Conditions et Boucles en Shell
Exemples

0. Préliminaires :

- installer l'environnement Linux (voir la fiche « Shell sous windows »)
- on édite, à l'aide d'un éditeur (nano par exemple) un nouveau fichier exercice1.sh dans lequel on écrit les 3 ligne suivantes : nano exercice1.sh

```
# !/bin/bash
variable="Bonjour le monde"
echo $variable
```

- On rend exécutable le script exercice1.sh

```
chmod u+x exercice1.sh
```

- On exécute le script

```
./exercice1.sh
```

1. Notions de base:

- commentaire

```
# ligne commentaire
```

- shebang

```
#!/bin/bash   # indiquer en début du script le shell utilisé, ici le bash au lieu de sh, csh, ksh, ...
```

- variables : a ou bien $a ?
toujours $a sauf :

```
a=1       # affectation
read a    # lecture
for a in liste   # boucle for
```

- affectations et cumul

```
a=5         # pas d'espaces de part et d'autres de l'opérateur =
b=6
(( x = 1 ))       # affectations en langage C avec ou sans espaces entre doubles parenthèses
a="$a $b"  # cumul dans a de la valeur de a et de b
a="$b $a"  # cumul dans a de la valeur de b et de a
```

- écriture à l'écran ou vers un ficher

```
echo "Bonjour le monde ! "    # affichage à l'écran
echo "la valeur de a est $a"
echo "phrase vers le fichier" > fichier
```

- lecture au clavier ou à partir d'un fichier :

```
read a    # saisie d'une phrase dans a y compris les espaces
read a b   # saisie du premier mot dans a et le reste de la phrase dans b
read a < fichier
read a b < fichier
```

- évaluation d'expressions et commandes

```
a=1
a=$(( a+1 ))        # $(( expression à évaluer )) avec  $a ou a dans l'expression
```

```
echo " $( ls ) "    # $( commande à exécuter )
```

3. condition if

- commande if ou test

```
# if [[ -f test1.c ]]
if test -f test1.c
then
   echo 'fichier test1.c existe.......'
else
   echo 'fichier test1.c does not exist.....'
fi
```

- tester les fichiers et répertoires

```
#if [ -e $nomfichier ]  le fichier existe
#if [ -d $nomfichier ]  le fichier est un répertoire
#if [ -f $nomfichier ]  le fichier est un document
```

comparer chaines de caractères

```
variable="Bonjour"
chaine="Bonjour"
if [ $variable == "Bonjour" ]
then
   echo "la variable est la chaîne Bonjour"
else
   echo "variable vide"
fi

a="abcdefg"
b="hijklmnopqrst"
if [[ $a < $b ]] ; then
  echo "la chaine $a précède alphabétiquement $b "
fi

if [ $a > $b ] ; then    # doubles ou simples crochets
  echo " $a après $b dans l'ordre alphabétique "
fi
```

- opérateurs logiques and or : -a -o

```
a=0
(( $a != 0 )) && echo "$a est non nul" || echo "$a est null"

a=-1
if [ $a -gt 0 -a $a -lt 10 ]
then
  echo "$a est entre les bornes"
else
  echo "$a non inclus"
fi
```

4. boucle for

- afficher les arguments

```
for a in $*   #ou bien for a
do
 echo -n "$a "
 # si la 'liste' est manquante, alors la boucle opère sur '$@' "
done
echo " "
```

- affichage de fichiers ayant l'extension .c

```
for f in *.c
do
        #file $f
        ls -l $f
done
```

- afficher les fichiers avec détails

```
for f in $(ls *.sh)
do
  file $f ;
done
```

- afficher les noms fichiers

```
for i in /var/*
do
        echo $i
done
```

exemple : renommer tous les fichiers du répertoire test dont le nom contient un espace en remplaçant l'espace par un tiret :

```
for fichier in test/*\ *; do
  mv "$fichier" "${fichier// /_}"
done
```

- boucle for infinie :

ermet à un script de continuer à s'exécuter jusqu'à une interruption par CTRL+C ou kill, etc.

```
for (( ; ; ))
do
   echo " boucle for infinie en cours"
   sleep 1
done
```

- quitter une boucle avec break

```
for x in zero un deux trois; do
  if [[ "$x" == "deux" ]]; then
    break
  fi
  echo "x: $x"
done
```

- quitter la boucle avec exit (quitte aussi le programme)

```
for (( x=1; x<=10; x++ ))
do
   #if(($x%5 == 0)) ; then
   #if [[ $x == 5 ]] ; then
   if [[ $(($x%5)) == 0 ]] ; then
      #exit
      Break
   fi
   echo "Boucle $x fois"
done
```

sauter une itération avec continue

```
for i in {1..5}; do
  if [ "$i" == '2' ] ; then
  #if [[ "$i" == '2' ]] ; then
  #if (( "$i" == '2' )); then
    continue
  fi
  echo "Nombre: $i"
done
```

tous les paramètres du programme : $*

```
for f in $*
do
  file "$f"
done
```

5. boucle while et exemples :

```
while [ "$x" -lt "$N" ]     # espaces pour crochets
do
  echo -n "$x "        # -n supprime le retour chariot.
  (( x += 1 ))
  x=$((x+1))      #  x=`expr $x + 1`
                  # x=$(($x + 1))
done
```

- boucle while syntaxe c

```
((a = 1))      # a=1
while (( a <= N ))  # Doubles parenthèses, et pas de "$" devant la variable.
do
  echo -n "$a "
  ((a += 1))   # let "a+=1"
done
```

- boucle while double crochets

```
i=0
#while [[ $i -le 5 ]]
#while [ $i -le 5 ]
```

```
while (( $i <= 5 ))
do
  echo Nombre: $i
  ((i++))
  if [[ "$i" == '2' ]]; then
    break
  fi
done
```

- boucle pour lecture dans un fichier

```
fichier=test1.c
while read -r ligne
# while IFS= read -r ligne; # le paramètre IFS= permet de garder les espaces de début et de fin
do
  echo $ligne
done < "fichier"
```

- boucle while infinie

```
# while :
#do
#  echo "Une boucle infinie While"
#done
```

- autre boucle while infinie

```
#while [[ 1 == 1 ]]
#do
#  echo "Une boucle infinie While"
#done
```

- Tableau et boucle

```
T=(1 2 3 4 5)
i=0
n=${#T[@]}
echo taille $n
for i in {0..4}
# for (( i=0; i< n; i++ ))
do
  echo "${T[i]}"
  #echo "$i"
done
```

6. Boucle until

```
x=0
# until [ $x -gt 5 ]   # ou bien
until (( $x > 5 )) ; do
  echo x: $x
  ((x++))
done
```

- boucle until infinie

until ((1 == 2)) ou bien until [1 == 2]