BASH – PRÁCTICA 4 - PROGRAMACIÓN SHELL-SCRIPTS
SOLUCIÓN PROPUESTA

1. Realiza un shell-script que comprueba si el fichero pasado por parámetro tiene permisos de lectura, en cuya caso, mostrará el contenido de forma paginada.

```
#!/bin/ bash

if [ $# -eq 1 ]
then
  if [-r $1 -a -f $1 ]
  then
    more $1    # cat $1 | more
  else
    echo "El fichero no tiene permisos de lectura"
  fi
else
  echo "ERROR, No se ha pasado el fichero"
fi
```

2. Realiza un shell-script que reciba por teclado el nombre de un directorio. Si es directorio no existe, mostrará un mensaje de error, y si existe, mostrará un listado de sus archivos.

```
#!/bin/bash

echo "Escribe el nombre del directorio"
read DIRECTORIO
if [ -d $DIRECTORIO ]
then
  ls $DIRECTORIO
else
  echo "ERROR, No existe el directorio" $DIRECTORIO
fi
```

3. Implementar un shell-script que mueva todos los ficheros que se le pasen por parámetro, y para los que tengamos permiso, al directorio Backup.

```
#!/bin/bash

if [ $# -ne 0  ]
then
  if [ ! -d Backup ]
  then
    mkdir Backup
  fi
  for FICHERO in $*
  do
    if [ -f $FICHERO -a -w $FICHERO ]
    then
      mv $FICHERO Backup
    fi
  done
else
  echo "ERROR, no se han pasado ficheros"
fi
```

4. Implementa un shell-scipt que cree un fichero de nombre Copia.bkp, donde se almacenen comprimidos todos los ficheros que se pasen por parámetro.

```
#!/bin/bash

if [ $# -ne 0  ]
then
  tar -cvfz Copia.bkp $* 2> /dev/null
else
  echo "ERROR, no se han pasado ficheros"
fi
```

5. Realiza un shell-script que limpie la pantalla, muestre todos los nombres de ficheros que hay en el directorio actual (mostrando sus atributos), la fecha actual y el directorio donde nos encontramos.

```
#!/bin/bash

clear
ls -l
date
pwd
```

6. Realizar un Shell-script que reciba dos parámetros. El primero será un fichero y el segundo, un número entero N. El Shell-script mostrará las primeras N líneas del fichero. Se debe comprobar el número de parámetros y que se pueda acceder al fichero.

```
#!/bin/bash

if [ $# -eq 2 ]
then
  if [ -f $1 -a -r $1 ]
  then
    head $2 $1 2> /dev/null
  else
    echo "No podemos acceder a" $1
  fi
fi
```

7. Realizar un script con la siguiente sintaxis.

ej7.sh [opciones] [argumentos]

Al ejecutar el shell-script, este deberá mostrar el nombre del usuario que lo ejecuta. Las opciones que puede recibir el shell son las siguientes:

Sin opciones ni argumentos, deberá mostrar el contenido del propio Shell-script.

-m Comprobará que los argumentos (tras -m) son ficheros, en cuyo caso muestra su contenido por pantalla.

-x Deberá comprobar que los argumentos (tras -x) son programas ejecutables, en cuyo caso los ejecutará.

-p Mostrará los propietarios de los ficheros que se reciben por parámetro

```
#!/bin/bash

# Función que va a recibir un conjunto de paramétros.
# Si son ficheros, va a mostrar su contenido
function mostrar_ficheros ()
{
  for FICHERO in $*
  do
    if [ -f $FICHERO -a -r $FICHERO ]
    then
      echo El contenido del fichero $FICHERO es
      more $FICHERO
    fi
  done
}

# Función que va a recibir un conjunto de paramétros.
# Si son ficheros con permisos de ejecución, los ejecuta.
function ejecutar_programas ()
{
  for FICHERO in $*
  do
    if [ -f $FICHERO -a -x $FICHERO ]
    then
      ./$FICHERO
    fi
  done
}
# Función que va a recibir un conjunto de ficheros
# pasados por parámetro y nos va a mostrar el nombre
# de su propietario
function propietario ()
{
  for FICHERO in $*
  do
    ls -l $FICHERO | cut -d" " -f3
  done
}
```

```
# Programa Principal
echo "Ejecuta este script el usuario" $LOGNAME
if [ $# -eq 0 ]
then
  more $0     # Mostramos el contenido de este fichero
else
  OPCION=$1 #Opción con la que se quiere ejecutar el script
  shift # Desplazamos los parámetros una posición a la izquierda
  case $OPCION in
    "-m")
      mostrar_ficheros $*
      ;;
    "-x")
      ejecutar_programas $*
      ;;
    "-p")
      propietario $*
      ;;
    *)
      echo "Opción incorrecta"
      ;;
  esac
fi
```

8. Realizar un script que muestre el nombre del propio script, el número de parámetros que recibe, muestra una lista con los parámetros recibidos, el PID del proceso que lo lanza.

```
#!/bin/bash

echo "El nombre del script es:" $0
echo "El número de parámetros recibidos en:" $#
echo "Los parámetros recibidos son:" $*
echo "El PID del proceso es:" $$
```

9. Implementar un script que muestre el siguiente menú:

1 - Sumar
2 - Restar
3 - Dividir
4 – Multiplicar
0 - Salir

- Después de mostrar el menú, se pedirá que se elija una opción. Si la opción elegida no está entre el 1 y el 4, se mostrará un mensaje de error. En caso de que la opción sea válida, se pedirán dos números por teclado y en función de la operación elegida, se devolverá el resultado por pantalla.
- Cada operación será implementada haciendo uso de funciones.
- Si la opción elegida no es válida, se volverá a mostrar el menú.
- El programa terminará, cuando se pulse 0.

```
#!/bin/bash

function suma ()
{
  echo "La suma es" $(($1 + $2))
}

function resta ()
{
  echo "La resta es" $(($1 - $2))
}

function multiplicacion ()
{
  echo "La multiplicación es" $(($1 * $2))
}

function division ()
{
  echo "La división es" $(($1 / $2))
}
OPCION=1
while [ $OPCION -ne 0 ]
do
  clear
  echo "Programa calculadora"
  echo "1. Sumar"
  echo "2. Restar"
  echo "3. Multiplicar"
  echo "4. Dividir"
  echo "0. Salir"
  echo "Elige una opción"
  read OPCION

  if [ $OPCION -ge 0 -a $OPCION -le 4 ]
  then
    echo "Introduce el primer número de la operación"
    read NUM1
    echo "Introduce el segundo número de la operación"
    read NUM2
    case $OPCION in
      "1") suma $NUM1 $NUM2 ;;
      "2") resta $NUM1 $NUM2;;
      "3") multiplicacion $NUM1 $NUM2 ;;
      "4") division $NUM1 $NUM2;;
    esac
    read   #esperamos hasta que se pulse intro
  else
    echo "La opción elegida es incorrecta"
  fi
done
```