```
#!/bin/bash

#      http://www.ss64.com/bash
#      http://www.gentoo.org/doc/en/articles/bash-by-example-p1.xml
#      http://www.gentoo.org/doc/en/articles/bash-by-example-p2.xml
#      http://www.tldp.org/LDP/abs/html/internalvariables.html
#      http://www.justlinux.com/nhf/Shells/Configuring_the_Bash_Shell.html
#
# L'ensemble des informations concernant les SHELL sont disponibles sur
# la toile ..... n'hesitez pas a taper vos requetes sur des moteurs
# de recherche
#
#########################################################################
#
#Le but de ce cours est :
#
#          1. de vous montrer l'etendue des possibilites offerte
#             par un shell (les lignes de commandes en sequences).
#          2. de vous initier a la programmation.
#
#Un shell est un interpréteur de lignes de commandes qui sert a
#dialoguer avec le système (DOS= fichier .bat; UNIX= sh).
#Il en existe plusieurs.
#Le plus commun et le plus portable est certainement le bash.
#
#- OUVRIR UN TERMINAL.
# cd
# cd MASTER   (commenter)
#- TAPER ps (voir csh).
#- TAPER bash.
#- TAPER ps (voir bash).
#- TAPER exit
#- TAPER ps (ne plus voir bash)

# Initier a $$ (PID de l'executable courant)

#Maintenant on ecrit un programme qui en etant executer fera tout
#une sequence de commande.
#
#
#########################################################################
# Comprendre un shell 01
#########################################################################

#nedit test01.sh

# preciser le \#!bin/bash
#PHASE 1
#                   ls
#                   ls >> old.txt
#                   mkdir REPERTOIRE_TEST
#                            ls
```

```
#                   ls >> new.txt
#                           rmdir REPERTOIRE_TEST
#
#executer ce shell :

#sh test01.sh
#
#ou
#
#ls test01.sh
#chmod +x test01.sh
#ls test01.sh
#./test01.sh
#
################## FAIRE cat de new.txt et old.txt
#
#PHASE 2
#                           pwd
#                           mkdir REPERTOIRE_TEST
#                           cd REPERTOIRE_TEST
#                           pwd
#                           cd ..
#                           rmdir REPERTOIRE_TEST
#                           pwd
#executer ce shell :
#
#sh test01.sh

#########################################################################
# Comprendre un shell 02
#########################################################################

#nedit test02.sh

# preciser le \#!bin/bash
#PHASE 1
#                           ls
#                           touch fichier.txt
#                           ls
#
#executer ce shell :

#sh test02.sh
#
#ou
#
#ls test02.sh
#chmod +x test02.sh
#ls test02.sh
#./test02.sh
#
#
```

```
#PHASE 2
#                       rm fichier.txt
#                       ls
#
#executer ce shell :
#
#sh test02.sh


#######################################################################
# Afficher des infos
#######################################################################
#echo -n je suis un etudiant
#echo je suis un etudiant
#echo -n je suis un etudiant

#######################################################################
# Test elementaire sur une variable et chaine de caracteres
#######################################################################
#
#a=' variable dans a'
#b='ls'
#
#echo a
#echo $a
#echo a$a

#######################################################################
# Executer une operation et rentrer la sortie dans shell ``
#######################################################################
#nom_utilisateur=`whoami`
#echo `$nom_utilisateur

#$b
#echo $b
#`echo $b$`
#echo `$b`
#echo `$b -a`
#echo ls
#echo `ls`
#
#echo "Le # ici ne commence pas un commentaire."
#echo 'Le # ici ne commence pas un commentaire.'
#echo Le \# ici ne commence pas un commentaire.
#echo Le # ici commence un commentaire.
#
#echo "a$a"
#echo '$a'
#echo \$a
#
######################### APPLICATION  ############################
#
```

```
#                       rep=REPERTOIRE_TEST
#                          ou='je suis dans le repertoire'
#                          echo Etape 1
#                   echo $ou `pwd`
#                   mkdir $rep
#                          echo Etape 2
#                   cd $rep
#                          echo $ou `pwd`
#                          echo Etape 3
#                   cd ..
#                   rmdir $rep
#                   echo $ou `pwd`
#
#########################################################################
# Variable deja code
#########################################################################

#echo $USER
#echo $HOME
#echo $TERM
#echo $PATH

#voir toutes les commandes
#env


# ICI INTEGRE LA CREATION D'UN ~/BIN ET LE CHANGEMENT DE $PATH ET
# ENFIN LA CREATION D'UN .BASHRC

#echo $PATH
#export PATH=$PATH:/users/user11/E05-06/l3step/etudiant/bin
#echo $PATH
# DANS ~/bin
#touch fichier_exe_01.sh
####  !/bin/bash
####  echo 'bon test'


#########################################################################
# valeurs numerique et calcul
#########################################################################
# methode 1
#i=2
#echo $i
#
#expr 3 + 9
#expr 5 \* 3
#expr 5 % 3
#
#i=`expr 3 \* $i`
#
#echo $i
```

```
#
#####    Autre methode
#
#i=2
#j=$((1+($i+1)**3))
#echo j = $j
#i=$((1+($i+5)**2))
#echo i = $i
#
#
#####    Autre methode
#
#let "k=i-2*3"
#echo k = $k
#
#j=9
#echo $j
#let j=j+1
#echo $j
#let j+=1
#echo j = $j
#
#
################################################################################
# Operateurs
################################################################################
#
#+
#    plus
#-
#    moins
#*
#    multiplication
#/
#    division
#**
#    exponentielle
#%
#    modulo, ou mod (renvoie le reste de la division d'un entier)
#+=
#    << plus-égal >> (incrémente une variable par une constante)
#-=
#    << moins-égal >> (décrémente une variable par une constante)
#*=
#    << multiplication-égal >> (multiplie une variable par une constante)
#/=
#    << division-égal >> (divise une variable par une constante)
#%=
#    << modulo-égal >> (reste de la division de la variable avec une constante)
#
################################################################################
# Test if
```

```
#########################################################################
#
#  SIMPLE sur valeurs arithmetiques
#
#a=4
#b=5
#if [ "$a" -gt "$b" ]
#then
#echo "a > b"
#else
#echo "a < b"
#fi
#
#-eq
#    est égal à
#    if [ "$a" -eq "$b" ]
#-ne
#    n'est pas égal à
#    if [ "$a" -ne "$b" ]
#-gt
#    est plus grand que
#    if ["$a" -gt "$b" ]
#-ge
#    est plus grand ou égal à
#    if [ "$a" -ge "$b" ]
#-lt
#    est plus petit que
#    if [ "$a" -lt "$b" ]
#-le
#    est plus petit ou égal à
#    if [ "$a" -le "$b" ]
#<
#    est plus petit que (à l'intérieur de parenthèses doubles)
#    (("$a" < "$b"))
#<=
#    est plus petit ou égal à (à l'intérieur de parenthèses doubles)
#
#    (("$a" <= "$b"))
#>
#    est plus grand que (à l'intérieur de parenthèses doubles)
#
#    (("$a" > "$b"))
#>=
#    est plus grand ou égal à (à l'intérieur de parenthèses doubles)
#    (("$a" >= "$b"))
#
#comparaison de chaînes de caractères
#
#=
#    est égal à
#
#
```

```
####################
# Introduction de elif
#
#if [ condition1 ]
#then
#   commande1
#   commande2
#   commande3
#elif [ condition2 ]
## Idem que else if
#then
#   commande4
#   commande5
#else
#   commande_par_defaut
#fi
#
####################
# test simple
#echo $(( 5 > 4 ))
#echo $(( $a > $b ))
#
####################
# test sur fichier
#
#if [ -f mon_fichier ]; then
#    echo "il existe"
#    else
#    echo "il n'existe pas"
#fi
#
#touch mon_fichier
#
#if [ -f mon_fichier ]; then
#    echo "il existe"
#        else
#    echo "il n'existe pas"
#fi

#-f teste l'existence d'un fichier
#-g teste l'existence d'un fichier le l'existence du GID bit
#-k idem mais avec le sticky bit
#-u idem mais avec l'UID bit
#-d teste l'existence d'un repertoire
#-x teste si le fichier existe et est executable
#-r teste si le fichier existe et est ouvert en lecture
#-w teste si le fichier existe et ouvert en ecriture
#-s teste si le fichier existe et a une taille superieur a 0 octet
#-L teste si le fichier existe et est un lien symbolique

#
#a=24
```

```
#b=47
#
##if [ "$a" -eq 245 ] && [ "$b" -eq 47 ]
##ou
#if [ "$a" -eq 245 ] && [ "$b" -eq 47 ]
#then
#  echo "Le test #1 a reussi."
#else
#  echo "Le test #1 a echoue."
#fi
#
#if [ "$a" -eq 98 ] || [ "$b" -eq 47 ]
#then
#  echo "Le test #2 a reussi."
#else
#  echo "Le test #2 a echoue."
#fi
#
#####################################################################
# While et Do
#####################################################################
#i=2
#while [ $i -le 4 ] ; do
#    echo -e "2 * $i \t = `expr 2 \* $i`"
#    i=`expr $i + 1`
#done

#
# Nouveau test en rentrant des lignes de commandes
#
# Lancer le programme en faisant "sh nom_fichier a b c" ou
# a b c sont des nombres
#i=$2
#while [ $i -le $3 ] ; do
#    echo -e "$1 * $i \t = `expr $1 \* $i`"
#    i=`expr $i + 1`
#done

#
#echo
#SECONDES=0
#LIMITE_TEMPS=10
#echo Limite de temps = "$LIMITE_TEMPS"
#INTERVALLE=1
#
#echo
#echo "Tapez sur Control-C pour sortir avant $LIMITE_TEMPS secondes."
#echo
#
#while [ "$SECONDES" -lt "$LIMITE_TEMPS" ]
#do
#
```

```
#let "SECONDES+=1"
#
#if [ "$SECONDES" -le 1 ]
#  then
#    unites=seconde
#  else
#    unites=secondes
#fi
#
#echo "Ce script tourne depuis $SECONDES $unites."
##  Sur une machine lente, le script peut laisser echapper un element du
##+ comptage quelque fois dans la boucle while.
#sleep $INTERVALLE
#echo "La variable \$SECONDS fait la meme chose : $SECONDS"
#
#done
#
#echo -e "\a"  # Beep!
#
################################################################################
#############
# Until
################################################################################
#############
#until [ "$var1" = fin ] # Condition du test ici, en haut de la boucle.
#do
#  echo "Variable d'entree $var1 "
#  echo "(Taper fin pour sortir)"
#  read var1
#  echo "variable #1 = $var1"
#done
#
#################### separateurs ; ############################
#
#echo le separateur de commande est .....; echo "le ;  bien sur"
#
#################### boucle for  ############################
# ----->EXEMPLE 01
#
#for planet in Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune Pluto
#do
#  echo $planet  # Each planet on a separate line.
#done
#
#----->EXEMPLE 02
#
#echo
#ls
#echo
#ls
#echo
#touch f1 f2 f3 g1 g2 g3
```

```
#ls
#sleep 1
#for file in [fg]*
#do
#  rm -f $file    # Removes only files beginning with "f" or "g" in $PWD.
#  echo "Removed file \"$file\"".
#sleep 1
#done
#
#echo
#ls
#echo
#
#
#----->EXEMPLE 03
#
#PASSWORD_FILE=/etc/passwd
#n=1           # User number
#
#for name in $(awk 'BEGIN{FS=":"}{print $1}' < "$PASSWORD_FILE" )
## Field separator = :    ^^^^^^
## Print first field             ^^^^^^^^
## Get input from password file              ^^^^^^^^^^^^^^^^^^
#do
#  echo "USER #$n = $name"
#  let "n += 1"
#done
#
#
#####################################################################
# Case et ;;
#####################################################################
#variable=abc
#case "$variable" in
#abc)  echo "$variable = les trois premieres lettres de l'alphabet" ;;
#xyz)  echo "$variable = les trois dernieres lettres de l'alphabet" ;;
#esac
#
#i=1
#for  (( i=0 ; i<4 ; i++ ))
#do
#variable=$i
#case "$variable" in
#1)  echo "i = 1" ;;
#2)  echo "i = 2" ;;
#3)  echo "i = 3" ;;
#esac
#done

#for  i in 1 2 3
#do
#variable=$i
```

```
#case "$variable" in
#1)  echo "i = 1" ;;
#2)  echo "i = 2" ;;
#3)  echo "i = 3" ;;
#esac
#done

#####################################################################
# Exemple de fonction
#####################################################################
#
#function hereis {
#  for name in "$@"
#    do
#    cat <<MSG
#        This is an example of an HERE IS FILE.
#        One argument is ${name}.
#        The date is `date`.
#MSG
# done
#}
#
#hereis  shellscript.html
```