En esta serie de ejercicios vamos a trabajar las condiciones

1. **Crear un shell script que si el ultimo comando ejecutado dio problema, muestre mensaje de error.**

Nota: $? Es 0 si el ultimo comando fue bien.

```
#!/bin/bash
echo "Ejecutando un comando..."
ls /directorio_que_no_existe

# Verifica si el comando anterior falló
if [ $? -ne 0 ]
then
   echo "Hubo un error al ejecutar el comando anterior."
else
   echo "El comando se ejecutó correctamente."
fi
```

2. **Crear un directorio con el nombre que le pasemos como parámetro. Si da problemas en la creación debe avisar con un mensaje.**

```
#!/bin/bash

if [ -z "$1" ]
then
   echo "No se pasó ningún nombre de directorio. Usa el script así: ./crear_directorio.sh
     nombre_directorio"
   exit 1
fi
mkdir "$1"
if [ $? -ne 0 ]
 then
   echo "Hubo un problema al intentar crear el directorio '$1'. Revisa los permisos o si ya
     existe."
 else
   echo "El directorio '$1' se creó correctamente."
     Fi
```


Manuel Fernando Gómez Hernández

3. **Realiza un script que pida dos cadenas. Lo primero debe comprobar que no están vaciías, y a continuación, hallar la mayor.**

```
#!/bin/bash

echo "Por favor, ingresa la primera cadena:"
read cadena1

if [ -z "$cadena1" ]
then
   echo "La primera cadena no puede estar vacía. Intenta de nuevo."
   exit 1
fi

echo "Por favor, ingresa la segunda cadena:"
read cadena2

if [ -z "$cadena2" ]
then
   echo "La segunda cadena no puede estar vacía. Intenta de nuevo."
   exit 1
fi

if [ "${#cadena1}" -gt "${#cadena2}" ]
then
   echo "La primera cadena es mayor: '$cadena1'"
elif [ "${#cadena1}" -lt "${#cadena2}" ]
then
   echo "La segunda cadena es mayor: '$cadena2'"
else
   echo "Ambas cadenas tienen la misma longitud."
fi
```

4. **Realiza un script que compare 3 números.**

```
#!/bin/bash

# Pedir los tres números al usuario
echo "Ingresa el primer número:"
read num1

echo "Ingresa el segundo número:"
read num2

echo "Ingresa el tercer número:"
read num3

if [ "$num1" -ge "$num2" ] && [ "$num1" -ge "$num3" ]
then
   echo "El mayor número es: $num1"
elif [ "$num2" -ge "$num1" ] && [ "$num2" -ge "$num3" ]
then
   echo "El mayor número es: $num2"
else
```

```
   echo "El mayor número es: $num3"
     fi
```

5. **Script que tienen como parámetros dos números y dice cual es mayor de los dos. Emplear test para realizar la comprobación**

```
#!/bin/bash

if [ "$#" -ne 2 ]
then
   echo "Debes proporcionar exactamente dos números como parámetros."
   echo "Ejemplo: ./comparar_numeros.sh 5 10"
   exit 1
fi
num1=$1
num2=$2

if test "$num1" -gt "$num2"
then
   echo "El mayor es: $num1"
elif test "$num1" -lt "$num2"
then
   echo "El mayor es: $num2"
else
   echo "Ambos números son iguales."
     Fi
```

6. **Lee dos números y luego un shell script que cambie el separador (variable IFS) por un parámetro que le pasemos (por ej ":")**

```
#!/bin/bash

if [ -z "$1" ]
then
   echo "Debes proporcionar un separador como parámetro. Ejemplo: ./script.sh :"
   exit 1
fi

nuevo_ifs=$1

# Pedir los dos números al usuario
echo "Ingresa el primer número:"
read num1

echo "Ingresa el segundo número:"
read num2

IFS="$nuevo_ifs"

echo "Los números separados por '$nuevo_ifs' son: $num1$nuevo_ifs$num2"

# Restaurar el valor original de IFS
     unset IFS
```

7. **Escribir dos preguntas de trivial “trivial.sh”. El programa conoce la respuesta correcta y evalúa la respuesta del usuario.**

```
#!/bin/bash

echo "Pregunta 1: ¿Cuál es el planeta más grande del sistema solar?"
echo "a) Marte"
echo "b) Júpiter"
echo "c) Saturno"
read respuesta1

# Evaluar la respuesta
if [ "$respuesta1" = "b" ]
then
   echo "¡Correcto!"
else
   echo "Incorrecto. La respuesta correcta es: b) Júpiter."
fi

echo "Pregunta 2: ¿Cuántos lados tiene un triángulo?"
echo "a) 3"
echo "b) 4"
echo "c) 5"
read respuesta2

# Evaluar la respuesta
if [ "$respuesta2" = "a" ]
then
   echo "¡Correcto!"
else
   echo "Incorrecto. La respuesta correcta es: a) 3."
fi

echo "¡Gracias por jugar al trivial!"
```

8. **Escribir un programa que dado un nombre de login de usuario determine si existe y si es así, presente su nombre de usuario, , número de usuario (UID), grupo primario y grupos secundarios si los tiene, con leyendas adecuadas.**

```
#!/bin/bash

echo "Ingresa el nombre de login del usuario:"
read usuario

# Verificar si el usuario existe en el sistema
if id "$usuario" &>/dev/null
then
   echo "Información del usuario '$usuario':"

   echo "Nombre de usuario: $usuario"

   uid=$(id -u "$usuario")
   echo "UID: $uid"

   grupo_primario=$(id -gn "$usuario")
   echo "Grupo primario: $grupo_primario"

   grupos_secundarios=$(id -Gn "$usuario" | sed "s/$grupo_primario//")
   echo "Grupos secundarios: $grupos_secundarios"
else
   echo "El usuario '$usuario' no existe en el sistema."
Fi
```

9. **Script que reciben un nombre como como parámetro, y valide que corresponde a un directorio. Después:**

```
#!/bin/bash

if [ -z "$1" ]
then
   echo "Debes proporcionar un nombre de directorio."
   exit 1
fi

directorio="$1"

# Verificar que el parámetro es un directorio
if [ ! -d "$directorio" ]
then
   echo "'$directorio' no es un directorio válido."
   exit 1
fi

echo "Procesando el directorio: $directorio"

# Mostrar los archivos que son enlaces simbólicos
echo "Enlaces simbólicos encontrados:"
find "$directorio" -type l -exec ls -l {} \;

# Mostrar los archivos que tienen enlaces hard
echo "Archivos con enlaces hard:"
find "$directorio" -type f -links +1 -exec ls -l {} \;
```

10. **Escribir un programa saludo que, según la hora escribe el saludo correspondiente y el nombre de pila de usuario sin el apellido (que iria seguido de coma si lo hay). En el archivo /etc/passwd los usuarios comienzan por el $LOGNAME y la quinta columna es el nombre de pila. Los saludos corresponden a las siguientes horas: Buenos días, de 05:00 hasta 12:59; Buenas tardes, de 13:00 hasta 19:59; Buenas noches 20:00 hasta 04:59. Ejemplo de mensaje: Buenos días, Juan.**

```
#!/bin/bash

# Obtener el nombre de pila desde /etc/passwd
nombre_completo=$(grep "^$LOGNAME" /etc/passwd | cut -d ':' -f 5)
nombre_pila=$(echo $nombre_completo | cut -d ' ' -f 1)

hora_actual=$(date +%H)

if [ "$hora_actual" -ge 5 ] && [ "$hora_actual" -lt 13 ]; then
   saludo="Buenos días"
elif [ "$hora_actual" -ge 13 ] && [ "$hora_actual" -lt 20 ]; then
   saludo="Buenas tardes"
else
   saludo="Buenas noches"
fi

    echo "$saludo, $nombre_pila."
```

11. **Escribir un script que al pasarle por argumento un archivo o directorio, devuelva el tamaño en MB. Si no recibe argumentos, sugiere al usuario cuáles son las posibles opciones para su ejecución.**

```
#!/bin/bash

if [ -z "$1" ]
then
   echo "No se pasó ningún argumento."
   echo "Usa el script así:"
   echo "./tamanio.sh archivo_o_directorio"
   exit 1
fi

# Verificar si el argumento es un archivo o directorio
if [ -e "$1" ]
then
   tamanio=$(du -sm "$1" | cut -f1)
   echo "El tamaño de '$1' es $tamanio MB."
else
   echo "'$1' no existe o no es un archivo/directorio válido."
   exit 1
     fi
```

12. **Escribir un script que al ejecutarlo como root reinicie el equipo después de 1 minuto.**

```
#!/bin/bash

# Verificar si el script se ejecuta como root
if [ "$(id -u)" -ne 0 ]; then
   echo "Este script debe ejecutarse como root."
   exit 1
fi

echo "El sistema se reiniciará en 1 minuto."

# Programar el reinicio
shutdown -r +1
```

13. **script que recibe como parámetro un directorio y devuelve una lista con los tipos de archivo en promedios numéricos que tiene dicho directorio.**

14. **script que devuelva una lista de texto de los archivo existentes en un directorio pasado como primer parámetro que son de un tipo dado de archivo pasado como segundo parámetro. Por ejemplo buscar solamente archivos "UTF-8 | Unicode" en el dir woek , escribir "/work UTF-8 | Unicode" .**

```
#!/bin/bash
if [ -z "$1" ] || [ -z "$2" ]
then
   echo "Debes proporcionar un directorio y un tipo de archivo como parámetros."
   echo "Uso: ./buscar_archivos.sh /ruta/del/directorio 'tipo de archivo'"
   exit 1
fi

directorio="$1"
tipo_archivo="$2"

# Verificar si el parámetro es un directorio válido
if [ ! -d "$directorio" ]
then
   echo "'$directorio' no es un directorio válido."
   exit 1
fi

# Buscar archivos en el directorio que coinciden con el tipo de archivo
echo "Buscando archivos de tipo '$tipo_archivo' en el directorio '$directorio':"
find "$directorio" -type f -exec file --mime-type -b {} \; | grep -i "$tipo_archivo" | while read -r
    tipo; do
   echo "$directorio $tipo"
    done
```

15. **script que al pasarle como parámetro un archivo de texto, ordene las líneas de texto ascendentemente al pasarle una "A" como segundo parámetro o descendentemente al pasarle una "Z".**

```
#!/bin/bash

if [ -z "$1" ] || [ -z "$2" ]
then
   echo "Debes proporcionar un archivo de texto y un parámetro ('A' para ascendente, 'Z' para descendente)."
   echo "Uso: ./ordenar_lineas.sh archivo.txt A"
   exit 1
fi

archivo="$1"
orden="$2"

if [ ! -f "$archivo" ]
then
   echo "'$archivo' no es un archivo válido."
   exit 1
fi
if [ "$orden" == "A" ]
then
   sort "$archivo"
elif [ "$orden" == "Z" ]
then
   sort -r "$archivo"
else
   echo "El segundo parámetro debe ser 'A' para ascendente o 'Z' para descendente."
   exit 1
fi
```

16. **Escriba un script que elimine un archivo o directorio pasado como parámetro, y le pregunte si está seguro de llevar a cabo la acción.**

```
#!/bin/bash

if [ -z "$1" ] || [ -z "$2" ]
then
   echo "Debes proporcionar un archivo de texto y un parámetro ('A' para ascendente, 'Z' para
     descendente)."
   echo "Uso: ./ordenar_lineas.sh archivo.txt A"
   exit 1
fi

archivo="$1"
orden="$2"

if [ ! -f "$archivo" ]
then
   echo "'$archivo' no es un archivo válido."
   exit 1
fi

if [ "$orden" == "A" ]
then
   sort "$archivo"
elif [ "$orden" == "Z" ]
then
   sort -r "$archivo"
else
   echo "El segundo parámetro debe ser 'A' para ascendente o 'Z' para descendente."
   exit 1
     fi
```

17. **Script que establece conexión a una red de datos. El script recibirá como parámetro el nombre de la red (urugnet, urupac, iie, adinet) e invoca el script correspondiente a la conexión “dip”. Se dispone de los siguientes scripts:**

```
#!/bin/bash
if [ -z "$1" ]
then
   echo "Debes proporcionar el nombre de la red (urugnet, urupac, iie, adinet)."
   echo "Uso: ./conectar_red.sh [nombre_red]"
   exit 1
fi
nombre_red="$1"
# Verificar si el nombre de la red es válido
case "$nombre_red" in
   urugnet)
      script_dip="dip_urugnet.sh"
      ;;
   urupac)
      script_dip="dip_urupac.sh"
      ;;
   iie)
      script_dip="dip_iie.sh"
      ;;
   adinet)
      script_dip="dip_adinet.sh"
      ;;
   *)
      echo "Red no reconocida. Las opciones válidas son: urugnet, urupac, iie, adinet."
      exit 1
      ;;
esac

if [ ! -f "$script_dip" ]
then
   echo "No se encontró el script '$script_dip'. Verifique que esté en la ubicación correcta."
   exit 1
```

```
fi

echo "Conectando a la red '$nombre_red'..."
    ./"$script_dip"
```

18. **Haz un script que muestre los ficheros (sólo los normales) que tengan al menos 10000 octetos en el directorio actual.**

```
#!/bin/bash

# Buscar y mostrar los archivos que tienen al menos 10000 octetos en el directorio actual
echo "Archivos con al menos 10000 octetos en el directorio actual:"

    find . -maxdepth 1 -type f -size +10000c -exec ls -lh {} \; | awk '{print $9, "- Tamaño:", $5}'
```

19. **Script que recibe un archivo y un directorio como argumentos. Comprobar que $1 es file y $2 es dir. El script deberá verificar e informar si el archivo existe en dicho directorio, sino el script deberá copiar el archivo en el directorio.**

```
#!/bin/bash

if [ -z "$1" ] || [ -z "$2" ]
then
   echo "Debes proporcionar un archivo y un directorio como parámetros."
   echo "Uso: ./copiar_archivo.sh archivo.txt /ruta/del/directorio"
   exit 1
fi

archivo="$1"
directorio="$2"

if [ ! -f "$archivo" ]
then
   echo "'$archivo' no es un archivo válido."
   exit 1
fi

if [ ! -d "$directorio" ]
then
   echo "'$directorio' no es un directorio válido."
   exit 1
fi

# Verificar si el archivo ya existe en el directorio
if [ -e "$directorio/$(basename "$archivo")" ]
then
   echo "El archivo '$archivo' ya existe en '$directorio'."
else
   cp "$archivo" "$directorio"
   echo "El archivo '$archivo' ha sido copiado a '$directorio'."
     fi
```

20. Funcion para validar varios patrones: un teléfono de 9 cifras, un numero con un solo decimal y un numero con cualquier cantidad de decimales. Después, solicita una tecla para continuar

```
#!/bin/bash
validar_patrones() {
  telefono_regex="^[0-9]{9}$"

  decimal_uno_regex="^[0-9]+\.[0-9]{1}$"

  decimal_cualquier_regex="^[0-9]+\.[0-9]+$"

  echo "Introduce un teléfono (9 cifras):"
  read telefono
  if [[ $telefono =~ $telefono_regex ]]; then
    echo "Teléfono válido: $telefono"
  else
    echo "Teléfono no válido."
  fi

  echo "Introduce un número con un solo decimal (ejemplo: 12.3):"
  read num_uno_decimal
  if [[ $num_uno_decimal =~ $decimal_uno_regex ]]; then
    echo "Número con un solo decimal válido: $num_uno_decimal"
  else
    echo "Número con un solo decimal no válido."
  fi
  echo "Introduce un número con cualquier cantidad de decimales (ejemplo: 12.3456):"
  read num_cualquier_decimal
  if [[ $num_cualquier_decimal =~ $decimal_cualquier_regex ]]; then
    echo "Número con decimales válido: $num_cualquier_decimal"
  else
    echo "Número con decimales no válido."
  fi
}
```

```
# Llamar a la función de validación
validar_patrones

echo "Presiona cualquier tecla para continuar..."
    read -n 1 -s
```

21. Script que recibe dos archivos como argumentos, el script debe validar que los parámetros recibidos sean realmente archivos y que el primer archivo (archivo de entrada) dado como argumento no está vacío, luego deberá guardar en el otro archivo (archivo de salida), las líneas que comienzan con cualquier letra de la “a” a la “m”(minúscula o mayúscula) del archivo de entrada.

```
#!/bin/bash

if [ $# -ne 2 ]; then
  echo "Debe proporcionar dos archivos como parámetros."
  echo "Uso: ./filtrar_lineas.sh archivo_entrada.txt archivo_salida.txt"
  exit 1
fi

archivo_entrada="$1"
archivo_salida="$2"

if [ ! -f "$archivo_entrada" ]; then
  echo "El archivo '$archivo_entrada' no es un archivo válido."
  exit 1
fi

if [ ! -f "$archivo_salida" ]; then
  echo "El archivo '$archivo_salida' no es un archivo válido."
  exit 1
fi

# Verificar si el archivo de entrada no está vacío
if [ ! -s "$archivo_entrada" ]; then
  echo "El archivo '$archivo_entrada' está vacío."
  exit 1
fi
```

```
# Filtrar las líneas que comienzan con letras de la 'a' a la 'm' (minúsculas o mayúsculas)
grep -i '^[a-m]' "$archivo_entrada" > "$archivo_salida"

    echo "Las líneas que comienzan con letras de la 'a' a la 'm' han sido guardadas en
    '$archivo_salida'."
```

22. **Script que pregunta el nombre que el usuario desea dar a un directorio.**

    Comprueba si el directorio existe. si existe dice "el directorio ya existe"
    Si no existe lo crea en el $HOME del usuario
    Luego pide el nombre de un archivo existente
    Copia el archivo al directorio creado previamente

```
#!/bin/bash

# Preguntar al usuario por el nombre del directorio
echo "Introduce el nombre del directorio que deseas crear:"
read nombre_directorio

# Directorio donde se creará el nuevo directorio
directorio_destino="$HOME/$nombre_directorio"

# Verificar si el directorio existe
if [ -d "$directorio_destino" ]; then
   echo "El directorio '$directorio_destino' ya existe."
else
   # Crear el directorio en el $HOME si no existe
   mkdir "$directorio_destino"
   echo "El directorio '$directorio_destino' ha sido creado."
fi

# Preguntar al usuario por el nombre del archivo existente
echo "Introduce el nombre de un archivo existente para copiar al
directorio:"
read archivo_origen

# Verificar si el archivo existe
```

```
if [ -f "$archivo_origen" ]; then
   # Copiar el archivo al nuevo directorio
   cp "$archivo_origen" "$directorio_destino"
   echo "El archivo '$archivo_origen' ha sido copiado a
'$directorio_destino'."
else
   echo "El archivo '$archivo_origen' no existe."
fi
```

23. Script para la creación de usuarios en varias máquinas remotas

```
#!/bin/bash

if [ $# -ne 2 ]; then
   echo "Uso: $0 <nombre_usuario> <archivo_ips>"
   echo "Ejemplo: $0 nuevo_usuario ips.txt"
   exit 1
fi

nombre_usuario="$1"
archivo_ips="$2"

# Verificar si el archivo de direcciones IP existe
if [ ! -f "$archivo_ips" ]; then
   echo "El archivo '$archivo_ips' no existe."
   exit 1
fi

# Leer las IPs de las máquinas remotas desde el archivo
while IFS= read -r ip; do
   if [[ -z "$ip" ]]; then
      continue
   fi
   # Intentar crear el usuario en la máquina remota
   echo "Creando usuario '$nombre_usuario' en la máquina $ip..."

   # Conexión remota para crear el usuario (necesita privilegios de root en las máquinas remotas)
   ssh "$ip" "sudo useradd -m $nombre_usuario && sudo passwd -e $nombre_usuario"

   if [ $? -eq 0 ]; then
      echo "El usuario '$nombre_usuario' ha sido creado correctamente en la máquina $ip."
   else
      echo "Error al crear el usuario '$nombre_usuario' en la máquina $ip."
   fi
     done < "$archivo_ips"
```

24. Realizar un script que reciba un número por parámetro y, si es mayor que 100, muestre el mensaje "*Número* es mayor que 100". En caso contrario que muestre el mensaje "*Número* NO es mayor que 100"

```
#!/bin/bash


if [ $# -ne 1 ]; then
   echo "Uso: $0 <número>"
   exit 1
fi


# Almacenar el número recibido como parámetro
numero=$1


# Comprobar si el número es mayor que 100
if [ "$numero" -gt 100 ]; then
   echo "Número es mayor que 100"
else
   echo "Número NO es mayor que 100"
      fi
```

25. **Realizar un script para averiguar el número más grande a partir de tres números pasados por parámetro. Mostrar un error si no se proporcionan los parámetros suficientes.**

```
#!/bin/bash

if [ $# -ne 3 ]; then
   echo "Error: Debes proporcionar tres números como parámetros."
   echo "Uso: $0 <numero1> <numero2> <numero3>"
   exit 1
fi

numero1=$1
numero2=$2
numero3=$3

# Encontrar el número más grande
if [ "$numero1" -ge "$numero2" ] && [ "$numero1" -ge "$numero3" ]; then
   echo "El número más grande es: $numero1"
elif [ "$numero2" -ge "$numero1" ] && [ "$numero2" -ge "$numero3" ]; then
   echo "El número más grande es: $numero2"
else
   echo "El número más grande es: $numero3"
     fi
```

26. **Realiza un script que comprueba si el fichero pasado por parámetro tiene permisos de lectura, en cuyo caso mostrará su contenido.**

```
#!/bin/bash

if [ $# -ne 1 ]; then
   echo "Uso: $0 <archivo>"
   exit 1
fi

# Almacenar el archivo pasado como parámetro
archivo=$1

# Comprobar si el archivo existe y tiene permisos de lectura
if [ -f "$archivo" ] && [ -r "$archivo" ]; then
   # Mostrar el contenido del archivo si tiene permisos de lectura
   echo "El archivo '$archivo' tiene permisos de lectura. Su contenido es:"
   cat "$archivo"
else
```

```
    # Si no tiene permisos de lectura o no existe
    echo "El archivo '$archivo' no tiene permisos de lectura o no existe."
fi
```

27. **Realizar un script que solo pueda ser ejecutado por el usuario root. En caso contrario devolverá una salida de error y mostrará un mensaje de error indicando que no somos root.**

```
#!/bin/bash

# Verificar si el script es ejecutado por el usuario root
if [ "$(id -u)" -ne 0 ]; then
    echo "Error: Este script debe ser ejecutado como root."
    exit 1
fi

# Si el script es ejecutado como root, continuar con la ejecución
echo "¡Hola, root! El script se ejecutó correctamente."
```

28. **Script que lista los usuarios que tenemos en el directorio «/home», una vez introducido el usuario, creará un .tar en una carpeta en «/home/copiaseg» con el nombre del usuario y la fecha en la que se creó la copia de seguridad.**

```
#!/bin/bash

# Verificar si el directorio de copias de seguridad existe, si no, crearlo
backup_dir="/home/copiaseg"
if [ ! -d "$backup_dir" ]; then
    echo "El directorio $backup_dir no existe. Creando directorio..."
    mkdir -p "$backup_dir"
fi

# Listar los usuarios que tienen directorios en /home (excepto root)
echo "Usuarios en /home:"
for user_dir in /home/*; do
    if [ -d "$user_dir" ] && [ "$(basename "$user_dir")" != "root" ]; then
        echo "$(basename "$user_dir")"
    fi
done

# Solicitar al usuario que ingrese el nombre del usuario para hacer la copia de seguridad
echo -n "Introduce el nombre del usuario para crear la copia de seguridad: "
read user

# Verificar si el directorio del usuario existe en /home
if [ -d "/home/$user" ]; then
    # Obtener la fecha actual para el nombre del archivo
    fecha=$(date +%Y-%m-%d_%H-%M-%S)

    # Crear el archivo tar en el directorio de copias de seguridad
    tar -cvf "$backup_dir/$user_$fecha.tar" -C "/home" "$user"

    if [ $? -eq 0 ]; then
        echo "Copia de seguridad de '$user' creada exitosamente en '$backup_dir/$user_$fecha.tar'."
    else
        echo "Hubo un error al crear la copia de seguridad."
    fi
else
```

```
    echo "El usuario '$user' no existe en el directorio '/home'."
    exit 1
fi
```