RESUMEN

Variables y asignación

- Puedes utilizar variables para almacenar valores y referenciarlos posteriormente.
- La asignación se realiza con el signo "=" sin espacios alrededor.
- Ejemplo: `nombre="Juan"`

Variables especiales

- `$?`: Almacena el código de salida del último comando ejecutado.
- `$$`: Almacena el PID (identificador de proceso) del script actual.
- `$#`: Almacena la cantidad de argumentos pasados al script.
- `$@` y `$*`: Representan todos los argumentos pasados al script como una lista.

Comandos básicos

- `echo`: Muestra un mensaje en la salida estándar.
- `read`: Lee la entrada del usuario y la asigna a una variable.
- `ls`: Lista los archivos y directorios en un directorio.
- `cat`: Muestra el contenido de un archivo.
- `mkdir`: Crea un nuevo directorio.
- `rm`: Elimina un archivo o directorio.
- `mv`: Mueve o renombra un archivo o directorio.
- `cp`: Copia un archivo o directorio.
- `sleep`: Pausa la ejecución de un script durante un período.
- `expr`: Ejecuta una expresión matemática.
- `cut`: Extrae secciones específicas de líneas de texto.
- `sort`: Ordena líneas de texto en orden alfabético o numérico.
- `uniq`: Encuentra y elimina líneas duplicadas consecutivas.

- `wc`**:** Cuenta las palabras, líneas y caracteres de un archivo o salida.

Estructuras de control

- `if-then-else`: Permite ejecutar diferentes acciones en función de una condición.
- `for`: Itera sobre una lista de elementos.
- `while`: Ejecuta un bloque de código mientras se cumpla una condición.

Manejo de archivos

- Puedes verificar la existencia de un archivo o directorio utilizando el comando `test` o los corchetes `[ ]`.
- Puedes utilizar el comando `find` para buscar archivos y directorios en una estructura jerárquica.
- Puedes leer y escribir archivos línea por línea utilizando el comando `while` y redireccionamiento de entrada/salida.

Expresiones condicionales

Puedes utilizar expresiones condicionales más complejas en tus estructuras de control utilizando los operadores lógicos y comparativos, como:

- Operadores de comparación: `eq` (igual), `ne` (no igual), `lt` (menor que), `gt` (mayor que), `le` (menor o igual que), `ge` (mayor o igual que).
- Operadores lógicos: `&&` (y lógico), `||` (o lógico), `!` (negación).

Expresiones regulares

- Las expresiones regulares te permiten buscar y manipular patrones de texto en tus scripts.
- Puedes utilizar el comando `grep` para buscar coincidencias de patrones en archivos o salida de comandos.
- Puedes utilizar el comando `sed` para realizar sustituciones y manipulaciones de texto basadas en expresiones regulares.

Redirección y tuberías

- `>`: Redirecciona la salida de un comando a un archivo, sobrescribiendo el contenido existente.
- `>>`: Redirecciona la salida de un comando a un archivo, añadiendo el contenido al final.
- `|`: Permite enviar la salida de un comando como entrada a otro.

Parámetros y argumentos

- Puedes pasar argumentos a un script al ejecutarlo desde la línea de comandos.
- Utiliza las variables especiales `$0`, `$1`, `$2`, etc., para acceder al nombre del script y a los argumentos.

Arreglos

- Puedes utilizar arreglos para almacenar múltiples valores en una sola variable.
- Ejemplo:

```
nombres=("Juan" "María" "Carlos")
echo ${nombres[0]}  # Acceder al primer elemento
echo ${nombres[@]}  # Mostrar todos los elementos
```

Expansión de comandos

- Puedes utilizar la expansión de comandos para capturar la salida de un comando en una variable.
- Ejemplo:

```
fecha=$(date +%Y-%m-%d)
echo "La fecha actual es: $fecha"
```

Ejercicios

```
# Muestra los n últimos
numero=$1;

# Ordena alfabeticamente los logins de usuario
usuarios=`cat /etc/passwd | sort -k1 | cut -d: -f1 |
tail -n$numero`;
```

```
echo $usuarios;
exit 0;
```

```
# Obtengo el mes
fecha=`date +%m`;

# Comprueba si está dentro de los primeros o ultimos 6 meses
if test $fecha -le 6
then
  echo "Está dentro de los primeros seis meses";
else
  echo "Está dentro de los últimos seis meses";
fi;
exit 0;
```

```
# Saluda a cada uno de los argumentos pasados
for i in $*
do
  echo "Hola $i";
done;
exit 0;
```

```
# Progresion aritmetica
numero=$1;
n=1;
veces=0;

while test $veces -lt $numero
do
  echo $n;
  n=`expr $n + 1`;
  veces=`expr $veces + 1`;
done;

echo "---"

# Progresion geometrica
n=1;
veces=0;

while test $veces -lt $numero
do
  echo $n;
  n=`expr $n \* 2`;
  veces=`expr $veces + 1`;
done;

exit 0;
```

```
# Factorial de un numero con WHILE
factorial=$1;
n=1;
resultado=1;

while test $n -le $factorial
do
  resultado=`expr $resultado \* $n`;
  n=`expr $n + 1`;
done;
echo "El factorial de $factorial es $resultado";

echo "-------------------------"

# Factorial de un numero con UNTIL
n=1;
resultado=1;

until test $n -gt $factorial
do
  resultado=`expr $resultado \* $n`;
  n=`expr $n + 1`;
done;
echo "El factorial de $factorial es $resultado";

exit 0;
```

```
# Autodestruir archivo
rm $0;
exit 0;
```

```
n1=$1;
n2=$2;
menor=0;
mayor=0;

# Compruebo cual es el menor y mayor
if test $n1 -lt $n2
then
  menor=$n1;
  mayor=$n2;
elif test $n1 -gt $n2
then
  menor=$n2;
  mayor=$n1;
else
  echo "Los número son iguales";
fi;

# Imprimo ascendentemente
until test $menor -gt $mayor
do
```

```
  echo $menor;
  menor=`expr $menor + 1`;
done;
# Imprimo descendentemente
until test $mayor -lt $menor # NO FUNCIONA
do
  echo $mayor;
  mayor=`expr $mayor - 1`;
done;

exit 0;
```

```
# Busco el interpete mas usado entre los usuarios
interprete=`cat /etc/passwd | cut -d: -f7 | sort | uniq -c |
sort -rn | head -n1 | tr -s " " | cut -d" " -f3`;

echo $interprete;
exit 0;
```

```
# Recorro cada uno de los parametros
for palabra in $*
do
  # Paso a minusculas y guardo la inicial de cada una
  palabra=`echo $palabra | tr '[:upper:]' '[:lower:]'`;
  inicial=`echo $palabra | cut -c1`;
  # Mando la palabra al archivo que comienza por su inicial
  echo $palabra >> ${inicial}.txt
done;

echo "Ha funcionado";
exit 0;
```

```
# Si existe el directorio accede a él
ruta=$1;

if test -d $ruta
then
  echo "Existe";
  cd $ruta;
else
  mkdir $ruta || echo "No se ha podido crear el directorio";
  echo "Se ha creado el directorio '$ruta'";
  cd $ruta;
fi;

exit 0;
```

```
# Compruebo quien ha ejecutado el archivo
usuario=`whoami`;

if test $usuario = "root"
then
  echo "El archivo $0 se ha ejecutado por el usuario root";
else
  echo "El archivo $0 se ha ejecutado por el usuario $usuario";
  # Ejecuto el archivo como root
  sudo $0;
  echo "Ahora por root";
fi;

exit 0;
```

```
# Obtengo el PID del archivo
echo "El PID del proceso es $$";

# Si existe, lo mata
if test -e $0
then
  kill $$;
  echo "El proceso se ha matado";
else
  echo "El proceso no se ha podido matar";
fi;

exit 0;
```

```
archivo=$1;
partes=$2;

# Compruebo que sea un archivo
if test -f $archivo
then
  echo "Existe";
else
  echo "No existe";
  exit 1;
fi;

# Cuento las lineas, reparto las lineas por parte y divido
nLineas=`wc -l < $archivo`;
lineasParte=`expr $nLineas / $partes`;
split -l $lineasParte $archivo;

# Creo los archivos necesarios para repartir las partes
cantidadArchivos=`ls -l x* | wc -l`;

# Compruebo que las partes son iguales al numero de archivos
if test $cantidadArchivos -ne $partes
then
```

```
  echo "Número de líneas no divisible por $partes";
  rm x*;
else
  echo "El archivo $archivo se ha dividido en $partes partes";
fi;

exit 0;
```

```
archivo=$1;
paquete=$2;

# Compruebo que el archivo pertenezca al paquete
if `dpkg -S $archivo | grep -q $paquete`
then
  echo "$archivo pertenece al paquete $paquete";
else
  echo "$archivo no pertenece al paquete $paquete";
fi;

exit 0;
```

```
a=`cat ./$a1 2> /dev/null`;

# Recorro las lineas del parametro 1 (archivo)
while read linea
do
  a1=`echo $linea | cut -d- -f1`;
  a2=`echo $linea | cut -d- -f2`;

  # Compruebo que existe el a1
  if test -e $a1
  then
    cp $a1 $a2;
    echo "$a1 se ha copiado en $a2";
  else
    cp /etc/group $a2;
    echo "/etc/group se ha copiado en $a2";
  fi;
done < $1;

exit 0;
```

```
directorio=$1;
palabra=$2;

# Compruebo que es un directorio
if test -d $directorio
then
  echo "Es un directorio";
else
```

```
  echo "No es un directorio";
fi;

# Recorro cada archivo del directorio
for archivo in $directorio/*
do
  # Compruebo que es un archivo
  if test -f $archivo
  then
    # Cuento las ocurrencias y las imprimo por archivo
    ocurrencias=`grep -o -i $palabra $archivo | wc -l`;
    echo "Archivo: $archivo - Ocurrencias: $ocurrencias";
  fi;
done;

exit 0;
```

```
directorio=$1;

# Compruebo que sea un directorio
if test -d $directorio
then
  echo "Es un directorio";
  # Determino el nombre del directorio a comprimir
  nombre=`date +"backup_%Y_%m_%d.tar.gz"`;
  # Comprimo el directorio
  tar -czf $nombre $directorio 2> /dev/null;
else
  echo "No es un directorio";
fi;

exit 0;
```

```
valor=$1;

# Compruebo que se ejecute como root
if `id -u != 0`
then
  echo "Debe ejecutarse como root";
fi;

# Compruebo que el valor esté dentro del rango válido
if `$valor -lt 0 || $valor -gt 6`
then
  echo "Error: el runlevel $valor no es válido (entre 0 y 6)";
fi;

# Ejecuto el runlevel
init $valor;

exit 0;
```

```
directorio=$1;

# Busco los archivos .c
for archivo in $directorio/*.c
do
  # Cuento los archivos .c que hay
  numero=`grep -c . $archivo`;
  total="$archivo $numero\n$total";
  echo "$total";
done;

echo -e "$total" | sort -n -r -k2 | tr -s "/" > linias.txt;

exit 0;
```

```
#
while read -r linea
do
  # Selecciono cada palabra de cada linea
  for palabra in $linea
  do
    # Cantidad de archivos que contienen la palabra, o en
    # los subdirectorios
    nArchivos=`grep -lRw $palabra . | wc -l`;
    echo "$palabra: $nArchivos";
  done;
done < $1;

exit 0;
```