BTS 1 - Services Informatiques aux Organisations

TP16 & TP17 – Scripts shell Linux

Table des matières

1. TP16 – Introduction aux scripts shell

1.1 L'exécution de scripts

Installation de l'éditeur Vim avec la commande **apt-get install vim**

```
root@DEB10Server: ~#vim un_script.sh
-bash: vim : commande introuvable
root@DEB10Server: ~#apt-get install vim
vim                 vim-bitbake         vim-gitgutter       vim-khuno           vim-poke            vim-snippets        vim-tjp
vim-addon-manager   vim-command-t       vim-git-hub         vim-lastplace       vim-puppet          vim-solarized       vim-tlib
vim-addon-mw-utils  vim-common          vim-gtk3            vim-latexsuite      vim-python-jedi     vim-subtitles       vim-ultisnips
vim-airline         vim-ctrlp           vim-gui-common      vim-ledger          vim-rails           vim-syntastic       vim-vader
vim-airline-themes  vim-doc             vim-haproxy         vim-migemo          vim-redact-pass     vim-syntax-gtk      vim-vimerl
vim-ale             vim-eblook          vim-icinga2         vim-motif           vim-runtime         vim-tabular         vim-vimerl-syntax
vim-athena          vim-editorconfig    vimix               vim-nox             vim-scripts         vim-textobj-user    vim-voom
vim-autopep8        vim-fugitive        vim-julia           vim-pathogen        vim-snipmate        vim-tiny            vim-youcompleteme
root@DEB10Server: ~#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 0 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
Ign :1 http://deb.debian.org/debian bookworm/main amd64 libgpm2 amd64 1.20.7-10+b1
Ign :2 http://deb.debian.org/debian bookworm/main amd64 libsodium23 amd64 1.0.18-1
Ign :3 http://deb.debian.org/debian bookworm/main amd64 vim-runtime all 2:9.0.1378-2
Ign :4 http://deb.debian.org/debian bookworm/main amd64 vim amd64 2:9.0.1378-2
Ign :1 http://deb.debian.org/debian bookworm/main amd64 libgpm2 amd64 1.20.7-10+b1
Ign :2 http://deb.debian.org/debian bookworm/main amd64 libsodium23 amd64 1.0.18-1
Ign :3 http://deb.debian.org/debian bookworm/main amd64 vim-runtime all 2:9.0.1378-2
Ign :4 http://deb.debian.org/debian bookworm/main amd64 vim amd64 2:9.0.1378-2
Ign :1 http://deb.debian.org/debian bookworm/main amd64 libgpm2 amd64 1.20.7-10+b1
Ign :2 http://deb.debian.org/debian bookworm/main amd64 libsodium23 amd64 1.0.18-1
Ign :3 http://deb.debian.org/debian bookworm/main amd64 vim-runtime all 2:9.0.1378-2
Ign :4 http://deb.debian.org/debian bookworm/main amd64 vim amd64 2:9.0.1378-2
Err :1 http://deb.debian.org/debian bookworm/main amd64 libgpm2 amd64 1.20.7-10+b1
  Erreur temporaire de résolution de « deb.debian.org »
0% [En cours]
```

Création d'un script avec l'éditeur Vim : **vim un_script.sh**

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

Modification du script **un_script.sh**

```
#!/bin/bash
date
uptime
uname -a
```

Liste des fichiers avec leurs droits : **ls -l**

```
root@DS1: ~#ls -l
total 16
-rw-r--r-- 1 root root  0 20 déc.  16:24 cal.txt
-rw-r--r-- 1 root root 74 20 déc.  16:06 etudiants.txt
-rw-r--r-- 1 root root 74 20 déc.  16:11 notes.csv
-rw-r--r-- 1 root root 74 20 déc.  16:07 prenoms_tries
-rw-r--r-- 1 root root 33 27 janv. 12:55 un_script.sh
root@DS1: ~#
```

Exécution du script : **sh un_script.sh**

```
root@DS1: ~#sh un_script.sh
mer. 29 janv. 2025 14:53:47 CET
 14:53:47 up  1:12,  1 user,  load average: 0,00, 0,00, 0,00
Linux DS1 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-11-22) x86_64 GNU/Linux
root@DS1: ~#_
```

Exécution du script en affichant la phase d'interprétation avec la trace des commandes : **bash -x un_script.sh**

```
root@DS1: ~#bash -x un_script.sh
+ date
mer. 29 janv. 2025 14:54:55 CET
+ uptime
 14:54:55 up  1:13,  1 user,  load average: 0,00, 0,00, 0,00
+ uname -a
Linux DS1 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-11-22) x86_64 GNU/Linux
root@DS1: ~#_
```

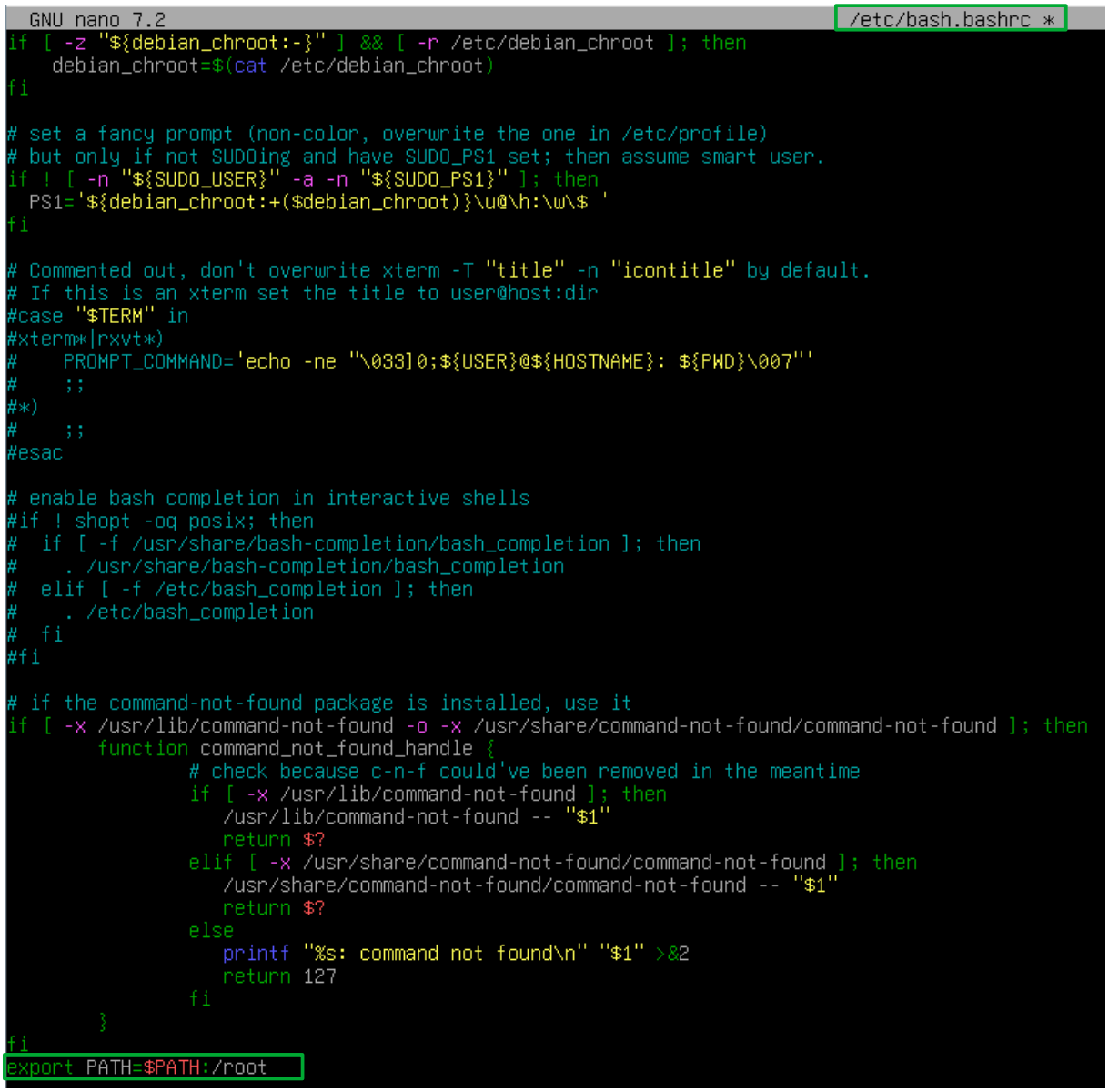
Exécution du script en tant que commande depuis le répertoire courant après avoir attribué les droits d'exécution : **chmod +x un_script.sh**

```
root@DS1: ~#./un_script.sh
mer. 29 janv. 2025 14:56:47 CET
 14:56:47 up  1:15,  1 user,  load average: 0,00, 0,00, 0,00
Linux DS1 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-11-22) x86_64 GNU/Linux
root@DS1: ~#chmod +x un_script.sh
root@DS1: ~#ls -l
total 16
-rw-r--r-- 1 root root  0 20 déc.  16:24 cal.txt
-rw-r--r-- 1 root root 74 20 déc.  16:06 etudiants.txt
-rw-r--r-- 1 root root 74 20 déc.  16:11 notes.csv
-rw-r--r-- 1 root root 74 20 déc.  16:07 prenoms_tries
-rwxr-xr-x 1 root root 33 27 janv. 12:55 un_script.sh
root@DS1: ~#./un_script.sh
mer. 29 janv. 2025 14:57:22 CET
 14:57:21 up  1:15,  1 user,  load average: 0,00, 0,00, 0,00
Linux DS1 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-11-22) x86_64 GNU/Linux
root@DS1: ~#
```

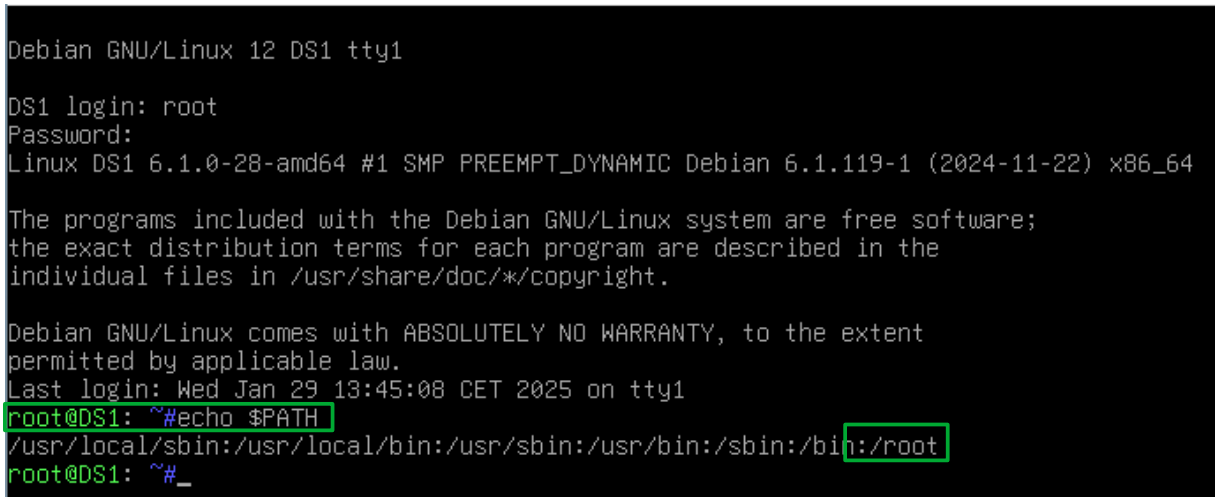
Affichage du contenu de la variable PATH (chemin jusqu'à bin) : **echo $PATH**

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

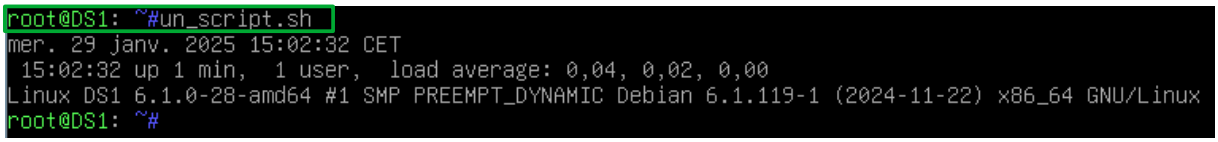
Modification du contenu de la variable PATH dans le fichier **/etc/bash.bashrc** pour exécuter le script à partir d'un répertoire spécifique.



Affichage du contenu de la variable PATH (chemin jusqu'à root) : **echo $PATH**

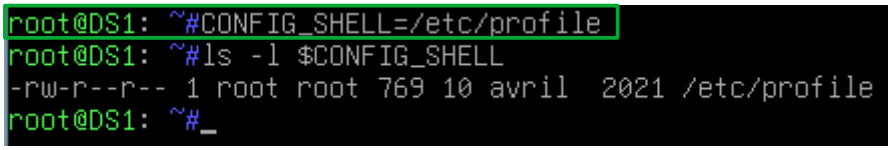


Exécution du script sans indiquer son chemin : **un_script.sh**



1.2 Les variables

Création de la variable CONFIG_SHELL avec la valeur /etc/profile :
CONFIG_SHELL=/etc/profile



Création de la variable A et affichage des 10 premières variables : **set | head**

Création d'un script interactif pour saisir une variable et l'afficher : **vim bonjour.sh**

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

Modification du fichier **bonjour.sh**

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

Exécution du script : **sh bonjour.sh**

```
root@DS1: ~#sh bonjour.sh
Votre nom ? Provenzano
Bonjour Mme Provenzano
root@DS1: ~#
```

Création d'un script qui affiche les paramètres : **vim param.sh**

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

Modification du fichier **param.sh**

```
#!/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   : $#"
~
~
~
```

Attribution des droits d'exécution : **chmod +x param.sh**, puis exécution **./param.sh**

```
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: ~#_
```

1.3 Les structures de contrôle

1.3.1 Structures conditionnelle

Création d'un script alternative : **vim creer_rep.sh**

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

Modification du fichier **creer_rep.sh**

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

Exécution du script : **sh creer_rep.sh**

```
root@DS1: ~#sh creer_rep.sh
Nom du réperoitre a creer ?
sauve
Operation reussie
root@DS1: ~#ls
bonjour.sh  cal.txt  creer_rep.sh  etudiants.txt  notes.csv  param.sh  prenoms_tries  sauve  un_script.sh
root@DS1: ~#_
```

Création d'un script alternative : **vim heureux.sh**

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

Modification du fichier **heureux.sh**

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

Exécution du script : **sh heureux.sh**

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

1.3.2 Boucle while

Création d'un script qui teste l'existence du fichier « flag » toutes les dix secondes. Le script signale sa présence et se termine : **vim flag_existe.sh**

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

Modification du fichier **flag_existe.sh**

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

Suppression du flag puis activation du script : **sh flag_existe.sh &**

```
root@DS1: ~#rm flag
rm: impossible de supprimer 'flag': Aucun fichier ou dossier de ce type
root@DS1: ~#sh flag_existe.sh &
[2] 554
root@DS1: ~#touch flag
root@DS1: ~#Le fichier flag esiste
Le fichier flag esiste
^C
[1]-  Fini                    sh flag_existe.sh
[2]+  Fini                    sh flag_existe.sh
root@DS1: ~#_
```

→ La commande **sleep** crée une temporisation. Sa durée est spécifiée en secondes en argument.
→ L'activation du script « flag_existe.sh » est réalisée en tâche de fond grâce au caractère **&**.

Une autre version du programme : **flag_existebis.sh**

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

Modification du fichier **flag_existebis.sh**

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

Suppression du flag puis activation du script : **sh flag_existebis.sh &**

```
root@DS1: ~#rm flag
rm: impossible de supprimer 'flag': Aucun fichier ou dossier de ce type
root@DS1: ~#sh flag_existebis.sh &
[1] 582
root@DS1: ~#touch flag
root@DS1: ~#Le fichier flag existe
^C
[1]+  Fini                    sh flag_existebis.sh
root@DS1: ~#_
```

1.3.3 Boucle for

Création d'un script qui permet d'effectuer un décompte de secondes : **vim boucle_for.sh**

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

Modification du fichier **boucle_for.sh**

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

Exécution du script : **sh boucle_for.sh**

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

1.3.4 Choix multiple

Création d'un script pour écrire un menu avec l'instruction case : **vim menu.sh**

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

Modification du fichier **menu.sh**

```
#!/bin/bash

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

echo -n "Votre choix ? "
read choix

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

Exécution du script : **sh menu.sh**

```
root@DS1: ~#sh menu.sh
1 - Afficher la date et l'heure
2 - Afficher la charge systeme
3 - Afficher la version du systeme
Votre choix ? 3
Linux DS1 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-11-22) x86_64 GNU/Linux
root@DS1: ~#_
```

→ Le cas * n'est pas obligatoire mais utile. Il correspond à une chaîne quelconque et propose un choix alternatif général en cas de saisie inappropriée.

1.3.5 La commande let

Création d'un script en utilisant une boucle while et en mettant en oeuvre l'incrémentation d'une variable via la commande **let** et la nommée **plusun.sh : vim plusun.sh**

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

Modification du fichier **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
~
```

La commande **bash plusun.sh** exécute le script **plusun.sh** avec l'interpréteur Bash. Elle demande à Bash de lire et d'exécuter les instructions écrites dans le script.

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

Attribution des droits d'exécution : **chmod +x plusun.sh**, puis exécution **plusun.sh**

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

1.3.6 Lecture d'un fichier et commende set

Création d'un fichier texte **users.txt** contenant quelques lignes comportant les champs suivants : login, mot-de-passe, nom de l'utilisateur et nom des groupes secondaires. Les champs sont séparés par un espace et chaque champ ne comporte pas d'espace.

```
root@DS1: ~#nano users.txt
```

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

Création d'un script qui affiche les paramètres : **vim lecture.sh**

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

Modification du fichier **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
~
```

Exécution du script : **sh lecture.sh**

```
root@DS1: ~#sh lecture.sh
login=kbeatini, mdp=watson, groupe=2SIO,SISR, nom=kevin
login=acanonne, mdp=navarone, groupe=2SIO,SISR, nom=axel
login=cmartinez, mdp=malaucoccyx, groupe=2SIO,SISR, nom=cedric
login=vgosse, mdp=toutafait, groupe=2SIO,SISR, nom=victor
login=jpichon, mdp=pelican, groupe=2SIO,SISR, nom=jonathan
login=rfumey, mdp=weed, groupe=2SIO,, nom=raphael
login=vmanceau, mdp=dakota, groupe=2SIO,SISR, nom=vincent
root@DS1: ~#_
```

1.3.7 Commande shift et boucle for

Création d'un script qui affiche les paramètres : **vim decal.sh**

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

Modification du fichier **decal.sh**

```
#!/bin/bash
# Affichage des variables paramètres de position avant décalage avec 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 command shift
rep=$1
shift
# Affichage des variables après exécution 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'arguments \$# : $#\n"
# Création du répertoire et déplacemnt dans le répertoire créé
mkdir $rep
cd $rep
# Création des fichiers dans le répertoire créé
for fichier i $*
do
        touch $fichier
        echo "Fichier $fichier créé"
done
~
```

Exécution du script : **./sh decal.sh**

```
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
root@DS1: ~#
```

Vérification de la création du répertoire et des fichiers : **ls -l /test**

```
root@DS1: ~#ls -l /test
total 0
-rw-r--r-- 1 root root 0 30 janv. 16:34 f1
-rw-r--r-- 1 root root 0 30 janv. 16:34 f2
-rw-r--r-- 1 root root 0 30 janv. 16:34 f3
-rw-r--r-- 1 root root 0 30 janv. 16:34 f4
-rw-r--r-- 1 root root 0 30 janv. 16:34 f5
-rw-r--r-- 1 root root 0 30 janv. 16:34 f6
root@DS1: ~#_
```

1.4 Les sous-programmes

Création d'un script qui utilise une fonction de présentation avec passage d'arguments à la suite de nom de la fonction (variable paramètres de position) : **vim affiche.sh**

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

Modification du fichier **affiche.sh**

```
presentation ()
{
        for i
        do
                echo "==== $i"
        done
        echo
}
# Début du programme
presentation "Bonjour" "Ce matin" "nous etudions" "les structures de contrôle" "ainsi que" "les fonctions"
date
presentation "Soyez attetif à la syntaxe" "Bon courage"_
~
~
```

Exécution du script : **sh affiche.sh**

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

jeu. 30 janv. 2025 16:41:44 CET
==== Soyez attetif à la syntaxe
==== Bon courage

root@DS1: ~#_
```

→ En mettant les arguments entre guillemets, on les délimite. Ainsi, dans le premier appel de la fonction, il y a, a priori, sept mots donc, en principe, sept arguments En réalité, avec les guillemets, il n'y en a que quatre.

Création d'un script qui affiche un menu. La sortie du programme est controlée par une fonction qui demande une confirmation : **vim menu2.sh**

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

Modification du fichier **menu2.sh**

```
#!/bin/bash

confirmation ()
{
        printf "Etes-vous sur $* (y/n) ? "
        read reponse
        if [ $reponse = "y" ] ;then return 0 ; else return 1 ; fi
}
while :
do
        clear
        echo "1 - Afficher la date et l'heure"
        echo "2 - Afficher la charge système"
        echo "3 - Afficher la version du système"
        echo "99 - FIN"
        printf "Votre choix ? "; read choix
        case "$choix" in
        1) date ;;
        2) uptime ;;
        3) uname -a ;;
        99)
                if confirmation "de vouloir quiter" ;then
                        break
                fi
                ;;
        *) echo "Choix incorrect" ;;
        esac
        sleep 3
done
exit 0
~
```

Exécution du script : **sh menu2.sh**

```
root@DS1: ~#sh menu2.sh
1 - Afficher la date et l'heure
2 - Afficher la charge système
3 - Afficher la version du système
99 - FIN
Votre choix ? 99
Etes-vous sur de vouloir quiter (y/n) ? y
root@DS1: ~#_
```

→ Réponse si le choix est 99

Exécution du script : **sh menu2.sh**

```
root@DS1: ~#sh menu2.sh
1 - Afficher la date et l'heure
2 - Afficher la charge système
3 - Afficher la version du système
99 - FIN
Votre choix ? 9
Choix incorrect
```

→ Réponse si le choix est autre que 1,2,3 ou 99

2. TP17 – Introduction aux scripts shell : autoformation

2.1 Saisies et exécution du script

Création d'un script : **vim bonjourbis.sh**

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

Modification du fichier **bonjourbis.sh**

→ La variable d'environnement $USER contient le nom de login.
L'option -n empêche le passage à la ligne. Le ; sert de séparateur des commandes sur une même ligne.
→ La dernière ligne du script recherche $USER en début de ligne dans le fichier passwd puis extrait l'uid (3ème champ) et l'affiche.

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

Attribution des droits d'exécution : **chmod a+x bonjourbis.sh**, puis exécution.

```
root@DS1: ~#chmod a+x bonjourbis.sh
root@DS1: ~#./bonjourbis.sh
Bonjour root
Nous sommes le : ven. 31 janv. 2025 09:32:27 CET
Votre numéro d'utilisateur est 0
root@DS1: ~#
```

→ Pour lancer l'exécution du script, tapez ./bonjourbis.sh, ./ indiquant le répertoire courant (ou bien indiquez le chemin absolu à partir de la racine) ; ceci dans le cas où le répertoire contenant le script n'est pas listé dans le PATH.

Création d'un script : **vim bonjourter.sh**

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

Modification du fichier **bonjourter.sh**

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

Exécution du script avec et sans arguments : **sh bonjourter.sh**

```
root@DS1: ~#sh bonjourter.sh
Syntaxe : bonjourter.sh nom prénom
root@DS1: ~#sh bonjourter.sh Provenzano Nina
Bonjour Nina Provenzano et bonne journée !
root@DS1: ~#_
```

2.2 Entrées-Sorties

Ce sont les voies de communication entre le programme bash et la console :

• **echo** affiche son argument texte entre guillemets sur la sortie standard c'est-à-dire l'écran. La validation d'une commande **echo** provoque un saut de ligne :

• On peut insérer les caractères spéciaux habituels, qui seront interprétés seulement si l'option **-e** suit **echo** :
\n (saut ligne), **\b** retour arrière), **\t** (tabulation), **\a** (alarme), **\c** (fin sans saut de ligne)

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

• **read** permet l'affectation directe par lecture de la valeur saisie sur l'entrée standard au clavier.

Création d'un script : **vim saisie_clavier.sh**

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

Modification du fichier **saisie_clavier.sh**

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

Exécution du script : **sh saisie_clavier.sh**

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

2.3 Les variables BASH

Variables programmeur

De façon générale, elles sont de type texte. On distingue les variables définies par le programmeur et les variables systèmes.

• Syntaxe : **variable=valeur** (le signe = ne doit pas être entouré d'espace)

• On peut initialiser une variable à une chaine vide : **chaine_vide=**

• Si **valeur** est une chaine avec des espaces ou des caractères spéciaux, elle doit être entourée de **" "** ou de **' '**.

• Le caractère \ permet de masquer le sens d'un caractère spécial comme **"** ou **' .**

. • Référence à la valeur d'une variable : faire précéder son nom du symbole **$**.

```
chaine="Bonjour à tous"
echo $chaine
```

• Pour afficher toutes les variables : **set**

• Pour empêcher la modification d'une variable, invoquer la commande **readonly**.

Substitution de variable

Si une chaine contient la référence à une variable, le shell doit d'abord remplacer cette référence par sa valeur avant d'interpréter la phrase globalement. Cela est effectué par l'utilisation de **" "**, dans ce cas obligatoire à la place de **' '**.

```
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 dis : $salut"
Alors moi je dis : Bonjour à tous !
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 : "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: ~#
```

Opérateur {} dans les variables

Dans certains cas en programmation, on peut être amené à utiliser des noms de variables dans d'autres variables. Comme il n'y a pas de substitution automatique, la présence de {} force l'interprétation des variables incluses.

```
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: ~#
```

Variables d'environnement

Ce sont les variables systèmes ($HOME, $PATH, $USER, $SHELL, $PWD…) dont la liste est consultable par la commande env | more.

2.4 La commande test

Tester un fichier :

option	signification
-e	il existe
-f	c'est un fichier normal
-d	c'est un répertoire
-r | -w | -x	il est lisible | modifiable | exécutable
-s	il n'est pas vide

Exemples :

```
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 ]
root@DS1: ~#echo $?
0
root@DS1: ~#
```

```
root@DS1: ~#[ -r "/etc/passwd" ]
root@DS1: ~#echo $?
0
root@DS1: ~#[ -r "/etc/passwd" ]
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: ~#adduser sio1
Ajout de l'utilisateur « sio1 » ...
Ajout du nouveau groupe « sio1 » (1002) ...
Ajout du nouvel utilisateur « sio1 » (1002) avec le groupe « sio1 » (1002) ...
Création du répertoire personnel « /home/sio1 » ...
Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour sio1
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []: sio1
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Cette information est-elle correcte ? [O/n]o
Ajout du nouvel utilisateur « sio1 » aux groupes supplémentaires « users » ...
Ajout de l'utilisateur « sio1 » au groupe « users » ...
root@DS1: ~#_
```

```
root@DS1: ~#su - sio1
sio1@DS1:~$ [ -r "/etc/shadow" ]
sio1@DS1:~$ echo $?
1
sio1@DS1:~$ [ -r "/etc/shadow" ] || echo "lecture du fichier interdite"
lecture du fichier interdite
sio1@DS1:~$ [ -r "/etc/shadow" ]
sio1@DS1:~$ echo $?
1
sio1@DS1:~$ _
```

Tester une chaîne :

option	signification
-z | -n	la chaine est vide / n'est pas vide
= | !=	les chaines comparées sont identiques | différentes

Exemples :

```
root@DS1: ~#[ -z "est-ce que la chaîe est vide ?" ] ; echo $?
1
root@DS1: ~#ch="Bonjour" ; [ "$ch" = "bonjour" ] ; echo $?
1
root@DS1: ~#[ $USER != "root" ] && echo "l'utilisateur n'est pas le \"root\" !"
root@DS1: ~#[ $USER != "root" ] && echo "l'utilisateur n'est pas le \"root\" !" || echo "l'utilisateur est le \"root\" !"
l'utilisateur est le "root" !
root@DS1: ~#su - sio1
sio1@DS1:~$ [ $USER != "root" ] && echo "l'utilisateur n'est pas le \"root\" !" || echo "l'utilisateur est le \"root\" !"
l'utilisateur n'est pas le "root" !
sio1@DS1:~$ _
```

Tester un nombre :

option	signification
-eq | -ne	égal | différent
-lt | -gt	strict. inf | strict. sup
-le | -ge	inf ou égal | sup ou égal

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 :

option	valeur
[expr1 -a expr2]	0 si les 2 expressions sont vraies (**a**nd)
[expr1 -o expr2]	0 si l'une des 2 expressions est vraie (**o**r)
[! expr1]	négation

Exemples :

```
root@DS1: ~#f="/root" ; [ -d $ -a -x $f ] ; echo $?
1
root@DS1: ~#ls -ld /root
drwx------ 5 root root 4096 31 janv. 15:14 /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: ~#
```

2.5 Structures conditionnelles

2.5.1 Conditionnelle simple

Création d'un script : **vim compte.sh**

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

Modification du fichier **compte.sh**

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

Exécution du script : **sh compte.sh**

```
root@DS1: ~#sh compte.sh
sio1:x:1002:1002:sio1,,,:/home/sio1:/bin/bash
sio1 a déà un compte
root@DS1: ~#
```

Création d'un script : **vim note.sh**

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

Modification du fichier **note.sh**

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

Exécution du script : **sh note.sh**

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

1.5.2 Conditionnelle imbriquées

Création d'un script : **vim examen.sh**

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

Modification du fichier **examen.sh**

```
echo "Saissisez 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
~
```

Création d'un script : **vim devoir.sh**

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

Modification du fichier **devoir.sh**

```
fichier=/home/sio1/devoir1.txt
if [ -f $fichier -a -r $fichier ]
then
echo "Je vais vérifier ton devoir"
elif [ ! -e $fichier ]
then
echo "Ton devoir n'exise pas !"
else
echo "Je ne peux pas le lire !"
fi
~
```

Exécution du script : **sh devoir.sh**

```
root@DS1: ~#sh devoir.sh
Je vais vérifier ton devoir
root@DS1: ~#su - sio1
sio1@DS1:~$ touch devoir.txt
sio1@DS1:~$ exit_
```

Exécution du script avec les modifications : **sh devoir.sh**

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

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

Création d'un script : **vim argument.sh**

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

Modification du fichier **argument.sh**

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

Exécution du script : **sh argument.sh**

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

1.5.3 Choix multiples

Création d'un script : **vim cas.sh**

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

Modification du fichier **cas.sh**

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

Attribution des droits : **chmod 775 cas.sh**, puis exécution **cas.sh**

```
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 - sio1
sio1@DS1:~$ cas.sh
Salut sio1
sio1@DS1:~$ _
```

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

Création d'un script : **vim poursuite.sh**

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

Modification du fichier **poursuite.sh**

```
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
```

Exécution du script : **sh poursuite.sh**

```
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: ~#
```

2.6 Structures itératives

2.6.1 Boucle for

Création d'un script : **vim liste.sh**

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

Modification du fichier **liste.sh**

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

Exécution du script : **sh liste.sh**

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

Création d'un script : **vim copie.sh**

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

Modification du fichier **copie.sh**

```
if [ ! -e /tmp/sio1 ]
then
mkdir /tmp/sio1
fi
for fich in_/home/sio1/*
do
        cp $fich /tmp/sio1
done
~
```

Exécution du script : **sh copie.sh**

```
root@DS1: ~#sh copie.sh
root@DS1: ~#ls -l /tmp/sio1
total 0
-rw-r--r-- 1 root root 0 31 janv. 16:08 devoir1.txt
-rw-r--r-- 1 root root 0 31 janv. 16:08 devoir.txt
root@DS1: ~#_
```

Création d'un script : **vim sanslistprecise.sh**

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

Modification du fichier **sanslistprecise.sh**

```
cd /tmp/sio1 ; set *
for nom  in $@
do echo $nom
done
~
```

Exécution du script : **sh sanslistprecise.sh**

```
root@DS1: ~#sh sanslistprecise.sh
devoir.txt
devoir1.txt
root@DS1: ~#_
```

2.6.2 Boucle while

Création d'un script : **vim true.sh**

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

Modification du fichier **true.sh**

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

Exécution du script : **sh true.sh**

```
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: ~#_
```

Création d'un script : **vim true1.sh**

```
root@DS1: ~#vim true1.sh
```

Modification du fichier **true1.sh**

```
fich=/etc/passwd
while read ligne
do
echo $ligne
more
done < $fich_
~
```

Exécution du script : **sh true1.sh**

```
root@DS1: ~#sh true1.sh
```

```
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
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:100:107::/nonexistent:/usr/sbin/nologin
nina:x:1000:1000:nina,,,:/home/nina:/bin/bash
guest:x:1001:1001:guest,,,:/home/guest:/bin/bash
tcpdump:x:101:109::/nonexistent:/usr/sbin/nologin
bind:x:102:110::/var/cache/bind:/usr/sbin/nologin
sio1:x:1002:1002:sio1,,,:/home/sio1:/bin/bash
```

Création d'un script : **vim sortie.sh**

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

Modification du fichier **sortie.sh**

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

Exécution du script : **sh sortie.sh**

```
root@DS1: ~#sh sortie.sh
sio1:x:1002:1002:sio1,,,:/home/sio1:/bin/bash
root@DS1: ~#
```

2.7 Commande diverses

2.7.1 La commande tr

Cette commande de filtre permet d'effectuer des remplacements de caractères dans une chaine.

```
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: ~#
```

```
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: ~#
```

2.7.2 La commande set

Cette commande permet de séparer une ligne en une liste de mots, chacun de ces mots étant affecté à une variable positionnelle. Le caractère de séparation est l'espace.

Création d'un script : **vim read_set_tr.sh**

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

Modification du 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
~
```

Exécution du script : **sh read_set_tr.sh**

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