Trabajo Práctico N°1:
Parte B:

Sobre su máquina virtual o real que instaló desarrolle lo siguiente:

IMPORTANTE, la práctica se resolvió en una terminal en *INGLÉS*
NO usaremos tilde, para evitarme escribir una tecla extra

1_ En su home (Perfil de Usuario) crear un directorio llamado **prácticas**

```
mkdir practicas
```

2_ Dentro de **prácticas** crear un directorio **tp1**

```
mkdir practicas/tp1
# o
mkdir tp1
```

3_ Dentro de **tp1** crear el archivo **perfil.txt**

```
touch practicas/tp1/perfil.txt
# o
touch perfil.txt
```

4_ Usando comandos de Shell, crear un usuario nombre "admi", para luego darle accesos privilegiados

```
useradd admi
```

5_ En **tp1** realizar una copia de los archivos **/etc/passwd** y **/etc/group**
y guardarlos como **usuarios** y **grupos** respectivamente

```
cp /etc/passwd usuarios

cp /etc/group grupos
```

6_ En el archivo **usuarios** creado en el punto anterior, identificar el **uid** (Identificador de usuario),
gid (Identificador de grupo), y **shell** (Interprete de comandos) y guárdalos en el archivo **perfil.txt** creado en el punto 3

```
cat usuarios | grep -w "admi" | cut -d ":" -f3,4,7 >> perfil.txt
#o
cut usuarios -d ":" -f3,4,7 >> perfil.txt
```

7_ Cambiar la fecha de acceso de **perfil.txt** a 2024-03-06

```
touch -a -t 202403060000 perfil.txt                    #  Completamos la hora y los minutos con cero
```

8_ Muestra las primeras 2 líneas del archivo **usuarios**

```
head -2 usuarios
```

9_ Muestre las últimas 4 líneas del archivo **grupos**

```
tail -4 usuarios
```

10_ Mostrar el contenido del archivo **/etc/fstab** y redireccionar su contenido al archivo **filesystem.txt**

```
cat /etc/fstab > filesystem.txt
```

11_ Indicar la cantidad de líneas, caracteres y palabras que contiene el archivo **filesystem.txt**,
redireccionar las salidas al archivo **cantidad**

```
wc -l - m - w filesystem.txt > cantidad
#o
echo "La cantidad de lineas      = $(wc -l < filesystem.txt)" > cantidad
echo "La cantidad de caracteres = $(wc -m < filesystem.txt)" >> cantidad
echo "La cantidad de palabras    = $(wc -w < filesystem.txt)" >> cantidad
```

12_ Redireccionar **history** a **/documentos/practicas/tp1/cmd.txt**

```
history > cmd.txt
```

13_ Renombrar el archivo **cmd.txt** como **hist.txt**

```
mv cmd.txt hist.txt
```

14_ Usando el comando **grep** buscar en el archivo **usuarios** el registro correspondiente al usuario creado anteriormente y mandar al archivo **newusuario**

```
cat usuarios | grep "admi:x" > newusuario
# o
grep "admi:x" usuarios > newusuario
# o
grep "admi" usuarios > newusuario
```

15_ Copiar el directorio **tp1** como **tp1-temp**

```
cp -r tp1 tp1-temp
```

16_ Copiar el documento desarrollado en la **PARTE_A** dentro de la carpeta **practicas/tp1**, con el nombre **Practica1ParteA.txt**

```
cp /documentos/practicas/tp1/PARTE_A/documento.txt /documentos/practicas/tp1/Practica1ParteA.txt
```

17_ Comprimir y empaquetar el directorio **tp1** y dejar dicho archivo en el directorio practicas

```
tar -czvf practicas/tp1.tar.gz tp1/
```

18_ Borrar el directorio **tp1-temp**

```
rm -r tp1-temp
```

Trabajo Práctico N°2:

1_ Crear el directorio **tp2** dentro de **/documentos/practicas** y realizar un script que muestre por pantalla
"Estoy aprendiendo lenguaje scripting"

```
#!/bin/bash
echo "Estoy aprendiendo lenguaje scripting"
```

2_ Hacer un script que realice los siguientes pasos:
- Limpiar la pantalla
- Ejecutar el comando **df**
- Buscar en la salida del comando anterior el espacio disponible en el sistema de archivos raíz
- Mostrar un mensaje por pantalla que diga:
  El espacio disponible en el sistema de archivos raíz es : xxxxxx

```
#!/bin/bash
clear
var=$(df -h | grep -w "/" | awk '{print $4}')
echo "El espacio disponible en el sistema de archivos raíz es: $var"
```

3_ Realizar un script que evalúe 2 **archivos** ordinarios, e indique entre ambos cuál posee más cantidad de líneas.

```
#!/bin/bash

if [ ! -f uno ]; then
    echo "uno no es un archivo"
    exit
fi

if [ ! -f dos]; then
    echo "dos no es un archivo"
fi

L_arch1=$(wc -l < uno)
L_arch2=$(wc -l < dos)

if [ $L_arch1 -gt $L_arch2 ]; then          #  -gt = Mayor
    echo "uno tiene mas lineas que dos"
elif [ $L_arch1 -lt $L_arch2 ]; then        #  -lt  = Menor
    echo "dos tiene mas lineas que uno"
else
    echo "Ambos tienen la misma cantidad de lineas
fi
```

4_ Idem anterior pero pasando los archivos por **parámetro**.

```
#!/bin/bash

if [ $# -ne 2]; then

	echo “La cantidad de parametros pasados fue de: $#”
	echo “Debes pasar asi: $0 archivo1 archivo2”
	exit

fi
archivo1=$1
archivo2=$2

if [ ! -f $archivo1 ]; then

	echo “El archivo $archivo1 no es un archivo ordinario”
	exit

fi

if [ ! -f $archivo2 ]; then

	echo “El archivo $archivo2 no es un archivo ordinario”
	exit

fi

l_archivo1=$( wc -l < $archivo1)
l_archivo2=$( wc -l < $archivo2)


if [ “$l_archivo1” -gt “$-archivo2”]; then

	echo “El archivo $archivo1 tiene mas lineas ($l_archivo1) que el $archivo2 ($l_archivo2)

elif [ “$l_archivo1” -lt “$-archivo2”]; then

	echo “El archivo $archivo2 tiene mas lineas ($l_archivo2) que el $archivo1 ($l_archivo1)

else

	echo “Ambos archivos tienen la misma cantidad de lineas ($l_archivo1)”

fi
```

5_ Hacer un script que indique la cantidad de archivos y directorios que hay en un directorio cualquiera pasado como argumento, usando **estructuras repetitivas.**

```
#!/bin/bash

if [ $# -eq 0]; then
	echo “Debes pasar un parametro”

elfi [ $# -ne 1]; then
	echo “Debes ingresar solamente un parametro”
	exit
fi


if [ ! -d $1 ]; then

	echo “No me pasaste como parametro un directorio, $1”
	exit
fi

arch=0
dir=0

for i in ./*; do

	if [ -f $i ] ; then

		let arch++

	fi

	if [ -d $i ]; then

		let dir++

	fi

done

echo “La cantidad de archivos es: $arch”
echo “La cantidad de directorios es: $dir”
```

6_ Hacer un scripts que realice los siguientes pasos:

- **Descomprimir** el directorio tp1 de la práctica 1 en un directorio llamado nuevo
- Del archivo **hist.txt** sacar las líneas que corresponden al comando ls y guardarlas en el archivo "**listado**"
- **Comprimir** el archivo "**listado**"

```
#!/bin/bash

cat nuevo/tp1/hist.txt | grep -w "ls" > listado


tar -cf comprimido.tar listado
```

7_ Hacer un scripts que **recorra** el directorio **tp1** copiado anteriormente y **busque** todos los archivos con extensión .txt y le asigna **permiso** de escritura a otros

```
# Para este script debemos estar parado en tp2, y en él se descomprimir el tp1 en una carpeta (nuevo)

#!/bin/bash
 for i in nuevo/tp1/*.txt; do

      chmod o+w $i
      ls -l $i

done
```

8_ Implementar con un script el punto 6 de la parte B de la práctica 1

```
#  Como estoy en tp2  no tengo el archivo "usuarios", entonces lo tengo que crear con el contenido de passwd"
cp /etc/paswd usuarios

#  Algo así queda: "1000:1000:/bin/bash"
cat usuarios | grep -w "admi:" | cut -d ":" -f3,4,7 > perfil.txt
```

9_ Implementar un script usando la sentencia **while**, en donde se lea el archivo passwd línea por línea y se imprima el mismo por Terminal.

```
while read -r line; do

      echo "$line"
done<etc/passwd
```

10_ Copiar el archivo usuarios generado en el punto 5 parte B del tp1 y utilizarlo en un script que permita simular dar de alta un usuario en términos de:

usu01:x:uid:guid: :/home/usu01:/bin/bash (línea que debe agregar al archivo usuarios)

El usuario **usu01** debe ser pasado como argumento

```
#!/bin/bash

#
if [ -z $1 ]; then
	echo “Debes ingresar un nombre para el nuevo usuario”
	exit
fi


uid=$(tail -1 usuarios | cut -d “:” -f3)
gid=$(tail -1 usuarios | cut -d “:” -f4)

let uid++
let gid++

echo “$1:x:$uid:$gid:$1:/home/$1:/bin/bash” >> usuarios
```

Trabajo Práctico N°3:

1_ Guardar las variables de entorno **HOSTNAME**, **HOME**, **LOGNAME** Y **PATH** en un archivo de nombre **var-set** ordenado por nombre de variable.

```
echo "HOME: $HOME"  > $var
echo "HOSTNAME: $HOSTNAME"  >> $var
echo "LONGNAME: $LONGNAME" >> $var
echo "PATH: $PATH" >> $var
```

2_ Guardar las variables de entorno **HOSTNAME**, **HOME**, **LOGNAME** Y **PATH** en un archivo de nombre **var-set**

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Debes pasar un solo parametro, el directorio tp2 que esta en practicas"
    exit
fi

if [ ! -d $1 ]; then
    echo "Debes pasar un directorio, no lo hiciste"
    exit
fi

for i in $1/*.sh; do
    echo "$i"
    chmod g+w,o+x $i                    #  Tiene que ir "coma", además NO debe tener espacio (después de la coma)
                                #  Me refiero a cuando queremos asignar a 2 diferentes
done
```

3_ Recorrer el directorio .../practicas/tp1/ y encontrar los archivos ordinarios que tengan permiso igual a 644. De dichos archivos guardar solo los nombres en un archivo de nombre “perm644”.

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Solo debes pasar un directorio: practicas/tp1"
    exit
fi

echo "Me pasaste: $1"
var=$(basename $1)
echo "El nombre del directorio: $var"
if [ "$var" != "tp1" ]; then
    echo "No me pasaste el directorio tp1"
    exit
fi

if [ ! -d $1 ]; then
    echo "No pasaste un directorio, mal muy mal"
    exit
fi

rm -f perm644

for i in $1/*; do

    if [ -f $i ]; then
        var=$(stat -c %a $i)
        if [ $var -eq 644 ]; then
            echo "$(basename $i)" >> perm644
        fi
    fi
done
```

4_ Usando el comando find, generar un scrip que al pasarle un directorio cualquiera pasado como parámetro:
a) Guarde en el archivo name los nombres de archivos que comienzan con la letra a.
b) Guarde en el archivo extend los nombres de archivos que tienen extensión .txt
c) Guarde en el archivo perm los nombre de archivos que tienen permisos iguales a 644
d) Guarde en el archivo tam los nombres de archivos que tienen tamaño mayor a 1K

```
#!/bin/bash

	if [ $# -ne 1 ]; then
		echo "Debes pasar un parametro, uno solo"
		exit
	fi

	if [ ! -d $1 ]; then
		echo "No pasaste como aprametro un directorio"
		exit
	fi

rm -f name
rm -f extend
rm -f perm
rm -f tam

var="$1"

find $var -type f -name "a*" -exec echo "{}" >> name \;
find $var -type f -name "*.txt" -exec echo "{}" >> extend \;
find $var -type f -perm 644 -exec echo "{}" >> perm \;
find $var -type f -size +1k -exec echo "{}" >> tam \;


echo -e "\nNombres con a*" ; cat name
echo -e "\nExtensiones .txt" ; cat extend
echo -e "\nArchivos con permisos 644" ; cat perm
echo -e "\nArchivos con menos de 1k" ; cat tam
```

5_ Usando una estructura repetitiva recorrer un directorio cualquiera pasado por parámetro y determinar que archivos fueron modificados (Comando stat) en un mes determinado.
Dicho mes también pasarlo por parámetro.

```
#!/bin/bash

if [ $# -ne 2 ]; then
	echo "Es necesario 2 parametros"
	echo "Debes pasar: $0 directorio mes"
	exit
fi

if [ ! -d $1 ]; then
	echo "El primer parametro no es un directorio"
	echo "Debes pasar: $0 directorio mes"
	exit
fi

if [ -z $2 ]; then
	echo "El segundo parametro debe ser una cadena de caracteres"
	echo "Debes pasar: $0 directorio mes"
	exit
fi

for i in $1/*; do
	if [ -f $i ]; then
		if [ $( stat -c %y $i | cut -d '-' -f2 ) -eq $2 ]; then
			echo "El archivo $i fue modificado en el mes $2"
		fi
	fi
done
```

6_ Hacer un informe de un directorio cualquiera pasado por parámetro que indique:

a) Qué archivos han sido modificados en los últimos 30 minutos
b) Qué archivos han sido accedidos en los últimos 60 minutos.
c) Qué archivos han sido modificados en los últimos 5 días
d) Qué archivos han sido modificados hace más de 10 días

 # En la opción "d" hubo un error de tipeo, es "accedidos" y no "modificados"

```
#!/bin/bash

if [ ! -d $1 ]; then
	echo "El parametro pasado no es un directorio"
fi

echo "Archivos modificados en los ultimos 30 minutos: "
find $1 -type f -mmin -30

echo "Archivos accedidos en los ultimos 60 minutos: "
find $1 -type f -amin -60

echo "Archivos modificados en los ultimos 5 dias: "
find $1 -type f -mtime -5

echo "Archivos accedidos hace mas de 10 dias: "
find $1 -type f -atime +10
```

7_ Hacer un script que al ejecutarse pida al usuario 2 números y después presente la suma, resta, producto y división de los mismos

```
#!/bin/bash

echo “Ingresar un numero”
read -r enumerador
echo “Ingresar otro numero”
read -r denominador

suma=$(( $enumerador + $denominador))
resta=$(( $enumerador - $denominador ))
producto=$(( $enumerador * $denominador ))

echo "La suma es: $suma"
echo "La resta es: $resta"
echo "EL proudcto es: $producto"

if [ $denominador = 0 ]; then
	echo "El denonimador es igual a 0, no se puede dividir"
else
	division=$(($enumerador / $denominador))
	echo "La division es: $division"
fi
```

8) Compruebe si un directorio cualquiera pasado como argumento existe, en tal caso contabilizar la cantidad de archivos y directorios, guardar ambos contadores en un archivo.
Usar una estructura repetitiva para resolverlo.

```
#!/bin/bash
rm -f archivo_ej8

if [ $# -ne 1 ]; then
	echo "Debes pasar un parametro, uno solo"
	exit
fi

if [ ! -d $1 ]; then
	echo "No existe el diectorio"
	exit
fi
contDIR=0
contFILE=0
for i in $1/*; do

	if [ -d $i ]; then
		let contDIR++
	fi

	if [ -f $i ]; then
		let contFILE++
	fi

done

echo "La cantidad de directorios es: $contDIR" > archivo_ej8
echo "La cantidad de directorios es: $contFILE" >> archivo_ej8
cat archivo_ej8
```

9) Muestre los números naturales del 1-20

```
for i in {1..20}; do
	echo "El nro es: $i"
done
```

10) Hacer un script que visualice un menú de tres opciones, la primera borra un fichero cualquiera, la segunda
visualiza un fichero, la tercera copia un archivo al directorio actual y la cuarta sale del script.

```
#!/bin/bash
echo "Elige una opción:"
echo "1 = Borrar archivo"
echo "2 = Visualizarlo"
echo "3 = Copiar archivo al directorio actual"
echo "4 = Salir del script"

printf "\n"
read -p "Ingresa tu opción del 1 al 4: " opcion

case $opcion in
1)	rm temp
	;;
2)	cat temp
;;
3)	cp temp temp-cp
;;
4)	echo  “Salgo del Script”
;;
    *)  echo "Opción no valida”
;;
esac
```

11_ Hacer un script que pida continuamente una palabra clave, si la palabra introducida es “secreto” que nos muestre un mensaje de Bienvenida.

```
#!/bin/bash

read -s -p "Ingresar la palabra clave: " pass

while [ $pass != "secreto" ]; do

	printf "\n"
	read -s -p "Te equivocaste. Intenta nuevamente: " pass
done

printf "\n"

#  Sin el último “printf “\n” ocurre que me deja sin salto de línea
```

12_ Hacer un script que compare dos cadenas de caracteres introducidas como parámetro, previamente comprobar si el número de parámetros es correcto.
Dejar un mensaje en pantalla que diga cadenas correctas o incorrectas.

```
#!/bin/bash

if [ $# -ne 2 ]; then

	echo "Debes ingresar 2 parametros unicamente"
	exit

fi

cadena1="$1"
cadena2="$2"

if [ -z $cadena1 ]; then

	echo "En el primer parametro no ingresaste una cadena de texto"
	exit
fi
if [ -z $cadena2 ]; then

	echo "En el segundo parametro no ingresaste una cadena de texto"
	exit
fi

if [ "$cadena1" == "$cadena2" ]; then

	echo "Las cadenas de texto son iguales"

else
	echo "Las cadenas de texto son diferentes"

fi
```

13_ Hacer un script que evalúe el tamaño de 2 directorios cualesquiera pasado por parámetro y determine cuál de ellos tiene mayor tamaño.

```
#!/bin/bash

if [ $# -ne 2 ]; then
	echo "Necesitas pasar 2 parametros"
	exit
fi

if [ ! -d $1 ]; then
	echo "El primer parametro pasado no es un directorio"
	exit
fi
if [ ! -d $2 ]; then
	echo "EL segundo parametro pasado no es un directorio"
	exit
fi

tam1=$(du -sb $1 | awk '{print $1}' )
tam2=$(du -sb $2 | awk '{print $1}' )

if [ $tam1 -gt $tam2 ]; then
	echo "El directorio ($1) con un tamanio de: $tam1 supera al directorio ($2) de un tamanio de: $tam2"

elif [ $tam1 -lt $tam2 ]; then
	echo "El directorio ($2) con un tamanio de: $tam2 supera al directorio ($1) de un tamanio de: $tam1"

else
	echo "no pasa nada"
	echo "Tienen el mismo tamanio"
fi
```

14_ Hacer un script que elimine una cadena de caracteres cualquiera, de un archivo ordinario pasado por parámetro. Genere un archivo nuevo como salida, y cambie los permisos a este archivo para que solo lo pueda leer y escribir el dueño. Chequear que se pase el parámetro y que el mismo sea un archivo ordinario.

```
#!/bin/bash

rm -f archivo

if [ $# -ne 1 ]; then
	echo "Debes pasar un parametro"
	exit
fi

if [ ! -f $1 ]; then
	echo "Debes pasar como parametro un archivo"
	exit
fi

printf "Ingresar una palabra a borrar: "
read -r var1

sed "s/$var1//g" $1 > archivo

sleep 1

chmod 600 archivo
```

#####
#####
El primer parcial llegó hasta acá **supuestamente :'v**
primera instancia*
#####

15_ Hacer un script que lea por parámetro el directorio tp2 (solo los ejercicios de script), a cada archivo leído cambiarle la palabra bash por sh, y guardar el archivo modificado en un directorio llamado “scripts-sh” dentro de tp3.

```
#!/bin/bash

if [ "$#" -ne 1 ];  then
	echo "Se debe pasar un parametro, uno solo"
	exit
fi

if [ ! -d "$1" ]; then
	echo "El parametro debe ser un directorio"
	exit
fi

if [ "$(basename "$1")" != tp2 ]; then
	echo "El directorio deber el directorio 'tp2'"
	exit
fi

if [ "$(cd "$1" && pwd)" != "/home/uwu/Escritorio/tp2" ]; then
	echo "El directorio tp2 ingresado por parametro es el incorrecto..."
	exit
fi

mkdir -p "script-sh"

#  Se supone que todos los ejercicios están en el directorio tp2
#  y que no hay necesidad de considerar los sub-directorios

for i in $1/*.sh; do
	aux="$(basename $i .sh)-m.sh"

	sed "s/bash/sh/g" $i > ./script-sh/${aux}

done
```

16_ Hacer un script que lea un archivo por parámetro y borre un rango de líneas del mismo.
Chequear que el archivo pasado por parámetro sea de tipo ordinario.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
	echo "se debe pasar un parametro, uno solo"
	exit
fi

if [ ! -f "$1" ]; then
	echo "el parametro pasado debe ser un archivo ordinario"
	exit
fi

read -p "Ingrese el primer valor del rango: " num1
read -p "Ingrese el segundo valor del rango: " num2

if [ "$num2" -le $(cat $1 | wc -l) ]; then
	sed "$num1,$num2 d" $1
else
	echo "El rango no esta dentro de los limites del archivo"
fi
```

17_ Hacer un script que acepte un fichero como parámetro, comprobar si este existe, en caso
que si exista convertir todas sus letras minúsculas en mayúsculas en lo que respecta al
contenido del mismo.

```
#!/bin/bash


if [ "$#" -ne 1 ]; then
	echo "Debes de pasar un solo parametro"
	exit
fi

if [ ! -f "$1" ]; then
	echo "El parametro pasado debe ser un archivo"
	exit
fi

sed "s/[a-z]/\U&/g" "$1"
```

18_ Hacer un scripts que al pasarle un directorio cualquiera que contenga archivos ordinarios, le
borre a cada uno de los archivos las líneas 2 a 5, el nuevo archivo de salida se debe llamar igual
que el original con el agregado al final del nombre “-m”, los archivos nuevos dejarlos en un
directorio de nombre “ar-modificados”

```
#!/bin/bash


if [ "$#" -ne 1 ]; then
	echo "Se debe de pasar un parametro, un unico"
	exit
fi

if [ ! -d "$1" ]; then
	echo "EL parametro pasado debe ser un directorio"
	exit
fi

mkdir -p "ar-modificados"
for i in $1/*; do
	if [ -f $i ]; then
	name=$(basename $i .sh)
	sed  "2,5d" $i > ./ar-modificados/"${name}-m.sh"
	fi
done
```

19_ Hacer un scripts que al pasarle un directorio cualquiera que contenga archivos ordinarios, le
inserte 3 espacios en blanco al principio de cada línea.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
	echo "Debes pasar un unico parametro"
	exit
fi

if [ ! -d $1 ]; then
	echo "Debe ser un directorio"
	exit
fi

for i in $1/*; do
	if [ -f $i ]; then
		sed "s/^/   /" $i # El signo $ es para el final, el ^ es el comienzo
	fi
done
```

20_ Escribir un script que copie todos los archivos ejecutables de la carpeta “tp2” a una carpeta
llamada “execu” y los no ejecutables a una carpeta llamada “no-execu”, además generar un
listado con los nombres de los archivos copiados.

```
#!/bin/bash


mkdir -p "execute"
mkdir -p "no-execute"

find ../tp2 -type f -executable -exec cp {}  ./execute/ \;
find ../tp2 -type f ! -executable -exec cp {}  ./no-execute/ \;

echo -e "\tListado de los ejecutables:\n"
ls execute
printf "\n"
echo -e "\tListado de los no ejecutables:\n"
ls "no-execute"
```

Anexo – Practica 3:

1_ Hacer un script que reciba como parámetro un archivo y un string y me informe si dicho string esta en el archivo. A su vez debe chequear la existencia del archivo y del parámetro

```
#!/bin/bash

if [ "$#" -ne 2 ]; then
	echo "Se deben de pasar 2 parametros"
	exit
fi

if [ ! -f "$1" ]; then
	echo "El primer parametro debe de ser un archivo"
	exit
fi

var="$2"
if [ -z "$var" ]; then
	echo "El segundo parametro esta vacio, debe ser un string"
fi

if [ -d "$2" ] || [ -h "$2" ]; then
	echo "El segundo parametro debe ser un archivo ordinario"
	exit
fi

if grep -q "$2" "$1"; then
	echo "El string se encuentra en el archivo"
else
	echo "EL string no se encuentra en el archivo"
fi
```

2_ Hacer un scripts que al pasarle un directorio cualquiera que contenga archivos ordinarios,muestre las 5 primeras líneas de cada archivo, las líneas mostradas de cada uno de ellos ir guardándola en un archivo nuevo.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
	echo "Se le debe de pasar un unico parametro"
	exit
fi

if [ ! -d "$1" ]; then
	echo "EL parametro pasado debe ser un directorio"
	exit
fi

if [ ! -f "archivo_nuevo" ]; then
	touch "archivo_nuevo"
fi

for i in $1/*; do

	if [ -f $i ]; then
		cat $i | head -5
		cat $i | head -5 >> archivo_nuevo
	fi
done
```

3_ Realizar un script, el cuál solicite un número y responda mostrando los 6 siguientes, los 6 números deben quedar guardados en orden inverso en el archivo num

```
#!/bin/bash

echo "Buenas usuario, ingrese un numero entero"
read -r number

for ((i=1+number; i <= 6+number; i++)); do
	echo "$i"
done

printf "\n"

if [ ! -f "num" ]; then
	touch num
fi
for ((j=6+number; j>=1+number; j--)); do
	echo "$j" >> "num"
done
```

4_ Realizar un script, donde por parámetro se ingresa una palabra, y esta se imprima 10 veces por pantalla y a su vez se guarde en un archivo con nombre word. Usar una sentencia iterativa para resolverlo, a su vez chequear que el parámetro no se pase en blanco cuando se ejecuta el script. Al finalizar comprimir el archivo word.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
	echo "se debe pasar un unico parametro"
	exit
fi

if [ -z "$1" ]; then
	echo "El parametro pasado esta en blanco"
	exit
fi

if [ -d "$1" ]; then
	echo "Pasaste un directorio eso esta muy mal"
	exit
fi

if [ ! -f word ]; then
	touch "word"
fi

for i in {1..10}; do
	echo "$1"
	echo "$1" >> word
done

tar -cf word.tar word
```

5_ Realizar un script que indique si un usuario pasado por argumento tiene como shell el bash, de lo contrario decir que tipo de shell tiene

```
#!/bin/bash

echo "Ingrese el usuario del que quiere ver su shell"
read -r usuario

var=$(cat /etc/passwd | grep "${usuario}:x:" | cut -d ":" -f7 | cut -d "/" -f3)

if [ "$var" = "bash" ]; then
	echo "El usuario tiene bash de shell"
else
	cat /etc/passwd | grep "${usuario}:x:" | cut -d ":" -f7 | cut -d "/" -f3
fi
```

6_ Realizar un script que al pasarle un archivo como parámetro nos devuelva el tamaño del mismo.
A su vez chequear que el parámetro no se pase en blanco y que el archivo exista o sea del tipo ordinario.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
	echo "solo se debe de pasar un parametro"
	exit
fi

if [ ! -f "$1" ]; then
	echo "Se debe de pasar un archivo ordinario que exista"
	exit
fi

stat -c %s "$1"
```

7_ Realizar un script utilizando la sentencia while, que lea un archivo cualquiera línea por línea y guarde cada una de ellas en un archivo llamado copia, a su vez valla mostrando cada línea por terminal con un retardo de 3 segundos.

```
#!/bin/bash
echo "Con que archivo te gustaria trabajar?"
read -r arch

if [ ! -f "$arch" ]; then
	echo "La cosa que pasaste/escribiste no es un archivo"
	exit
fi

if [ -f "copia" ]; then
	touch "copia"
fi

while  read -r line ; do
	echo "$line" >> copia
	sleep 1
	echo "$line"
done <"$arch"
```

8_ Hacer un script que lea un directorio cualquiera por parámetro y guarde el nombre de los archivos comunes solamente en solo-archi, y contabilice los mismos, dicho contador guardarlo al final del archivo solo-archi. Usar una sentencia iterativa para resolverlo y chequear que el parámetro no se pase en blanco.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
	echo "Se debe de pasar un parametro"
	exit
fi

if [ ! -d "$1" ]; then
	echo "EL parametro pasado debe ser un directorio"
	exit
fi

cont=0
for i in $1/*; do
	if [ -f $i ]; then
		name=$(basename $i)
		echo "$name" >> "solo-archi"
		let cont++
	fi
done

echo "$cont" >> "solo-archi"
```

9_ Hacer un script que guarde en el archivo perm todos los archivos que tienen permisos igual 755 y en el archivo exten todos aquellos que terminan con extensión .conf de un directorio cualquiera pasado como parámetro, a su vez chequear que el parámetro no sea pasado en blanco.

```
#!/bin/bash

#  Verifico si paso parametros o no (es decir que no haya pasado en "blanco")
if [ "$#" -ne 1 ]; then
        echo "Debe pasar un unico parametro"
        exit
fi
#  Verifico que sea un directorio (y que exista)
if [ ! -d "$1" ]; then
        echo "El parametro pasado debe ser un directorio"
        exit
fi
find $1 -type f -perm 755  > perm755
find $1 -type f -name “*.conf”  > exten
```

10_ Hacer un script que reciba un directorio cualquiera pasado por parámetro, y calcule la cantidad
de líneas de cada uno de sus archivos ordinarios (comunes), además contabilizar el total de líneas
de todos los archivos y guardarlo en el archivo TOT_GENEARAL.

```
#!/bin/bash


if [ "$#" -ne 1 ]; then
        echo "Se debe de pasar un unico parametro"
        exit
fi

if [ ! -d "$1" ]; then
        echo "Debe de ser un directorio"
        exit
fi

cont=0
lineas=0
for i in $(find $1 -type f); do
        lineas=$(cat $i | wc -l)
        (( cont+= $lineas ))
done

echo "Las lineas totales son: $cont" > TOT_GENERAL
```