Para un usuario cualquiera, cuyo nombre se introducirá por teclado, buscar los archivos de texto (.txt) que se encuentren en su directorio. Mostrar el contenido de aquellos cuyo tamaño sea menor de 100 MB. No olvidéis comprobar si existe la carpeta del usuario antes de intentar buscar.

Escribir un script que acepte 2 parametros: ruta (directorio) y nombre de usuario. El mismo deberá contar y mostrar la cantidad de archivos en la ruta especificada que pertenecen al usuario pasado por parametro. Validar cantidad de parametros: 2 El primer paramertro debe existir y ser un directorio

Escribir un script que muestre por pantalla el login, UID y home de usuario de una lista de usuarios pasados como parámetros. Si el usuario no existe mostrar "El usuario $1 no existe".

Versión 1

```
#!/bin/bash
while [ $# -gt 0 ]
  do
    usuario=`cat /etc/passwd | grep "^$1"`
    if [ "$usuario" == "" ]
     then
       echo "El usuario " $1 " no existe"
    else
       echo -n  "Usuario :"
       echo $usuario | cut -f1 -d":"
       echo -n "UID :"
       echo $usuario | cut -f3 -d":"
       echo -n "Home :"
       echo -n  $usuario | cut -f6 -d":"
    fi
    shift
  done
```

Versión 2

```
#!/bin/bash
for usuario in $*
  do
    id  $usuario > /dev/null
    if [ $? -eq 0 ]
     then
       echo -n  "Usuario :"
       echo $usuario | cut -f1 -d":"
       echo -n "UID :"
       echo $usuario | cut -f3 -d":"
       echo -n "Home :"
       echo -n  $usuario | cut -f6 -d":"
    fi
    shift
  done
```

Escribir un script que busque un proceso cuyo PID es pasado como parámetro, si el proceso existe termine su ejecución en caso contrario muestre por pantalla "El proceso con el PID $1 ingresado no existe".

Versión 1

```
#!/bin/bash
ps $1
if [ $? -eq 0 ]
  then
    kill -9 $1
else
   echo "El proceso con el PID $1 no existe"
fi
```

Versión 2

```
#!/bin/bash
pids=`ps -A -o pid`
for pid in $pids
  do
    if [ "$pid" == "$1" ]
      then
        kill -9 $1
        exit
    fi
  done
echo "El proceso con el PID $1 no existe"
```

5-Describir la funcionalidad del siguiente script:

```
#!/bin/bash
if [ $UID -eq 0 ]; then
   ps $1  > /dev/null
   if [ $? -eq 0 ]; then
            kill -9 $1
   else
      echo "El proceso con el PID $1 no existe"
   fi
else
   echo  "Usted no es un administrador"
fi
```

Describir la funcionalidad del siguiente script:

```
#!/bin/bash
pids=`ps -A -o pid`
for pid in $pids
  do
    if [ "$pid" == "$1" ]
      then
        kill -9 $1
        exit
    fi
  done
echo "El proceso con el PID $1 no existe"
```

2-Describir la funcionalidad del siguiente script:

```
#!/bin/bash
while [ true ]
do
      hora=$(date +%H)
      minuto=$(date +%M)
      if [ $hora == "22" ] && [ $minuto == "30" ]; then
           tar cvfz ~/$hora$minuto.tar.gz  /etc/init.d && echo "Respaldo se genero correctamente"
      fi
      sleep 60
done
```

Describir la funcionalidad del siguiente script:

```
#!/bin/bash
while [ $# -gt 0 ]
do
     usuario=$(cat /etc/passwd | grep "^$1")
     if [ "$usuario" == "" ]
     then
         echo $1 " no existe"
     else
```

```
        echo -n "UID:"
        echo $usuario | cut -f3 -d":"
        echo -n  "LOGNAME :"
        echo $usuario | cut -f1 -d":"
        echo -n "PWD:"
        echo -n  $usuario | cut -f6 -d":"
        echo -n "SHELL:"
        echo -n  $usuario | cut -f7 -d":"
    fi
    echo
    shift
done
```

Codificar un script que acepte como parámetro únicamente un archivo regular, verifique la existencia del mismo y muestre por pantalla un menú que permita, sobre este, ejecutar las siguientes operaciones:

1) Mostrar su tamaño en MB
2) Mostrar su tipo de archivo
3) Respaldarlo (comprimir) en su ~
4) Crear en su ~ un enlace simbólico que apunte a este
5) Fin del programa (Cierra el menú)

Ante cualquier otra opción muestre el mensaje “Opción no válida”.

Describir la funcionalidad del siguiente script:

```
#!/bin/bash
clear
if [ $# -eq 2 ]
then
        case $1 in
                "redes" | "algebra" | "programacion") ;;
                *) echo "La materia no existe"
                    exit 1
                    ;;
        esac
        case $2 in
                "lu") grep -i $1 alumnos.csv | cut -f 1,2,3,7 -d"," | sort -k 1  -t ","
                      ;;
                "nota") grep -i $1  alumnos.csv | cut -f 1,2,3,7 -d"," | sort -k 4 -t "," -n
                        ;;
                "apellido") grep -i $1  alumnos.csv | cut -f 1,2,3,7 -d"," | sort -k 2 -t ","
                            ;;
                *) echo "Criterio de ordenacion incorrecto"
                  ;;
        esac
else
        echo "Se deben enviar 2 parametros"
fi
```