Práctica 3

Ejercicio 1: Imprimir en la salida estándar el mensaje hola mundo

Shebang #!

```
#!/usr/bin/awk -f
BEGIN {
  print "parámetros:"
  for (i = 1; i < ARGC; i++)
    print "  " i ": " ARGV[i]
}
```

¿Para qué sirve? ¿Qué pasa si no lo pongo?

¿Cómo comento en bash? ¿ Para qué sirve el carácter #?

Solución

```
#!/bin/bash

# Si la primer línea de mi script comienza con  la cadena #! se interpretará como el path

# al intérprete a utilizar (podría ser python, perl, php, etc...)

# Ahora el script en sí

echo "Hola mundo"
```

Ejercicio 2 - Utilizando variables que contengan el nombre de un alumno, la carrera que estudia y la facultad donde estudia imprimir un mensaje en pantalla del estilo: El alumno X estudia la carrera Y en la Facultad Z.

Uso de comillas. Un ejemplo...

```
x=100
y='500$x'
echo $y          # Mostrará.. '500$x'
y="500$x"
echo $y          # Mostrará.. '500100'
```

interpretan el string dentro de las comillas

La comilla doble (") sirve para sustituir lo que hay dentro

Luego las comillas invertidas (`) se utilizan con comandos, y se reemplazan por la salida del comando que está entre las comillas invertidas. Ejemplo..

```
usuarios=`who | wc -l`
```

Otra forma de sustituir seria: usuarios=$(who | wc -l) cumple la misma función que las comillas (`)

Solución

```
NOMBRE="Fulano De Tal"

facultad=Informatica

carrera_1="Licenciatura en Sistemas"

carrera_2="Licenciatura en Informatica"

echo El alumno $NOMBRE de la Facultad de $facultad cursa $carrera_1 y $carrera_2
```

3- Renombrando Archivos: haga un script que renombre solo archivos de un directorio pasado como parámetro agregándole una CADENA, contemplando las opciones:
a) "-a CADENA": renombra el fichero concatenando CADENA al final del nombre del archivo.
b) "-b CADENA": renombra el fichero concatenado CADENA al principio del nombre del archivo.

¿Cómo pasamos argumentos a nuestro script?

```
./nombre_script  arg1 arg2 argN
```

¿Como sabemos cuántos argumentos pasaron?

La variable $# almacena el número de argumentos o parámetros recibidos por el script o función

¿Cómo obtenemos TODOS los argumentos?

La variable $* nos devuelve todos los argumentos recibidos por el script o función

```
#!/bin/bash

#Recibe parámetros -a CADENA o -b CADENA
#$1 nombre del directorio, $2 es -a รณ -b, $3 es CADENA
#Ejemplo de Ejecución bash -x ejercicio14RenombraArchivos
/root/scripts/ejercicio14 -b zzzzzz
if [ $# -ne 3 ]
then
    echo "La Cantidad de parametros es incorrecta"
    exit 1
fi
if [ ! -d $1 ]
then
    echo "El primer parametro no es un directorio existente"
    exit 2
fi
if [ -z $3 ]
then
    echo "El tercer parametro es nulo"
    exit 3
fi
```

```
case $2 in
    "-a")
            for i in `ls $1`; do
                mv $1/$i $1/$i$3
            done
    ;;
    "-b")
            for i in `ls $1`; do
                mv $1/$i $1/$3$i
            done
    ;;
    *)
            echo "El segundo parametro debe ser -a o -b"
            exit 4
esac
```

4- Crear un script que verifique cada 10 segundos si un usuario se ha logueado en el sistema (el nombre del usuario será pasado por parámetro). Cuando el usuario finalmente se loguee, el programa deberá mostrar el mensaje ”Usuario XXX logueado en el sistema” y salir. (ejercicio18UsuarioLogueado.sh)

```
#!/bin/bash
#Recibe parámetro un nombre de usuario
#Ejemplo de Ejecución ejercicio18usuario
if [ $# -ne 1 ]
then
   echo "La Cantidad de parametros es incorrecta"
   exit 1
fi

#Se podría validar contra el /etc/passwd que el usuario exista
while true; do
seLogueo=`who | grep $1 | wc -l`

if [ $seLogueo != 0 ]
then
   echo "usuario $1 logueado"
   exit 0
fi
sleep 10
done
```

5- Crear un script que presente un menú al usuario con las siguientes opciones:
a- Inicializar vector: inicializa el vector con números del 1 al 10.
b- Longitud de vector: imprime la cantidad de elementos del vector.
c- Elementos: imprime todos los elementos del vector.

¿Cómo se declara una función?

Function nombre {block} o nombre(){block}

¿Cómo se definen los vectores? ¿Cómo accedo a su contenido?

${valores[*]} o ${valores[@]} *# Muestra todos los valores de un array*

${!valores[*]} *# Muestra todos los índices de un array*

${#valores[*]} *# Devuelve el número de valores en un array*

${#valores[0]} *# Devuelve la longitud del indice 0*

```
#!/bin/bash

select OPCION in opcion_1 opcion_2 opcion_3
  do
  if [ $OPCION ]; then
      echo "Opcion elegida: $OPCION"
      break
  else
      echo "Opcion no valida"
  fi
done
```

Permite seleccionar al usuario una opción de una lista de opciones en un menú.

```
vector=();

function inicializar()
{
vector=(1 2 3 4 5 6 7 8 9 10);
echo "vector inicializado con éxito";
}

function longitud()
{
echo "la longitud del vector es ${#vector[*]}";
}

function elementos()
{
    echo "los elementos son ${vector[@]}";

}
```

```
function salir()
{
exit;
}


select opcion in inicializar longitud elementos salir
do
    $opcion
done
```