TP 16 et 17 : Scripts shell Linux

Sommaire

Malaussena Maxence

TP16 – Introduction aux scripts shell.

-Tout d'abord j'installe le paquet 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 :
  libgpm2 libsodium23 vim vim-runtime
0 mis à jour, 4 nouvellement installés, 0 à enlever et 43 non mis à jour.
Il est nécessaire de prendre 8 768 ko dans les archives.
Après cette opération, 41,5 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] o
```

-Puis je créer un script avec 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  34 30 janv. 11:14 un_script.sh
root@DS1: ~#
```

-Je l'exécute avec le shell de base :

```
root@DS1: ~#sh un_script.sh
jeu. 30 janv. 2025 11:15:39 CET
 11:15:39 up 5 min,  1 user,  load average: 0,03, 0,04, 0,00
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: ~#
```

-Puis en affichant la phase d'interprétation et la trace des commandes :

```
root@DS1: ~#bash -x un_script.sh
+ date
jeu. 30 janv. 2025 11:16:14 CET
+ uptime
 11:16:14 up 5 min,  1 user,  load average: 0,02, 0,03, 0,00
+ 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: ~#
```

-Et enfin 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  34 30 janv. 11:14 un_script.sh
root@DS1: ~#./un_script.sh
jeu. 30 janv. 2025 11:17:29 CET
 11:17:29 up 7 min,  1 user,  load average: 0,00, 0,02, 0,00
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: ~#
```

-J’affiche le contenu de la variable PATH :

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

-Je modifie le contenu de la variable PATH à partir du fichier /etc/bash.bashrc pour pouvoir exécuter le script à partir d’un répertoire quelconque :

```
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, overwrite the one in /etc/profile)
# but only if not SUDOing and have SUDO_PS1 set; then assume smart user.
if ! [ -n "${SUDO_USER}" -a -n "${SUDO_PS1}" ]; then
  PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi

# Commented out, don't overwrite xterm -T "title" -n "icontitle" by default.
# If this is an xterm set the title to user@host:dir
#case "$TERM" in
#xterm*|rxvt*)
#    PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
#    ;;
#*)
#    ;;
#esac

# enable bash completion in interactive shells
#if ! shopt -oq posix; then
#  if [ -f /usr/share/bash-completion/bash_completion ]; then
#    . /usr/share/bash-completion/bash_completion
#  elif [ -f /etc/bash_completion ]; then
#    . /etc/bash_completion
#  fi
#fi

# 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 12 DS1 tty1

DS1 login: root
Password:
Linux DS1 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 30 11:10:52 CET 2025 on tty1
root@DS1: ~#echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root
root@DS1: ~#_
```

-Maintenant je peux exécuter le script sans indiquer son chemin :

```
root@DS1: ~#un_script.sh
jeu. 30 janv. 2025 11:14:07 CET
 11:14:07 up 0 min,  1 user,  load average: 0,00, 0,00, 0,00
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: ~#_
```

-Je créer une variable et je l'utilise :

```
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: ~#_
```

-Je créer une variable, je l'affiche, idem pour toutes les variables (les 10 premières) et enfin je détruis une variable :

```
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: ~#A="bonjou Mr"
root@DS1: ~#echo $A
bonjou Mr
root@DS1: ~#set | head
A='bonjou Mr'
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_fignore:globasciiranges:globski
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 | rep A=
-bash: rep : commande introuvable
root@DS1: ~#set | grep A=
root@DS1: ~#
```

- Je créer un script qui affiche les paramètres :

```
#!/bin/bash
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: ~#
```

-J'écrit un script d'une autre manière :

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

```
#!/bin/bash
echo "Nom du répertoire a creer ?"
read nom_rep
if mkdir $nom_rep 2> /dev/null
then
        echo "Operation reussie"
else
        echo "Echec"
        fi
~
```

```
root@DS1: ~#sh creer_rep.sh
Nom du répertoire a creer ?
sauve
Operation reussie
root@DS1: ~#sh creer_rep.sh
Nom du répertoire a creer ?
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: ~#
```

-Je refais une alternative avec la commande test :

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

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

```
root@DS1: ~#sh heureux.sh
Etes-vous hereux ? oui
Bravo
root@DS1: ~#_
```

- Je créer 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] 1456
root@DS1: ~#touch flag
root@DS1: ~#Le fichier flagLe fichier flag existe

-bash: Le : commande introuvable
[1]+  Fini                    sh flag_existe.sh
root@DS1: ~#_
```

-Puis je code une autre version de ce programme :

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

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

```
root@DS1: ~#rm flag
root@DS1: ~#sh flag_existebis.sh &
[1] 1474
root@DS1: ~#touch flag
root@DS1: ~#Le fichier flaLe fichier flag existe

-bash: Le : commande introuvable
[1]+  Fini                    sh flag_existebis.sh
root@DS1: ~#_
```

-Je créer un autre script qui fait un décompte :

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

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

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

-J'écrit un script avec un menu grâce à l'instruction case (vue en php) :

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

```
#!/bin/bash

echo "1 - Afficher la date et l'heure"
echo "2 - Afficher la charge systeme"
echo "3 - Afficher 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 systeme
3 - Afficher 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: ~#_
```

-J'écris un autre script avec une boucle while(=tant que) :

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

-Je créer le fichier texte users.txt avec des login, mdp, nom d'utilisateur :

```
 GNU nano 7.2                                  user.txt *                                  S
kbeatini watson kevin 2ISO,SISR
acanonne navarone axel 2SIO,SISR
cmartinez malaucoccyx cedric
vgosse toutafait victor
jpichon pelican jonathan
rfumey weed raphael
vmanceau dakota vicent
```

-Puis je code un script qui parcourt le fichier créer précédemment, qui récupère les champs et les affiches :

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

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

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

-J'écris un script qui reçoit en premier argument le nom d'un répertoire (/test) puis, à partir du deuxième argument, un nombre quelconque de noms de fichiers :

```
#! /bin/bash
echo -e "Affichage avant\n"
echo "1er argument \$1 : $1"
echo "2eme argument \$2 : $2"
echo "3eme argument \$3 : $3"
echo "4eme arguement \$4 : $4"
echo "Tous les arguments \$* : $*"
echo -e "Nombre d'arguments \$
rep=$1
shift

echo -e "Affichage des variables 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'arguments \$# : $#\n"
mkdir $rep
cd $rep_
```

-Puis je mets le droit d'exécution :

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

1er argument $1 : /test
2eme argument $2 : f1
3eme argument $3 : f2
4eme arguement $4 : f3
Tous les arguments $* : /test f1 f2 f3 f4 f5 f6
```

-Après j'exécute le script et je vérifie la création du répertoire et des fichiers (PS je n'ai pas les screens).

-J'écrit un script utilisant une fonction de présentation avec passage d'arguments à la suite du nom de la fonction :

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

#Debut de 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

jeu. 30 janv. 2025 18:56:16 CET
==== Soyez attentif à la syntaxe
==== Bon courage
```

-Puis un script qui affiche un menu :

```
root@DS1: ~#vim menu2.sh_
```

```
#!/bin/bash

confirmation ()
{
        printf "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 ? 1
jeu. 30 janv. 2025 19:08:33 CET
```

TP17 – Introduction aux scripts shell : autoformation.

-Tout d'abord je code le 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`
```

```
root@DS1: ~#chmod a+x bonjourbis.sh
root@DS1: ~#./bonjourbis.sh
Bonjour root
Nous sommes le jeu. 30 janv. 2025 20:48:51 CET
Votre numéro d'utilisateur est 0
root@DS1: ~#
```

-Idem pour le 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 prenom"
fi
```

```
root@DS1: ~#sh bonjourter.sh
Syntaxe : bonjourter.sh nom prenom
root@DS1: ~#sh bonjourter.sh Malaussena Maxence
Bonjour Maxence Malaussena et bonne journée !
root@DS1: ~#
```

-Le echo affiche son argument texte entre guillemets :

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

-Je peux insérer les caractères spéciaux habituels, qui seront interprétés seulement si l'option -e suit echo (souf \n \b \t \a \c) :

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

-Je code le script saisie_clavier.sh :

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

-Puis je l'appelle :

```
root@DS1: ~#sh saisie_clavier.sh
Donnez votre prénom et votre nom :
Maxence Malaussena
Bonjour Maxence Malaussena
root@DS1: ~#_
```

-Je saisie dans la console une variable que j'identifie :

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

```
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: ~#_
```

-Puis je manipule des variables d'environnement PS(je n'ai pas fait tous les screens):

```
root@DS1: ~# [ -e ./fichier ]
root@DS1: ~#echo $?
1
root@DS1: ~#touch fichier
root@DS1: ~# [ -e ./fichier ]
root@DS1: ~#echo $?
0
root@DS1: ~# [ -s ./fichier ]
root@DS1: ~#echo $?
1
root@DS1: ~#date > fichier
root@DS1: ~# [-s ./fichier ]
-bash: [-s : commande introuvable
root@DS1: ~# [ -s ./fichier ]
root@DS1: ~#echo $?
0
```

```
root@DS1: ~#[ -r "/etc/passwd" ]
root@DS1: ~#echo $?
0
root@DS1: ~# [ -r "/etc/shadow" ]
root@DS1: ~#echo $?
0
root@DS1: ~#su - sio1
su: l'utilisateur sio1 n'existe pas ou l'entrée de l'utilisateur ne contient pas tous les champs requis
root@DS1: ~#su - sio
sio@DS1:~$ [ -r "/etc/shadow" ]
sio@DS1:~$ echo $?
1
sio@DS1:~$ [ -r "/etc/shadow" ] || echo "lecture du fichier interdite"
lecture du fichier interdite
sio@DS1:~$ [ -r "/etc/passwd" ]
sio@DS1:~$ echo $?
0
sio@DS1:~$
```

```
root@DS1: ~#f="/root" ; [ -d $f -a -x $f ] ; echo $?
0
root@DS1: ~#ls -ld /root
drwx------ 5 root root 4096 30 janv. 21:03 /root
root@DS1: ~#note=9 ; [ $note -lt 8 -o $note -ge 10 ] && echo "papapoupou"
root@DS1: ~#note=7 ; [ $note -lt 8 -o $note -ge 10 ] && echo "papapoupou"
papapoupou
root@DS1: ~#_
```

-Je code un script compte.sh pour verfier si le nom d'un compte est déjà pris :

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

```
if grep "^sio1" /etc/passwd
then
echo "sio1 a déjà un compte"
fi
~
```

Malhereusement je n'ai pas réussi a l'executer

-Je créer un script note.sh qui demande de saisir une note, et si celle si est supérieur ou égale a 16 cela affiche très bien !:

```
echo "Sasissez votre note :"
read note
if [ $note -gt 16 ]
then echo "C'est très bien !"
fi
~
```

```
root@DS1: ~#sh note.sh
Sasissez votre note :
17
C'est très bien !
root@DS1: ~#_
```

-Je créer un script avec vim, celui-ci a pour but de simuler le resultat du bac et nous dit si nous sommes admis ou la mention ou si il faut passer au ratrappages :

```
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
~
```

-Je créer le script devoir.sh qui verfie si il y a un devoir déposer :

```
fichier=/home/sio/devoir1.txt
if [ -f $fichier -a -r $fichier ]
then
echo "je vais vérifier ton devoir"
elif [ ! -e $fichier ]
then
echo "ton devoir n'existe pas !"
else
echo "je ne peux pas le lire !"
fi
~
```

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

```
root@DS1: ~#sh devoir.sh
je vais vérifier ton devoir
root@DS1: ~#_
```

-Je créer un script arguement.sh :

```
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: ~#_
```

-Je créer le script cash.sh et suppose quele script doive réagir différemment selon la valeur de $USER :

```
case $USER in
root)
        echo "Mes respects Monsieur le $USER" ;;
guest | sio?)
        echo "Salut $USER" ;;
*)
        echo "Bonjour $USER" ;;
esac
~
~
```

```
root@DS1: ~#chmod 755 cas.sh
root@DS1: ~#cas.sh
Mes respects Monsieur le root
root@DS1: ~#cp /root/cas.sh /bin/
root@DS1: ~#su - sio
sio@DS1:~$ cas.sh
Bonjour sio
sio@DS1:~$
```

-Je créer un script poursuite.sh qui attend une réponse soit oui/non de l'utilisateur :

```
echo "Voulez-vous vraiment exécuter 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 exécuter 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: ~#_
```

-La liste peut être explicite :

```
for nom in Benoit Nicolas Pierre
do
echo "$nom, bonjour"
done_
~
~
```

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

-Ou être calculé à partir d'une expression modèle :

```
if [ ! -e /tmp/sio ]
then
mkdir /tmp/sio
fi
for fich in /home/sio/*
do
        cp $fich /tmp/sio
done
~
```

```
root@DS1: ~#sh coppie.sh
root@DS1: ~#ls -l /tmp/sio
total 0
-rw-r--r-- 1 root root 0 30 janv. 21:47 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 :

```
cd /tmp/sio ; set *
for nom in $@
do echo $nom
done_
~
~
```

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

-Je créer un script true.sh qui dit bonjour toutes les secondes :

```
while true
do
echo "Bonjour Mr $USER"
sleep 1
done_
~
```

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

-Lecture des lignes d'un fichier :

Je créer un fichier sortie.sh :

```
fich="/etc/passwd"
grep "^sio" $fich | while true
do
        read ligne
        if [ "$ligne" = "" ] ; then break ; fi
        echo $ligne
done
~
```

```
root@DS1: ~#sh sortie.sh *
sio:x:1008:1008:,,,:/home/sio:/bin/bash
root@DS1: ~#_
```

-Dans celle-ci je 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 « : » :

```
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: ~#
```

-Je créer un fichier read_set_tr.sh :

```
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: ~#
```