Dennis Tikhomirov **DevOps training, Spring’19 Homework**

Task 3. Bashshell

1 «if else if»

1.1 Create and execute a file with a procedure that compares the parameter

passed to it with a certain set of characters (password)
I’ve created script for validation login/password
https://github.com/dennis00010011b/epam-devops-training/blob/master/Task3/task31.sh

```
#! bin/bash
# This script provides simple login validation using if and case statements
# I know the realization could be more simpler with for-each loop
incorrect_pass () {
echo "Incorrect password"
exit
}
incorrect_name () {
echo "Incorrect login name"
exit
}
declare -A logins
logins[0]='user1'
logins[1]='user2'
logins[2]='user3'
declare -A passwords
passwords[0]='pass1'
passwords[1]='pass2'
passwords[2]='pass3'

echo "Enter login:"
read login
if [ $login = 'exit' ];
then
exit
elif [ $login = ${logins[0]} ] || [ $login = ${logins[1]} ] || [ $login = ${logins[2]} ];
then
echo "Enter password:"
read passwd
else
incorrect_name
fi
case $login in
  ${logins[0]} )
   if [ $passwd != ${passwords[0]} ];
   then
   incorrect_pass
   fi
  ;;
  ${logins[1]} )
   if [ $passwd != ${passwords[1]} ];
   then
   incorrect_pass
   fi
  ;;
  ${logins[2]} )
   if [ $passwd != ${passwords[2]} ];
   then
   incorrect_pass
   fi
   ;;
   *)
     incorrect_name
   ;;
esac
echo "SUCCESS! You logged in."
```

1.2 It is necessary to organize the branching of the computational process in the procedure depending on the value of the variable being transferred (at least 3 branches) (for example: x> 10, x = 10, x <10)

```
#! bin/bash
echo “Enter digit”
read  x
if       [ $x -gt $1];
  then echo “Value more than $1”;
  elif [ $x -lt $1 ];
    then echo “Value less than $1”;
    else echo “Value equals $1”;
fi
```

2 «for, while..»

2.1 It is required to develop a script that implements output to the screen from the file name specified by the parameter of the subdirectory with the indication of their type.

```
#! bin/bash
while IFS= read -r line
  do
    ls -l $line
done < “$1”
```

2.2 Try to complete the task from TASK 1 (Cinderella) by means of an executable file (bash shell). Compare the results.

https://github.com/dennis00010011b/epam-devops-training/blob/master/Task3/task322.sh

```
# Script organizes files from directory specified by parameter
# to directory ./cinderella (wiil be created if doesn't exist)
#!/bin/bash
function ensure () {
if [ ! -d $1 ]
then mkdir $1
fi
}
d="cinderella"
ensure "$d"

ensure $d/video
ensure $d/audio
ensure $d/books
cp "$1"/*.pdf ./"$d"/books
cp "$1"/*.chm ./"$d"/books
cp "$1"/*.djvu ./"$d"/books
cp "$1"/*.mp3 ./"$d"/audio

ensure "$d"/video/80x
ensure "$d"/video/200x
ensure "$d"/video/latest


cp "$1"/*198* ./"$d"/video/80x
cp "$1"/*200* ./"$d"/video/200x
for i in 2014 2015 2016
do
cp "$1"/*"$i"* ./"$d"/video/latest
done
```

3 It is required to develop a script that implements a dialogue with the user in the form of a menu in the console mode. The script should be executed in a cyclic mode until the "Exit" item is selected. The first menu item should display information about the developer (full name). The following menu items should perform tasks according to the options in the table.

Variant 2

a. Delete files of the specified extension (specified in the parameter) in the specified folder (specified in the parameter) (* check the existence of paths / disks, etc.)

b. In the specified folder (specified in the parameter) find the file with the longest name

https://github.com/dennis00010011b/epam-devops-training/blob/master/Task3/task33.sh

```
#!/bin/bash
                              ################################
                              # get number of files with given extension
                              #Input: $1=path, $2=extension

                              function amount_files () {
                               AMOUNT=$(ls -1p $1/*.$2 | wc -l)
                              }
                              ################################
                              #read a path name from console


                              function read_dir() {
                               echo "Enter directory path:"
                               read DIR
                               if  [ ! -d $DIR ];
                                 then echo "Directory dosn't exist"
                               fi
                              }
                              ################################
                              # Menu

                              select VAR in "About us" "Delete files" "Find file with longest name" "Exit"
                              do
                               case $VAR in

                              ###############################
                              # Info about developer

                               "About us")
                                  echo "Developed by Dennis Tikhomirov"
                                 ;;
                              ###############################
                              # Exit from programm

                               "Exit")
                                  echo "Good bye!"
                                  exit
                               ;;
                              ###############################
                              # Delete files with given extension
```

```
 "Delete files")
    read_dir
    echo "Enter file's extension:"
    read EXT
    amount_files $DIR $EXT
    if [ $AMOUNT != "0" ];
      then
        echo "There are $AMOUNT files in directory $DIR"
        ls -p $DIR/*.$EXT
        echo "Delete them?[y/n]:"
        read CONFIRMATION
        if [ $CONFIRMATION = "y" ];
          then
            rm $DIR/*.$EXT
        fi
      else
        echo "Can't find files with extension $ext in directory $DIR"
    fi
 ;;
###############################
# Find in given directory the file/files witn longest name

 "Find file with longest name")
    read_dir
    MAX=0
    LENGTH=0
    NAMES=()


# Find max length of file's name in directory $DIR
  for F in $DIR/*.*
    do
      LENGTH=${#F}
      if [ $LENGTH -gt $MAX ];
        then
          MAX=$LENGTH
      fi
   done
   if [ $MAX -eq 0 ];
     then
       echo "Directory $DIR is empty"
     else
```

```
# Find all files with longest name
       for F in $DIR/*.*
         do
           LENGTH=${#F}
           if [ $LENGTH -eq $MAX ];
             then
               NAMES+=($F)
           fi
         done
       echo "Directory contains ${#NAMES[@]} files with longest name,
       length is $(($MAX-${#DIR}-1)) symbols"

#print array of file's names
     echo "${NAMES[@]}"
   fi
 ;;

#####################################
# Incorrect choice
 *)
   echo "Incorrect choice. Please try again"
 ;;
esac
done
```