



Big Data

2025/26

Ciclo	Big Data Aplicado y Sistemas de Big Data
Nombre	Carmen García Rodríguez
Correo	carmengr36@educastur.es
Nº Unidad Didáctica	1

Tabla de Contenido

Tabla de Contenido.....	1
1. Conceptos Básicos y Variables.....	2
2. Obtener Ayuda y Localizar Archivos.....	4
3. Navegación y listado de archivos.....	6
4. Manipulación de Archivos y Directorios.....	7
5. Archivado y compresión.....	8
6. Redirección, Tuberías y Filtros.....	10
7. Scripts Básicos.....	12
8. Ejercicios Avanzados.....	16

1. Conceptos Básicos y Variables

1.1. Muestra el contenido de tu variable de entorno HOME . Luego, usa cd junto con esa variable para navegar a dicho directorio y verifica con pwd que te encuentras en la ubicación correcta.

```
carmen@servidor:~$ echo $HOME
/home/carmen
carmen@servidor:~$ cd $HOME
carmen@servidor:~$ pwd
/home/carmen
```

1.2. Ejecuta el comando whoami . Ahora, crea una variable local llamada USUARIO_ACTUAL que contenga el resultado del comando anterior y muéstrala en la terminal.

```
carmen@servidor:~$ whoami
carmen
carmen@servidor:~$ USUARIO_ACTUAL=$(whoami)
carmen@servidor:~$ echo $USUARIO_ACTUAL
carmen
```

1.3. Intenta crear un archivo llamado dos palabras.txt sin usar comillas. Observa el resultado con ls . ¿Qué ha ocurrido y por qué? Ahora, bórralo(s) y créalo correctamente.

```
carmen@servidor:~$ touch dos palabras.txt
carmen@servidor:~$ ls
dos palabras.txt typescript
```

El comando `ls` muestra que hemos creado dos archivos en lugar de uno ya que el comando `touch` delimita los archivos con espacios, por lo que interpretó que queríamos crear `dos` y `palabras.txt`

```
carmen@servidor:~$ rm dos palabras.txt
carmen@servidor:~$ touch "dos palabras.txt"
carmen@servidor:~$ ls
'dos palabras.txt' typescript
```

1.4. Usa el comando type para averiguar si ls y cd son internos o externos al shell. ¿Qué diferencia práctica crees que implica esto?

```
carmen@servidor:~$ type ls
ls is aliased to `ls --color=auto'
carmen@servidor:~$ type cd
cd is a shell builtin
```

`cd` es un comando interno (shell builtin) y `ls` es un comando externo que se ejecuta a través de un alias. Las diferencias prácticas pueden ser que los comandos externos se ejecutan en un nuevo proceso hijo de shell, por lo que son más lentos y solo afectan a ese nuevo proceso hijo.

1.5. Muestra tu PATH actual. Crea un directorio ~/mi_bin y añádelo temporalmente al principio de tu PATH . Verifica que el cambio se ha realizado correctamente.

```
carmen@servidor:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
carmen@servidor:~$ mkdir ~/mi_bin
carmen@servidor:~$ export PATH=~/mi_bin:$PATH
carmen@servidor:~$ echo $PATH
/home/carmen/mi_bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
carmen@servidor:~$
```

2. Obtener Ayuda y Localizar Archivos

2.1. Abre la página del manual para el comando `chmod` . ¿En qué sección del manual se encuentra? ¿Qué indica ese número de sección sobre el tipo de comando?

Se abre el manual con `man chmod`.

La cabecera del manual `CHMOD(1)` indica que está en la sección 1. Esto significa que es un comando del tipo “programa ejecutable” como `ls` o `grep`

2.2. Usando la función de búsqueda dentro de la página del manual de `ls`, encuentra la opción que ordena los archivos por tamaño.

Con `man ls` se abre el manual. Con `/size` podemos buscar las partes del manual donde aparece la palabra `size`. La opción que ordena por tamaño es `-S`, de mayor a menor.

```
carmen@servidor:~$ ls -S
mi_bin 'dos palabras.txt' typescript
```

2.3. Imagina que has olvidado dónde se guarda el archivo de configuración de usuarios. Sabiendo que se llama `passwd` , usa `find` para buscarlo desde el directorio raíz (`/`). Anota la ruta completa que has encontrado.

```
carmen@servidor:~$ find / -name passwd 2>/dev/null
/etc/passwd
/etc/pam.d/passwd
/usr/share/linbian/overrides/passwd
/usr/share/bash-completion/completions/passwd
/usr/share/doc/passwd
/usr/bin/passwd
```

El archivo está en `/etc/passwd`.

`2>dev/null` sirve para que el resultado no muestre mensajes de error de directorios a los que no puede acceder, si no el resultado sería algo así:

```
/find: '/proc/1444/task/1444/fdinfo': Permission denied
find: '/proc/1444/task/1444/ns': Permission denied
find: '/proc/1444/fd': Permission denied
find: '/proc/1444/map_files': Permission denied
find: '/proc/1444/fdinfo': Permission denied
find: '/proc/1444/ns': Permission denied
find: '/proc/1445/task/1445/fd': Permission denied
find: '/proc/1445/task/1445/fdinfo': Permission denied
find: '/proc/1445/task/1445/ns': Permission denied
find: '/proc/1445/fd': Permission denied
find: '/proc/1445/map_files': Permission denied
find: '/proc/1445/fdinfo': Permission denied
find: '/proc/1445/ns': Permission denied
find: '/sys/kernel/tracing': Permission denied
find: '/sys/kernel/debug': Permission denied
find: '/sys/fs/pstore': Permission denied
find: '/sys/fs/bpf': Permission denied
find: '/lost+found': Permission denied
find: '/root': Permission denied
find: '/run/udisks2': Permission denied
find: '/run/user/1000/systemd/inaccessible/dir': Permission denied
```

2.4. Crea un archivo vacío llamado test_locate.txt en tu directorio home. Inmediatamente después, búscalo con locate . ¿Aparece en los resultados? ¿Por qué sí o por qué no?

```
carmen@servidor:~$ touch ~/test_locate.txt
carmen@servidor:~$ locate test_locate.txt
```

`locate` no encuentra el archivo. Esto sucede porque lo acabamos de crear y `locate` busca en una base de datos pre-indexada que se actualiza cada cierto tiempo (una vez al día, al actualizar el equipo o al a través de una tarea programada de `cron` o `systemd`)

2.5. Basado en el ejercicio anterior, ¿qué comando (probablemente con sudo) necesitas ejecutar para que locate sí encuentre tu archivo? Ejecútalo y verifica que ahora sí lo encuentras.

```
carmen@servidor:~$ sudo updatedb
carmen@servidor:~$ locate test_locate.txt
/home/carmen/test_locate.txt
```

Podemos actualizar la base de datos con `sudo updatedb` y al volver a ejecutar `locate` vemos que ya encuentra el archivo

3. Navegación y listado de archivos

3.1. Navega al directorio /etc . Desde ahí, sin usar cd , lista el contenido de tu directorio home usando una ruta con el atajo ~ .

```
carmen@servidor:~$ cd /etc
carmen@servidor:/etc$ ls ~
'dos palabras.txt'  mi_bin  test_locate.txt  typescript
```

3.2. Desde tu directorio home , navega a / y luego a var y finalmente a log usando una sola línea de comando y rutas relativas.

```
carmen@servidor:/etc$ cd ~
carmen@servidor:~$ cd ../../var/log
carmen@servidor:/var/log$
```

3.3. Lista el contenido de /etc en formato largo. En la salida, identifica el propietario, el grupo y los permisos del archivo passwd .

```
carmen@servidor:~$ ls -l /etc | grep passwd
-rw-r--r-- 1 root root      1737 oct 15 09:46 passwd
-rw-r--r-- 1 root root      1688 oct 15 09:36 passwd-
```

El comando **grep** devuelve solo los resultados que incluyen lo que le indicamos

3.4. Compara la salida de ls -l /etc y ls -lh /etc . ¿Qué hace la opción h y por qué es útil para las personas?

ls -l /etc muestra por defecto el tamaño de los archivos en bytes

```
-rw-r--r-- 1 root root      4343 jun 25 12:42 sudo.conf
-r--r----- 1 root root      1800 ene 29  2024 sudoers
drwxr-xr-x 2 root root      4096 ago  5 17:02 sudoers.d
-rw-r--r-- 1 root root      9804 jun 25 12:42 sudo_logsrvd.conf
drwxr-xr-x 2 root root      4096 ago  5 17:14 supercat
-rw-r--r-- 1 root root      2209 mar 24  2024 sysctl.conf
```

ls -lh /etc muestra el tamaño de los archivos de una forma más legible usando unidades que se adapten al tamaño del archivo, es este caso kb pero también podrían ser mb o gb en función del tamaño del archivo.

```
-r--r----- 1 root root      1,8K ene 29  2024 sudoers
drwxr-xr-x 2 root root      4,0K ago  5 17:02 sudoers.d
-rw-r--r-- 1 root root      9,6K jun 25 12:42 sudo_logsrvd.conf
drwxr-xr-x 2 root root      4,0K ago  5 17:14 supercat
-rw-r--r-- 1 root root      2,2K mar 24  2024 sysctl.conf
drwxr-xr-x 2 root root      4,0K oct 20 08:12 sysctl.d
drwxr-xr-x 2 root root      4,0K ago  5 17:14 sysstat
```

3.5. Ejecuta ls -R ~ . ¿Qué hace la opción R ? ¿Por qué podría ser peligroso usarla en el directorio raíz (/)?

La opción **-R** indica que se liste de manera recursiva, es decir, no lista solamente la carpeta que se indica si no que entra dentro de todas las subcarpetas que contenga y lista su contenido incluyendo el contenido de sus carpetas, así hasta mostrar todo el árbol de archivos. Esto puede ser un problema ejecutado en la carpeta home debido a la magnitud de la operación, esta consumirá muchos recursos y puede bloquear el equipo

4. Manipulación de Archivos y Directorios

4.1. Crea la estructura de directorios `proyecto/src` , `proyecto/doc` y `proyecto/bin` usando un único comando `mkdir` .

```
carmen@servidor:~$ mkdir -p proyecto/{src,doc,bin}
```

4.2. Crea un archivo `~/notas.txt` . Muévelo a `~/proyecto/doc` y, en el mismo comando, renómbralo a `README.md` .

```
carmen@carmen-portatil:~$ touch ~/notas.txt
carmen@carmen-portatil:~$ mv ~/notas.txt ~/proyecto/doc/README.md
```

4.3. Copia el archivo `README.md` de `proyecto/doc` a `proyecto/bin` . Luego, borra el archivo original de la carpeta `doc` .

```
carmen@carmen-portatil:~$ cp ~/proyecto/doc/README.md ~/proyecto/bin/
carmen@carmen-portatil:~$ rm ~/proyecto/doc/README.md
```

4.4. Intenta borrar el directorio `proyecto` con `rmdir` . ¿Qué error obtienes? Ahora, usa `rm` con la opción correcta para borrar el directorio y todo lo que contiene.

```
carmen@carmen-portatil:~$ rmdir ~/proyecto
rmdir: fallo al borrar '/home/carmen/proyecto': El directorio no está vacío
carmen@carmen-portatil:~$ rm -rf ~/proyecto
```

No se puede eliminar porque `rmdir` solo puede eliminar directorios vacíos. Para borrarlo hay que usar `rm -r` para que actúe de manera recursiva borrando directorios y su contenido. También se puede añadir `-f` para que no pida confirmación.

4.5. Navega a `/etc` . Usando un solo comando `ls` con globbing , lista todos los archivos que empiecen con la letra `s` y terminen con `.conf` .

```
carmen@carmen-portatil:~$ ls /etc/s*.conf
/etc/sensors3.conf /etc/sudo.conf /etc/sudo_logsrvd.conf /etc/sysctl.conf
```


5. Archivado y compresión

5.1. Crea un archivo tar llamado log_backup.tar que contenga todos los archivos del directorio /var/log . ¿Qué advertencias de “permiso denegado” aparecen y por qué?

El comando para crear el archivo con el contenido de /var/log es

```
tar -cvf ~/log_backup.tar /var/log
```

Al ejecutarlo da errores de permiso denegado ya que la mayoría de los archivos de este directorio pertenecen a root y necesitamos permisos de superusuario para acceder.

Es necesario ejecutar el comando con sudo

```
/var/log/ubuntu-advantage.log.3.gz
carmen@carmen-portatil:~$ sudo tar -cvf ~/log_backup.tar /var/log
```

5.2. Comprime el archivo log_backup.tar con gzip . Compara el tamaño del archivo original y el comprimido usando ls -lh .

```
/var/log/ubuntu-advantage.log.3.gz
carmen@carmen-portatil:~$ gzip ~/log_backup.tar

carmen@carmen-portatil:~$ ls -lh ~/log_backup.tar*
-rw-r--r-- 1 root root 2,4G oct 20 19:20 /home/carmen/log_backup.tar
-rw-rw-r-- 1 carmen carmen 263M oct 20 18:53 /home/carmen/log_backup.tar.gz
carmen@carmen-portatil:~$
```

Podemos ver que sin comprimir ocupa 2,4Gb y comprimido solo 263Mb

5.3. Lista el contenido del archivo log_backup.tar.gz sin extraerlo para verificar que los archivos están dentro.

```
carmen@carmen-portatil:~$ tar -tzf ~/log_backup.tar.gz
var/log/
var/log/dist-upgrade/
var/log/boot.log.3
var/log/syslog.2.gz
var/log/dpkg.log
var/log/apt/
var/log/apt/history.log.8.gz
var/log/apt/term.log.8.gz
var/log/apt/term.log.6.gz
var/log/apt/term.log
var/log/apt/term.log.4.gz
var/log/apt/term.log.10.gz
var/log/apt/history.log.4.gz
var/log/apt/term.log.3.gz
var/log/apt/history.log.7.gz
var/log/apt/term.log.2.gz
var/log/apt/history.log.12.gz
var/log/apt/history.log.11.gz
var/log/apt/history.log.9.gz
var/log/apt/history.log.3.gz
var/log/apt/history.log.6.gz
```

5.4. Extrae únicamente el archivo syslog (o messages) de log_backup.tar.gz a tu directorio /tmp .

```
tar -xzf ~/log_backup.tar.gz -C /tmp var/log/syslog
```

```
carmen@carmen-portatil:~$ tar -xzf ~/log_backup.tar.gz -C /tmp var/log/syslog
carmen@carmen-portatil:~$ ls /tmp/var/log/syslog
/tmp/var/log/syslog
carmen@carmen-portatil:~$ cd /tmp/var/log/syslog
```

El comando tar con la ruta exacta del archivo dentro del archivo comprimido.

- x Indica a **tar** que debe extraer archivos.
- z Indica a **tar** que el archivo está comprimido con **gzip** y debe descomprimirse primero.
- f Especifica el archivo de entrada (**~/log_backup.tar.gz**).
- C /tmp Le dice a **tar** que cambie al directorio **/tmp** antes de comenzar la extracción. Esto asegura que el archivo se extraiga allí.

5.5. Crea tres archivos (a.txt , b.log , c.jpg) y luego crea un archivo zip que los contenga.

```
carmen@carmen-portatil:/tmp/var/log$ touch a.txt b.log c.jpg
carmen@carmen-portatil:/tmp/var/log$ zip archivo.zip a.txt b.log c.jpg
  adding: a.txt (stored 0%)
  adding: b.log (stored 0%)
  adding: c.jpg (stored 0%)
```

5.6. Elimina los tres archivos originales y luego recupéralos desde el archivo zip.

```
carmen@carmen-portatil:/tmp/var/log$ rm a.txt b.log c.jpg
carmen@carmen-portatil:/tmp/var/log$ unzip archivo.zip
Archive:  archivo.zip
  extracting: a.txt
  extracting: b.log
  extracting: c.jpg
```

5.7. Usa zcat (o gzcat) para leer el contenido de un archivo de log comprimido (ej: en /var/log , busca uno que termine en .gz) sin crear un archivo descomprimido.

```
carmen@carmen-portatil:/tmp/var/log$ echo "este es mi archivo de log comprimido" > comprimido.log
carmen@carmen-portatil:/tmp/var/log$ gzip comprimido.log
carmen@carmen-portatil:/tmp/var/log$ zcat comprimido.log.gz
este es mi archivo de log comprimido
```

6. Redirección, Tuberías y Filtros

6.1. Guarda la lista de archivos de tu directorio home (formato largo) en un archivo mis_archivos.txt .

```
carmen@servidor:~$ ls -l ~ > mis_archivos.txt
```

>: Es el operador de **redirección**. Este símbolo toma la salida estándar del comando que le precede (`ls -l ~`) y la envía al archivo especificado (`mis_archivos.txt`), **sobrescribiendo** el contenido si el archivo ya existe.

```
GNU nano 7.2 mis_archivos.txt
total 8
-rw-rw-r-- 1 carmen carmen    0 oct 20 06:55 dos palabras.txt
drwxrwxr-x 2 carmen carmen 4096 oct 20 07:33 mi_bin
-rw-rw-r-- 1 carmen carmen    0 oct 21 06:50 mis_archivos.txt
drwxrwxr-x 5 carmen carmen 4096 oct 20 09:26 proyecto
-rw-rw-r-- 1 carmen carmen    0 oct 20 08:15 test_locate.txt
-rw-rw-r-- 1 carmen carmen    0 oct 16 07:07 typescript
```

6.2. Sin borrar el contenido anterior, añade la fecha y hora actual al final del archivo mis_archivos.txt .

```
carmen@servidor:~$ date >> mis_archivos.txt
```

>> También envía el resultado al archivo especificado pero en vez de sobrescribir como hace > lo añade al final del contenido.

```
total 8
-rw-rw-r-- 1 carmen carmen    0 oct 20 06:55 dos palabras.txt
drwxrwxr-x 2 carmen carmen 4096 oct 20 07:33 mi_bin
-rw-rw-r-- 1 carmen carmen    0 oct 21 06:50 mis_archivos.txt
drwxrwxr-x 5 carmen carmen 4096 oct 20 09:26 proyecto
-rw-rw-r-- 1 carmen carmen    0 oct 20 08:15 test_locate.txt
-rw-rw-r-- 1 carmen carmen    0 oct 16 07:07 typescript
mar 21 oct 2025 06:53:08 UTC
```

6.3. Usa grep y una tubería (|) para contar el número de directorios que hay en /etc (Pista: `ls -l | grep '^d'`).

```
carmen@servidor:~$ ls -l | grep '^d'
drwxrwxr-x 2 carmen carmen 4096 oct 20 07:33 mi_bin
drwxrwxr-x 5 carmen carmen 4096 oct 20 09:26 proyecto
```

Al añadir `grep '^d'` el listado se filtra y muestra solamente los elementos que empiecen por `d` y en el formato de listado que muestra `ls -l` los directorios se marcan con una `d` delante de los permisos.

6.4. Muestra las 10 últimas líneas del archivo /etc/passwd y, usando otra tubería, extrae solo los nombres de usuario (el primer campo).

tail -n 10 /etc/passwd | cut -d: -f1

```
carmen@servidor:~$ tail -n 10 /etc/passwd | cut -d: -f1
pollinate
polkitd
syslog
uuid
tcpdump
tss
landscape
fwupd-refresh
usbmux
carmen
```

tail empieza a seleccionar por el final y **-n 10** indica que seleccione 10 líneas

-d: especifica que el carácter que separa los campos es :

-f1: ya que sabemos que el nombre de usuario es el primer campo lo seleccionamos con **f1**

6.5. Muestra una lista de todos los procesos del sistema (ps aux), ordénala por uso de CPU (tercera columna) y muestra solo las 5 líneas superiores.

ps aux --sort=-%cpu | head -n 6

```
carmen@servidor:~$ ps aux --sort=-%cpu | head -n 6
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      8781  0.6  2.1 594384 42960 ?        Ssl   07:10   0:01 /usr/libexec/fwupd/fwupd
root      8482  0.1  0.0      0      0 ?        I     06:29   0:04 [kworker/0:1-cgroup_destroy]
root       16  0.1  0.0      0      0 ?        S     03:29   0:23 [ksoftirqd/0]
root     8338  0.1  0.0      0      0 ?        I     05:14   0:09 [kworker/0:0-events]
root       1  0.0  0.6  22644 13952 ?        Ss    03:29   0:04 /usr/lib/systemd/systemd --system
```

--sort=-%cpu ordena la lista en función del uso de CPU. El signo menos (-) antes del campo **%cpu** asegura que la ordenación sea **descendente** (de mayor a menor uso de CPU).

head -n 6 selecciona solo las 6 primeras líneas.

6.6. ¿Cuál es la diferencia entre usar > y >> para redirigir la salida de un comando a un archivo? Demuéstralo con un ejemplo.

Tanto **>** como **>>** son operadores de redirección, es decir, envían la salida de un comando a un archivo. La diferencia es que **>** sobrescribe el archivo y **>>** añade la salida del comando a continuación de el contenido que ya tenga el archivo.

Usando **mis_archivos.txt** :

```
carmen@servidor:~$ whoami >> mis_archivos.txt _
```

```
total 8
-rw-rw-r-- 1 carmen carmen    0 oct 20 0
drwxrwxr-x 2 carmen carmen 4096 oct 20 0
-rw-rw-r-- 1 carmen carmen    0 oct 21 0
drwxrwxr-x 5 carmen carmen 4096 oct 20 0
-rw-rw-r-- 1 carmen carmen    0 oct 20 0
-rw-rw-r-- 1 carmen carmen    0 oct 16 0
mar 21 oct 2025 06:53:08 UTC
carmen
```

Usando **>>** escribe el nombre de usuario al final del archivo

```
carmen@servidor:~$ whoami > mis_archivos.txt
```

```
GNU nano 7.2  
carmen
```

Usando > sobrescribe el archivo por lo que ahora solo contiene el nombre de usuario.

6.7. Ejecuta `find /etc -name "*.conf"` . Redirige la salida estándar a un archivo `config_files.txt` y los errores (si los hay) a `errors.txt` .

`find /etc -name "*.conf" > config_files.txt 2> errors.txt`

```
carmen@servidor:~$ find /etc -name "*.conf" > config_files.txt 2> errors.txt  
carmen@servidor:~$
```

```
GNU nano 7.2 config_files.txt  
etc/locale.conf  
etc/e2scrub.conf  
etc/modprobe.d/mdadm.conf  
etc/modprobe.d/blacklist-ath_pci.conf  
etc/modprobe.d/blacklist-firewire.conf  
etc/modprobe.d/iwlwifi.conf  
etc/modprobe.d/blacklist-framebuffer.conf  
etc/modprobe.d/blacklist.conf  
etc/modprobe.d/blacklist-rare-network.conf  
etc/modprobe.d/amd64-microcode-blacklist.conf  
etc/modprobe.d/intel-microcode-blacklist.conf  
etc/fonts/conf_avail/57-dejavu-sans.conf
```

```
GNU nano 7.2 errors.txt  
find: '/etc/polkit-1/rules.d': Permission denied  
find: '/etc/credstore': Permission denied  
find: '/etc/ssl/private': Permission denied  
find: '/etc/credstore.encrypted': Permission denied  
find: '/etc/multipath': Permission denied
```

7. Scripts Básicos

7.1. Crea un script que imprima tu nombre de usuario y el directorio de trabajo actual usando las variables de entorno correspondientes.

```
carmen@carmen-portatil:~$ nano info_user.sh
#!/bin/bash
echo "user : $USER"
echo "directorio actual : $PWD"
```

7.2. Haz el script anterior ejecutable solo para ti (`chmod u+x ...`) y ejecútalo. Luego, intenta ejecutarlo como otro usuario (si es posible) o explica qué pasaría.

```
carmen@carmen-portatil:~$ chmod u+x info_user.sh
carmen@carmen-portatil:~$ ./info_user.sh
user : carmen
directorio actual : /home/carmen
```

Si otro usuario intentara lanzar el script le daría el error Permiso denegado

7.3. Modifica el script para que acepte un argumento. Si el argumento es “hola”, debe imprimir “mundo”. Si es cualquier otra cosa, no debe imprimir nada.

```
#!/bin/bash
if [ "$1" == "hola" ]; then
    echo "mundo"
fi

carmen@carmen-portatil:~$ ./info_user.sh hola
mundo
carmen@carmen-portatil:~$ ./info_user.sh carmen
```

7.4. Mejora el script anterior para que, si no se proporciona ningún argumento, muestre un mensaje de uso: “Error: Debes proporcionar un argumento.”

```
#!/bin/bash
if [ "$1" == "hola" ]; then
    echo "mundo"
fi

if [ "$#" -eq 0 ]; then
    echo "Error: Debes proporcionar un argumento"
    exit 1 # Salir del script con código de error 1
fi

carmen@carmen-portatil:~$ ./info_user.sh
Error: Debes proporcionar un argumento
```

7.5. Escribe un script que reciba dos números. Debe imprimir “iguales” si son iguales y “diferentes” si no lo son.

```
#!/bin/bash

if [ "$#" -ne 2 ]; then
    echo "Error: Debes proporcionar exactamente dos números como argumentos."
    echo "Uso: $0 <numero1> <numero2>"
    exit 1
fi

NUM1=$1
NUM2=$2

if [ "$NUM1" -eq "$NUM2" ]; then
    echo "iguales"
else
    echo "diferentes"
fi
```

```
carmen@carmen-portatil:~$ ./info_user.sh 2 2
iguales
carmen@carmen-portatil:~$ ./info_user.sh 2 4
diferentes
```

7.6. Escribe un script que, dado un directorio como argumento, use un bucle for para iterar sobre su contenido (ls \$1) y añada la extensión .bak a cada archivo.

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "Error: Debes proporcionar la ruta de un directorio."
    echo "Uso: $0 <ruta_del_directorio>"
    exit 1
fi

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

DIR_PATH="$1"

echo "Procesando archivos en: $DIR_PATH"

for FILENAME in $(ls -A "$DIR_PATH"); do
    FULL_PATH="$DIR_PATH/$FILENAME"

    if [ -f "$FULL_PATH" ]; then
        mv "$FULL_PATH" "$FULL_PATH.bak"
        echo "-> Renombrado: $FILENAME a $FILENAME.bak"
    fi
done

echo "Proceso completado."
```

```
carmen@carmen-portatil:~$ ls -lisa test_dir
total 8
12592673 4 drwxrwxr-x  2 carmen carmen 4096 oct 21 20:26 .
11665410 4 drwxr-x--- 37 carmen carmen 4096 oct 21 20:28 ..
12593000 0 -rw-rw-r--  1 carmen carmen    0 oct 21 20:26 archivo1.txt
12593001 0 -rw-rw-r--  1 carmen carmen    0 oct 21 20:26 documento2
12593002 0 -rw-rw-r--  1 carmen carmen    0 oct 21 20:26 .oculto
carmen@carmen-portatil:~$ ./info_user.sh test_dir
Procesando archivos en: test_dir
-> Renombrado: archivo1.txt a archivo1.txt.bak
-> Renombrado: documento2 a documento2.bak
-> Renombrado: .oculto a .oculto.bak
Proceso completado.
```


8. Ejercicios Avanzados

1. Muestra los shells de los usuarios listados en `/etc/passwd` , elimina las líneas duplicadas y ordénalos alfabéticamente. (Pista: `cut` , `sort` , `uniq`).

`cut -d: -f7 /etc/passwd | sort | uniq`

```
carmen@carmen-portatil:~$ cut -d: -f7 /etc/passwd | sort | uniq
/bin/bash
/bin/false
/bin/sync
/usr/sbin/nologin
```

`-f7` porque la ruta del shell es el campo número 7 en `passwd`

2. Usando `ps` , `grep` y `wc` , crea un comando de una sola línea que te diga cuántos procesos está ejecutando el usuario `root` actualmente.

`ps -ef | grep '^root' | wc -l`

```
carmen@carmen-portatil:~$ ps -ef | grep '^root' | wc -l
6178
```

`-e` selecciona todos los procesos.

`-f` muestra la lista en formato "full" o completo, lo que incluye la columna del usuario al inicio.

`wc` (word count) cuenta líneas, palabras o caracteres.

`-l` indica a `wc` que cuente el **número de líneas** (`l` for lines), lo que equivale al número total de procesos del usuario `root`.

3. Lista todos los archivos en `/etc` , filtra los resultados para mostrar solo aquellos que han sido modificados en "Oct" (octubre) y guarda esa lista en `october_files.txt`.

```
carmen@carmen-portatil:~$ ls -l /etc | grep ' oct ' > october_files.txt
carmen@carmen-portatil:~$
```

GNU nano 6.2				october_files.txt			
drwxr-xr-x	3	root	root	4096	oct 27	2024	apparmor
-rw-r--r--	1	root	root	6890	oct 27	2024	ca-certificates.conf
drwxr-xr-x	5	root	lp	4096	oct 21	18:26	cups
-rw-r--r--	1	root	root	1166	oct 20	16:39	group
-rw-r-----	1	root	shadow	967	oct 20	16:39	gshadow
-rw-r--r--	1	root	root	92	oct 15	2021	host.conf
-rw-r--r--	1	root	root	85980	oct 20	16:39	ld.so.cache
-rw-r--r--	1	root	root	267	oct 15	2021	legal
-rw-r--r--	1	root	root	41672	oct 20	16:37	mailcap
-rw-r--r--	1	root	root	91	oct 15	2021	networks
-rw-r--r--	1	root	root	582	oct 15	2021	profile
-rwxr-xr-x	1	root	root	1148	oct 22	2024	qemu-ifup
drwxr-xr-x	4	root	root	4096	oct 4	11:59	ssl

4. Usando globbing , lista todos los archivos en /etc que contengan un número en su nombre.

```
carmen@carmen-portatil:~$ ls -l /etc/*[0-9]*
-rw-r--r-- 1 root root 685 ene 8 2022 /etc/e2scrub.conf
-rw-r--r-- 1 root root 744 ene 8 2022 /etc/mke2fs.conf
-rw-r--r-- 1 root root 7649 ago 8 2023 /etc/pnm2ppa.conf
-rw-r--r-- 1 root root 10593 mar 31 2022 /etc/sensors3.conf

/etc/apache2:
total 4
drwxr-xr-x 2 root root 4096 abr 8 2024 conf-available

/etc/dbus-1:
total 8
drwxr-xr-x 2 root root 4096 abr 1 2022 session.d
drwxr-xr-x 2 root root 4096 jul 7 13:21 system.d

/etc/gdm3:
total 44
```

5. Usando find , busca en /usr/bin todos los archivos que sean ejecutables, pero que no sean propiedad del usuario root .

```
find /usr/bin -type f -executable \! -user root
```

```
carmen@servidor:~$ find /usr/bin -type f -executable \! -user root
carmen@servidor:~$
```

No devuelve nada porque todos los archivos ejecutables que hay en /usr/bin pertenecen a root. Esto suele ser la configuración de seguridad normal en linux.

6. Compara la diferencia de tamaño y velocidad al comprimir un archivo grande (puedes usar /var/log/syslog) con gzip y con bzip2 .

```
carmen@carmen-portatil:~$ cp /var/log/syslog prueba_grande
carmen@carmen-portatil:~$ time gzip -k prueba_grande

real    0m0,057s
user    0m0,039s
sys     0m0,005s
carmen@carmen-portatil:~$ time bzip2 -k prueba_grande

real    0m0,137s
user    0m0,121s
sys     0m0,015s
carmen@carmen-portatil:~$
```

Vemos que todos los tiempo son mayores al comprimir con bzip2

```
sys     0m0,015s
carmen@carmen-portatil:~$ ls -lh prueba_grande prueba_grande.gz prueba_grande.b
z2
-rw-r----- 1 carmen carmen 1,4M oct 21 20:56 prueba_grande
-rw-r----- 1 carmen carmen 150K oct 21 20:56 prueba_grande.bz2
-rw-r----- 1 carmen carmen 209K oct 21 20:56 prueba_grande.gz
carmen@carmen-portatil:~$
```

Sin embargo el archivo bzip2 ocupa bastante menos que el comprimido con gzip

7. Crea un archivo tar de tu directorio home, pero esta vez, usa la opción para seguir enlaces simbólicos. Antes, crea un enlace simbólico en tu home para que puedas ver la diferencia.

```
carmen@servidor:~$ echo "Esto es el archivo original en /tmp" > /tmp/archivo_original.txt
```

```
carmen@servidor:~$ ln -s /tmp/archivo_original.txt $HOME/enlace_a_archivo_tmp.txt
```

```
carmen@servidor:~$ tar -cvhf **home_con_contenido_enlaces.tar** $HOME/
tar: Removing leading `/' from member names
/home/carmen/
/home/carmen/.bashrc
tar: Removing leading `/' from hard link targets
/home/carmen/config_files.txt
/home/carmen/test_locate.txt
/home/carmen/.local/
/home/carmen/.local/share/
/home/carmen/.local/share/nano/
/home/carmen/.bash_history
/home/carmen/.sudo_as_admin_successful
/home/carmen/.lessht
/home/carmen/.ssh/
/home/carmen/.ssh/authorized_keys
/home/carmen/errors.txt
/home/carmen/.cache/
/home/carmen/.cache/motd.legal-displayed
tar: /home/carmen/**home_con_contenido_enlaces.tar**: archive cannot contain itself; not dumped
/home/carmen/mi_bin/
/home/carmen/.profile
/home/carmen/enlace_a_archivo_tmp.txt
/home/carmen/mis_archivos.txt
/home/carmen/typescript
/home/carmen/proyecto/
/home/carmen/proyecto/src/
/home/carmen/proyecto/doc/
/home/carmen/proyecto/bin/
/home/carmen/dos_palabras.txt
/home/carmen/.bash_logout
```

```
carmen@servidor:~$ tar -tf **home_con_contenido_enlaces.tar**
home/carmen/
home/carmen/.bashrc
home/carmen/config_files.txt
home/carmen/test_locate.txt
home/carmen/.local/
home/carmen/.local/share/
home/carmen/.local/share/nano/
home/carmen/.bash_history
home/carmen/.sudo_as_admin_successful
home/carmen/.lessht
home/carmen/.ssh/
home/carmen/.ssh/authorized_keys
home/carmen/errors.txt
home/carmen/.cache/
home/carmen/.cache/motd.legal-displayed
home/carmen/mi_bin/
home/carmen/.profile
home/carmen/enlace_a_archivo_tmp.txt
home/carmen/mis_archivos.txt
home/carmen/typescript
home/carmen/proyecto/
home/carmen/proyecto/src/
home/carmen/proyecto/doc/
home/carmen/proyecto/bin/
home/carmen/dos_palabras.txt
home/carmen/.bash_logout
```

8. Escribe un script que reciba una ruta a un archivo. Debe verificar si es un archivo regular, un directorio o si no existe, mostrando un mensaje diferente en cada caso. (Pista: if [-f ...], if [-d ...]).

```
#!/bin/bash

# Verificar si se proporcionó un argumento
if [ -z "$1" ]; then
    echo "Error: Debes proporcionar una ruta como argumento."
    echo "Uso: $0 <ruta/al/archivo_o_directorio>"
    exit 1
fi

RUTA="$1"

# Verificar si es un archivo regular
if [ -f "$RUTA" ]; then
    echo "La ruta '$RUTA' es un ARCHIVO REGULAR."
# Verificar si es un directorio
elif [ -d "$RUTA" ]; then
    echo "La ruta '$RUTA' es un DIRECTORIO."
# Si no cumple las anteriores, no existe
elif [ ! -e "$RUTA" ]; then
    echo "Error: La ruta '$RUTA' NO EXISTE."
# Caso alternativo (podría ser un enlace, un socket, etc.)
else
    echo "La ruta '$RUTA' existe, pero no es un archivo regular ni un directorio (p. ej., un enlace o dispositivo)."
fi
```

```
carmen@carmen-portatil:~$ ./info_user.sh /etc/passwd
La ruta '/etc/passwd' es un ARCHIVO REGULAR.
carmen@carmen-portatil:~$ ./info_user.sh /etc
La ruta '/etc' es un DIRECTORIO.
carmen@carmen-portatil:~$ ./info_user.sh /etc/rutainexistente
Error: La ruta '/etc/rutainexistente' NO EXISTE.
carmen@carmen-portatil:~$
```

9. Crea un script que intente crear un directorio llamado `test_dir` en `/`. Usando el código de salida (`$?`), el script debe informar si tuvo éxito o si falló por un problema de permisos.

```
#!/bin/bash

TARGET_DIR="/test_dir"

# Intentar crear el directorio. El error de permisos será capturado por $?
mkdir "$TARGET_DIR" 2>/dev/null

# Capturar el código de salida del comando mkdir
EXIT_CODE=$?

# Verificar el código de salida
if [ $EXIT_CODE -eq 0 ]; then
    echo "ÉXITO: El directorio '$TARGET_DIR' fue creado correctamente."
    # Limpieza (se necesita sudo para eliminar lo creado sin sudo)
    # rmdir "$TARGET_DIR" 2>/dev/null
elif [ $EXIT_CODE -eq 1 ]; then
    # El código de salida 1 puede ser un error general,
    # pero al intentar crear en / sin sudo, Bash reporta 1.

    # Intentamos una verificación más rigurosa: si el directorio no existe y el error fue 1,
    # es casi seguro que fue un fallo de permisos.
    if [ ! -d "$TARGET_DIR" ]; then
        echo "FALLO: No se pudo crear el directorio '$TARGET_DIR'."
        echo "Causa probable: Permiso denegado. Intenta ejecutar con 'sudo'."
    else
        echo "FALLO: El directorio ya existe o hubo un error inesperado (código $EXIT_CODE)."

```

```
carmen@carmen-portatil:~$ ./info_user.sh
FALLO: No se pudo crear el directorio '/test_dir'.
Causa probable: Permiso denegado. Intenta ejecutar con 'sudo'.
carmen@carmen-portatil:~$ sudo ./info_user.sh
[sudo] contraseña para carmen:
ÉXITO: El directorio '/test_dir' fue creado correctamente.
carmen@carmen-portatil:~$
```

10. Escribe un script que reciba cualquier número de argumentos. El script debe iterar sobre ellos y solo imprimir aquellos que sean números mayores que 10.

```
#!/bin/bash

# Este script itera sobre todos los argumentos y solo imprime aquellos
# que son números enteros mayores que 10.

echo "Argumentos mayores que 10:"

# Usamos 'for ARG in "$@"' para iterar sobre todos los argumentos pasados al script.
for ARG in "$@"; do
    # 1. Verificar si el argumento es un número entero.
    # La expresión '[' "$ARG" =~ ^[0-9]+$ ]' verifica si la cadena consiste solo en dígitos.
    if [ "$ARG" =~ ^[0-9]+$ ]; then
        # 2. Si es un número, verificar si es mayor que 10.
        # Usamos el operador de comparación numérica -gt (greater than).
        if [ "$ARG" -gt 10 ]; then
            echo "$ARG"
        fi
    fi
done

echo "---"
```

carmen@carmen-portatil:~\$./info_user.sh 5 hola 10 11 99 adios 1
Argumentos mayores que 10:
11
99
