour Mr root
^C
root@DS1: ~#_
```

• Sortie de boucle
Exemple :
→ Lecture des lignes d'un fichier

```
root@DS1: ~#vim sortie.sh
```

```
#!/bin/bash
fich="/etc/passwd"
grep "^nicolas" $fich | while true
do
        read ligne
        if [ "$ligne" =  "" ] ; then break ; fi
        echo $ligne
done
```

```
root@DS1: ~#sh sortie.sh
nicolas:x:1000:1000::/home/nicolas:/bin/bash
root@DS1: ~#_
```

7.7. Commandes diverses
7.7.1. La commande tr

• Cette commande de filtre permet d'effectuer des remplacements de caractères dans une chaine.
Exemples :
→ On souhaite transformer une chaine en minuscules :

```
root@DS1: ~#chaine="Bonjour, comment allez-VOUS aujourd'hui ?"
root@DS1: ~#echo $chaine | tr 'A-Z' 'a-z'
bonjour, comment allez-vous aujourd'hui ?
root@DS1: ~#_
```

→ Pour permettre l'utilisation de la commande set (cf. ci-dessous), il est nécessaire que le séparateur de champ sur une ligne soit l'espace, et non pas par exemple « : ». On souhaite

création d'un fichier passwd.txt qui introduit un espace à la place de « : » dans une copie de /etc/passwd :

```
root@DS1: ~#cat /etc/passwd | tr ":" " " > passwd.txt
root@DS1: ~#head passwd.txt
root x 0 0 root /root /bin/bash
daemon x 1 1 daemon /usr/sbin /usr/sbin/nologin
bin x 2 2 bin /bin /usr/sbin/nologin
sys x 3 3 sys /dev /usr/sbin/nologin
sync x 4 65534 sync /bin /bin/sync
games x 5 60 games /usr/games /usr/sbin/nologin
man x 6 12 man /var/cache/man /usr/sbin/nologin
lp x 7 7 lp /var/spool/lpd /usr/sbin/nologin
mail x 8 8 mail /var/mail /usr/sbin/nologin
news x 9 9 news /var/spool/news /usr/sbin/nologin
root@DS1: ~#
```

7.7.2. La commande set

Exemple :
→ On reprend un exemple de lecture de fichier analogue à celui de la page 6. La commande tr est utilisée à la place du recours à la variable prédéfinie IFS.

```
root@DS1: ~#vim read_set_tr.sh
```

```
#!/bin/bash
fichier="/etc/passwd"
cat $fichier | head | tr ":" " " | while true
do
        read ligne
        if [ "$ligne" = "" ] ; then break ; fi
        set $ligne
        echo $1
done
```

```
root@DS1: ~#sh read_set_tr.sh
root
daemon
bin
sys
sync
games
man
lp
mail
news
root@DS1: ~#
```