**Tabla: Uso de funciones con tuberías y redirecciones en Bash**

| **Caso** | **¿Puede usar tuberías?** | **¿Puede usar redirecciones?** | **Ejemplo** |
| --- | --- | --- | --- |
| **Función que genera salida estándar** | Sí, puede enviar su salida a través de tuberías. | Sí, puede redirigir su salida a un archivo. | `echo "mensaje" |
| **Función que toma entrada estándar** | Sí, puede recibir datos de entrada a través de tuberías. | No aplica, ya que solo toma entrada, no genera salida. | `cat archivo.txt |
| **Función que toma entrada y genera salida** | Sí, puede recibir datos y generar salida para tuberías. | Sí, puede tomar entrada desde un archivo y generar salida a un archivo. | `cat archivo.txt |
| **Función que no genera salida** | No, no puede pasar salida a través de tuberías. | Sí, puede redirigir salida a un archivo (si se usa echo o algo similar). | mkdir nuevo\_directorio (no usa tuberías, solo redirección) |
| **Función que realiza una acción (como cd)** | No, no genera salida ni toma entrada estándar. | No aplica, ya que las acciones como cd no interactúan con tuberías. | cd /home/usuario (sin tuberías ni redirecciones) |
| **Función que procesa texto o archivos** | Sí, puede recibir texto o archivos a través de tuberías. | Sí, puede tomar archivos como entrada o generar salida a un archivo. | `echo "texto" |

**Resumen:**

* **Tuberías**:
  + Las funciones que generan o consumen **salida estándar** pueden trabajar con tuberías.
  + Las funciones que **realizan acciones** (como cambiar directorios o crear archivos) no pueden usarse con tuberías.
* **Redirecciones**:
  + Las funciones que generan **salida estándar** pueden redirigirla a un archivo usando > o >>.
  + Las funciones que **no generan salida** no tienen sentido para redirección de salida.
  + Las funciones que **requieren entrada estándar** pueden recibir datos de archivos mediante < o <<.

### **📋 Copiar y Pegar en Nano**

|  |  |
| --- | --- |
| **Acción** | **Atajo** |
| **Copiar (Cortar en Nano)y borrar** | CTRL + K |
| **Pegar** | CTRL + U |
| **Copiar varias líneas** | ALT + 6 |
| **Seleccionar texto** | CTRL + 6, luego mueve el cursor |

**Resumen de comandos que pueden y no pueden usar tuberías:**

| **Pueden usar tuberías** | **No pueden usar tuberías** |
| --- | --- |
| ps, ls, cat, grep, awk, sed, sort, cut, head, tail, find | cd, exit, mkdir, kill, echo, for, while, exit |

cat – Concatena archivos

sort – Ordena líneas de texto

uniq – Reporta u omite lineas repetidas

grep – Imprime lineas que coincidan con un patrón

wc – Imprime el número de líneas, palabras y bytes para cada archivo

head – Imprime la primera parte de un archivo

tail – Imprime la última parte de un archivo

tee – Lee de la entrada estándar y escribe en la salida estándar y en archivos

type – Indica como se interpreta el nombre de un comando

which – Muestra que programa ejecutable será ejecutado

help – Ofrece ayuda para funciones del shell

man – Muestra el manual de un comando

apropos – Muestra una lista de comandos apropiados

info – Muestra información sobre un comando

whatis – Muestra una descripción muy breve de un comando

alias – Crea un alias para un comando

**FILTROS ALGUNOS COMUNES**

grep,tee,tail,head,wc,uniq

**Expansión de nombres de archivo**

El mecanismo según el cual trabajan los comodines se llama expansión de nombres de archivo.

Echo .[!.]\*

Expansión aritmética

El shell permite realizar aritmética mediante la expansión. Ésto nos permite usar el prompt del shell

como una calculadora:

[me@linuxbox ~]$ echo $((2 + 2))4

echo $(($((5\*\*2)) \* 3))75 se puede anidar

Expansión con llaves

[me@linuxbox ~]$ echo Front-{A,B,C}-Back

Front-A-Back Front-B-Back Front-C-Back

Las expansiones con llaves puede ser anidadas:

[me@linuxbox ~]$ echo a{A{1,2},B{3,4}}b

aA1b aA2b aB3b aB4b

Expansión con parámetros

[me@linuxbox ~]$ echo $USERme

Sustitución de comandos

La sustitución de comandos nos permite usar la salida de un comando como una expansión:

[me@linuxbox ~]$ echo $(ls)

Desktop Documents ls-output.txt Music Pictures Public

Templates

Videos

Permisos

id – Muestra la identidad del usuario

chmod – Cambia el modo de un archivo

umask – Establece los permisos por defecto

su – Ejecuta un shell como otro usuario

sudo – Ejecuta un comando como otro usuario

chown – Cambia el propietario de un archivo

chgrp – Cambia la propiedad de grupo de un archivo

passwd – Cambia la contraseña de un usuario

**Propietarios, miembros del grupo, y todos los demás**

Cuando un usuario posee un archivo o directorio, el usuario tiene control sobre su acceso. Los usuarios pueden,

sucesivamente, pertenecer a un grupo consistente en uno o más usuarios a quienes se le ha dado

acceso a archivos y directorios por sus propietarios. Además de conceder acceso a un grupo, un

propietario puede también conceder algún tipo de derechos de acceso a todo el mundo, a quienes en

el lenguaje de Unix nos referimos como el mundo.

Para encontrar información sobre tu identidad,

usa el comando id: [me@linuxbox ~]$ id

| ****Concepto**** | ****Definición**** |
| --- | --- |
| **/etc/shadow** | Archivo que almacena contraseñas encriptadas de los usuarios. No es accesible por usuarios normales. |
| **Permisos en Unix** | Controlan quién puede leer, escribir o ejecutar un archivo o directorio. |
| **Propietario** | Usuario que posee un archivo o directorio y puede cambiar sus permisos. |
| **Grupo** | Conjunto de usuarios con acceso a ciertos archivos según permisos asignados. |
| **Mundo (others)** | Cualquier usuario que no sea el propietario ni pertenezca al grupo del archivo. |
| **Comando** id | Muestra el UID, GID y los grupos a los que pertenece un usuario. |
| **UID (User ID)** | Número único que identifica a un usuario en el sistema. |
| **GID (Group ID)** | Número único que identifica a un grupo en el sistema. |
| **/etc/passwd** | Archivo que almacena información de usuarios (UID, GID, shell, home). |
| **/etc/group** | Archivo que almacena los grupos del sistema y sus miembros. |
| **Superusuario (root)** | Usuario con UID 0, con control total sobre el sistema. |
| **Modelo de grupos en Linux moderno** | Cada usuario tiene su propio grupo con el mismo nombre, facilitando la asignación de permisos. |
| **unmask** | El comando umask controla los permisos por defecto dados a un archivo cuando éste es creado. |

**umask – Establecer los permisos por defecto**

| ****Concepto**** | ****Explicación**** | ****Sintaxis/Ejemplo**** |
| --- | --- | --- |
| **umask** | Controla los permisos por defecto de nuevos archivos y directorios. | umask valor |
| **Ver umask actual** | Muestra la máscara de permisos activa. | umask |
| **Establecer umask** | Define una nueva máscara de permisos. | umask 0002 |
| **Máscara de permisos** | Define qué permisos se eliminan en la creación de archivos. |  |
| **Ejemplo umask 0000** | No restringe permisos, archivos tendrán rw-rw-rw-. | umask 0000 |
| **Ejemplo umask 0022** | Elimina permisos de escritura para grupo y otros, archivos tendrán rw-r--r--. | umask 0022 |
| **Ejemplo umask 0077** | Solo el propietario tiene acceso total, archivos tendrán rw-------. | umask 0077 |
| **Restablecer umask por defecto** | Vuelve a la máscara estándar del sistema. | umask 0002 |

### **Explicación rápida de cómo funciona umask**

* La máscara define qué permisos se eliminan.
* Se usa en **notación octal** (cada dígito representa **usuario, grupo y otros**).
* Donde haya un **1 en binario**, se desactiva el permiso correspondiente.

### **Resumen del proceso**

1️⃣ **Los permisos por defecto** cuando creas archivos o directorios son:

* **Archivos:** 666 (rw-rw-rw-) → Lectura y escritura para todos.
* **Directorios:** 777 (rwxrwxrwx) → Todos pueden leer, escribir y ejecutar.

2️⃣ **La** umask **resta permisos** antes de que el archivo/directorio sea creado.

* Ejemplo: si umask es 0022, entonces:
  + Archivos: 666 - 022 = 644 (rw-r--r--)
  + Directorios: 777 - 022 = 755 (rwxr-xr-x)

**Cambiar identidades**

### **📌 Cambio de identidad en Linux**

| ****Método**** | ****Descripción**** | ****Sintaxis**** | ****Ejemplo**** |
| --- | --- | --- | --- |
| su | Cambia de usuario y abre un nuevo shell. Requiere la contraseña del usuario al que queremos cambiar. | su [usuario] | su - usuario2 → Cambia a "usuario2" |
| su - | Cambia a otro usuario cargando su entorno y home. | su - (por defecto cambia a root) | su - root → Cambia a superusuario |
| su -c 'comando' | Ejecuta un solo comando como otro usuario. | su -c 'comando' | su -c 'ls /root' |
| exit | Sale del usuario actual y vuelve al anterior. | exit | exit (para salir del usuario root) |
| sudo | Ejecuta un comando como otro usuario sin abrir un shell nuevo. Usa la propia contraseña del usuario. | sudo comando | sudo apt update |
| sudo -l | Muestra los permisos que tenemos con sudo. | sudo -l | sudo -l → Lista los comandos permitidos |
| sudo -u usuario comando | Ejecuta un comando como otro usuario. | sudo -u usuario comando | sudo -u usuario2 whoami |

### **📌 Diferencias entre** su **y** sudo

| ****Característica**** | su | sudo |
| --- | --- | --- |
| Abre un nuevo shell | ✅ Sí | ❌ No |
| Requiere contraseña del otro usuario | ✅ Sí | ❌ No (usa la propia contraseña) |
| Se usa para cambiar a otro usuario completamente | ✅ Sí | ❌ No (ejecuta comandos puntuales) |
| Puede restringirse a comandos específicos | ❌ No | ✅ Sí (configurado en /etc/sudoers) |

### **📌 Ejemplo práctico**

1️⃣ **Convertirse en root** (modo interactivo):

su -

2️⃣ **Ejecutar un solo comando como root**:

su -c 'ls /root'

3️⃣ **Ejecutar un comando como root sin cambiar de sesión**:

sudo apt update

4️⃣ **Ejecutar un comando como otro usuario**:

sudo -u usuario2 whoami

5️⃣ **Ver qué comandos podemos usar con** sudo:

sudo -l

**chown – Cambia el propietario y el grupo de un archivo**

El comando chown se utiliza para cambiar el propietario y el grupo del propietario de un archivo o

un directorio. Se requieren privilegios de superusuario para utilizar este comando. La sintaxis de

chown tiene este aspecto:

**chown [propietario][:[grupo]] archivo...**

**chgrp – Cambiando el grupo del propietario**

**Exacto. Si un usuario no está en el grupo sudo, no podrá usar sudo.**

## **👤 Cómo agregar un usuario al grupo** sudo

Si quieres darle permisos de sudo a un usuario (por ejemplo, practicante), usa el siguiente comando como **root** o con otro usuario que tenga permisos de sudo:

sudo usermod -aG sudo practicante

🔹 -aG → **Añadir (**-a**) al grupo (**-G**)** sin eliminar otros grupos del usuario.  
🔹 sudo → **Grupo de privilegios** para ejecutar comandos como root.

## **🔄 Aplicar cambios**

Para que los cambios tengan efecto, el usuario debe cerrar sesión y volver a iniciar sesión. O bien, puedes forzar la actualización con:

su - practicante

## **✅ Verificar si el usuario tiene permisos de** sudo

Después de agregarlo, comprueba sus permisos ejecutando:

sudo -l

Si el usuario **tiene permisos**, verá una lista de los comandos que puede ejecutar con sudo.

Si **no tiene permisos**, aparecerá:

practicante is not in the sudoers file. This incident will be reported.

Esto significa que el usuario aún no puede usar sudo.

## **🛑 Cómo eliminar a un usuario del grupo** sudo

Si quieres quitarle permisos de sudo a un usuario, usa:

sudo deluser practicante sudo

### **📌 Resumen rápido**

| Acción | Comando |
| --- | --- |
| Añadir usuario a sudo | sudo usermod -aG sudo usuario |
| Aplicar cambios sin cerrar sesión | su - usuario o newgrp sudo |
| Verificar si tiene sudo | sudo -l |
| Quitar usuario del grupo sudo | sudo deluser usuario sudo |

Si necesitas más ayuda, dime qué estás intentando hacer. 🚀

| Comando | Requiere contraseña de | Carga entorno de root | Se usa cuando… |
| --- | --- | --- | --- |
| sudo su | Tu usuario | No completamente | No tienes la contraseña de root pero tienes privilegios de sudo,desde tu home  Te conviertes en root usando sudo, lo que significa que necesitas estar autorizado en el archivo /etc/sudoers |
| su - | Root | Sí, completamente | Quieres actuar como root con su entorno y configuraciones |

Si quieres **usar root de manera más segura**, lo mejor es configurar sudo para ejecutar solo los comandos necesarios en vez de usar su directamente. 🚀

**La sintaxis ALL=(ALL:ALL) ALL proviene del archivo /etc/sudoers, que define qué permisos tienen los usuarios para ejecutar comandos con sudo.**

### 📌 Explicación de ALL=(ALL:ALL) ALL

usuario HOSTS=(USUARIOS:GRUPOS) COMANDOS

| Elemento | Descripción |
| --- | --- |
| ALL | Se aplica en todos los hosts (útil en sistemas con múltiples servidores). |
| (ALL:ALL) | El primer ALL significa que el usuario puede ejecutar comandos como **cualquier usuario**. |
|  | El segundo ALL significa que el usuario puede ejecutar comandos como **cualquier grupo**. |
| ALL | Puede ejecutar **cualquier comando** en el sistema. |

✅ **Ejemplo en** /etc/sudoers

bryanzavala ALL=(ALL:ALL) ALL

👉 El usuario bryanzavala puede ejecutar **cualquier comando** como cualquier usuario y grupo en cualquier máquina.

## 🔄 Variaciones de la sintaxis

| Configuración | Explicación |
| --- | --- |
| usuario ALL=(ALL) ALL | El usuario puede ejecutar **cualquier comando** como **cualquier usuario**, pero no cambia el grupo. |
| usuario ALL=(ALL:ALL) /bin/ls, /usr/bin/cat | Puede ejecutar solo /bin/ls y /usr/bin/cat como cualquier usuario y grupo. |
| usuario ALL=(root) ALL | Solo puede ejecutar comandos como **root**, no como otros usuarios. |
| usuario ALL=(ALL) NOPASSWD: ALL | Puede ejecutar cualquier comando como cualquier usuario **sin que le pida contraseña**. |
| usuario ALL=(ALL) NOPASSWD: /bin/systemctl restart apache2 | Puede reiniciar apache2 sin pedir contraseña, pero necesita sudo para otros comandos. |
| %grupo ALL=(ALL) ALL | Todos los usuarios en grupo pueden ejecutar cualquier comando como cualquier usuario. |

practicante ALL=(ALL) /usr/bin/ls, /usr/bin/cat

✔ Solo podrá ejecutar ls y cat, **pero no otros comandos** con sudo.

## **¿Qué significa** NOPASSWD**?**

🔹 Normalmente, cuando usas sudo, el sistema **te pide la contraseña**.  
🔹 Si agregas NOPASSWD:, puedes ejecutar ese comando **sin escribir la contraseña**.

practicante ALL=(ALL) NOPASSWD: /usr/bin/ls

### **🛠 Resumen**

| Comando | Descripción |
| --- | --- |
| ls -l archivo | Ver permisos de un archivo y su grupo propietario. |
| groups usuario | Ver a qué grupos pertenece un usuario. |
| id usuario | Ver el UID, GID y grupos de un usuario. |
| namei -l archivo | Ver permisos de cada directorio en la ruta de un archivo. |
| getent group grupo | Ver los miembros de un grupo. |
| ls -ld directorio | Ver permisos de un directorio. |

**Exacto. Si un usuario no está en el grupo** sudo**, no podrá usar sudo**

| Acción | Comando |
| --- | --- |
| Añadir usuario a sudo | sudo usermod -aG sudo usuario |
| Aplicar cambios sin cerrar sesión | su - usuario o newgrp sudo |
| Verificar si tiene sudo | sudo -l |
| Quitar usuario del grupo sudo | sudo deluser usuario sudo |

**Aquí tienes los comandos en forma de tabla:**

| Acción | Comando |
| --- | --- |
| **Crear usuario "practicante" con home** | sudo useradd -m practicante |
| **Crear usuario "invitado" con home** | sudo useradd -m invitado |
| **Crear grupo "desarrolladores"** | sudo groupadd desarrolladores |
| **Crear grupo "admin\_local"** | sudo groupadd admin\_local |
| **Añadir "practicante" al grupo "desarrolladores"** | sudo usermod -aG desarrolladores practicante |
| **Verificar los grupos de "practicante"** | groups practicante |
| **Ver lsita de grupos** | cat/etc/group |
| **Ver lista de usuarios** | cat/etc/passwd  nombre\_usuario:x:UID:GID:información\_comentario:directorio\_home:shell\_por\_defecto |

**Para asegurarte de eliminar completamente un usuario y su grupo en Linux, sigue estos pasos:**

### **1️⃣ Eliminar un usuario y su directorio home**

sudo userdel -r nombre\_usuario

📌 **La opción** -r borra su directorio /home/nombre\_usuario y su correo en /var/mail/nombre\_usuario.

### **2️⃣ Eliminar el grupo asociado**

sudo groupdel nombre\_grupo

🔹 **Ejemplo**:

sudo groupdel desarrolladores

📌 Si el grupo es primario de un usuario que aún existe, primero cambia su grupo principal.

Aquí tienes los comandos y sus opciones más comunes en formato tabla:

| **Comando** | **Definición** | **Opciones más comunes** |
| --- | --- | --- |
| ps | Muestra los procesos en ejecución | ps - Muestra los procesos actuales del usuario.  ps -e - Muestra todos los procesos.  ps -f - Muestra los procesos con más detalles.  ps aux - Muestra todos los procesos con información completa. |
| ls | Muestra el contenido de un directorio | ls - Lista archivos y carpetas.  ls -l - Muestra detalles (permisos, tamaño).  -F --classify Esta opción añadirá un carácter indicador al final de cada nombre listado. Por ejemplo, una “/” es que el nombre es un directorio.  -r --reverse Muestra los resultados en orden inverso. Normalmente, ls muestra los resultados en orden alfabético ascendente.  -s Ordena los resultados por tamaño, -t Ordena por hora de modificación.  ls -a - Muestra todos los archivos, incluidos los ocultos.  ls -lh - Muestra un formato legible para humanos.  ls -R - Muestra recursivamente los archivos de subdirectorios. |
| cat | Muestra el contenido de un archivo | cat archivo.txt - Muestra el contenido de un archivo. cat archivo1.txt archivo2.txt - Muestra múltiples archivos.  `cat archivo.txt |
| echo | Imprime texto o variables en la salida estándar | echo "Hola Mundo" - Imprime texto.  echo $VAR - Imprime el valor de una variable.  echo -n "Texto" - Imprime sin salto de línea. |
| grep | Filtra texto buscando patrones específicos  **Con grep**: Siempre es recomendable usar -E y **comillas dobles**. Esto te permite trabajar con expresiones regulares extendidas sin problemas y asegura que las variables se expandan correctamente. | grep 'patrón' archivo.txt - Busca en un archivo.  grep -r 'patrón' directorio/ - Búsqueda recursiva en directorios.  grep -i 'patrón' archivo.txt - Ignora mayúsculas/minúsculas.  grep -v 'patrón' archivo.txt - Muestra líneas sin el patrón.  grep -l 'patrón' \* - Muestra archivos que contienen el patrón.  **grep -E "pattern" archivo Utiliza expresiones regulares extendidas**  **grep -w** "word" file Coincide solo con **palabras completas** (no subcadenas). |
| awk | Procesa y manipula texto en líneas o columnas | awk '{print $1}' archivo.txt - Muestra la primera columna.  awk '{print $1, $3}' archivo.txt - Muestra la primera y tercera columna.  awk '/patrón/ {print $1}' archivo.txt - Muestra la primera columna de las líneas que contienen el patrón.  awk '{printf "Producto: %s | Precio: %d | Cantidad: %d\n", $1, $2, $3}' ventas.txt  Awk -F ‘delimitador’ '{print $3}' datos.txt |
|  |  |  |
| sed | Realiza sustituciones o ediciones en texto | sed 's/patrón/reemplazo/' archivo.txt - Reemplaza el primer patrón.  sed 's/patrón/reemplazo/g' archivo.txt - Reemplaza todos los patrones.  sed '1,5d' archivo.txt - Elimina las líneas 1 a 5. |
| sort | Ordena las líneas de texto | sort archivo.txt - Ordena las líneas de un archivo.  sort -r archivo.txt - Ordena de forma descendente.  sort -n archivo.txt - Ordena numéricamente.  **sort -g**: Ordena correctamente los números, incluso si tienen diferentes dígitos. |
| cut | Extrae partes específicas de una línea de texto | cut -d' ' -f1 archivo.txt - Extrae la primera columna.  cut -c1-5 archivo.txt - Extrae los primeros 5 caracteres. |
| head | Muestra las primeras N líneas de un archivo o entrada | head archivo.txt - Muestra las primeras 10 líneas. head -n 20 archivo.txt - Muestra las primeras 20 líneas. |
| tail | Muestra las últimas N líneas de un archivo o entrada | tail archivo.txt - Muestra las últimas 10 líneas.  tail -n 20 archivo.txt - Muestra las últimas 20 líneas.  tail -f archivo.txt - Muestra las líneas que se añaden en tiempo real. |
| find | Busca archivos y directorios en el sistema | find /ruta -name 'archivo.txt' - Busca un archivo por nombre.  find /ruta -type d - Busca directorios.  find /ruta -type f -name '\*.txt' - Busca ficheros tipo archivo con el patron .txt. (aqi combinas ) |
| cd | Cambia el directorio de trabajo | cd /ruta/del/directorio - Cambia a un directorio específico.  cd .. - Sube un nivel en el directorio.  cd ~ - Cambia al directorio home del usuario. |
| mkdir | Crea un nuevo directorio | mkdir nuevo\_directorio - Crea un directorio.  mkdir -p ruta/del/nuevo\_directorio - Crea directorios padres si no existen. |
| rm | Elimina archivos o directorios | rm archivo.txt - Elimina un archivo.  rm -r directorio/ - Elimina un directorio recursivamente.  rm -f archivo.txt - Fuerza la eliminación sin confirmación. |
| touch | Crea un archivo vacío o actualiza la fecha de modificación | touch archivo.txt - Crea un archivo vacío.  touch -t 202502030830 archivo.txt - Establece una fecha específica en el archivo. |
| chmod | Cambia los permisos de acceso a archivos y directorios | chmod 755 archivo.txt - Establece permisos de lectura, escritura y ejecución.  chmod +x archivo.sh - Da permisos de ejecución a un archivo. |
| cp | Copia archivos o directorios | cp archivo.txt copia.txt - Copia un archivo.  cp -r directorio/ copia\_directorio/ - Copia un directorio recursivamente.  cp -t <directorio\_destino> <archivo1> <archivo2> se utiliza para especificar el **directorio de destino** de la copia, **antes de los archivos**. |
| mv | Mueve o renombra archivos o directorios | mv archivo.txt nuevo\_archivo.txt - Renombra un archivo.  mv archivo.txt /ruta/del/directorio/ - Mueve un archivo a otro directorio. |
| kill | Envía una señal a un proceso (usualmente para terminarlo) | kill PID - Mata un proceso con un PID específico.  kill -9 PID - Mata un proceso de forma forzada. |
| man | Muestra el manual de un comando | man comando - Muestra el manual de un comando (ej. man ls). |
| **wc** | Te hace un recuento de las palabras,lineas o caracteres que tiene un archivo de texto. | Wc -l ficheroi cuenta lineas  Wc -m fichero caracteres  Wc -w fichero palabras |
| **write** | Envia mensajes a la terminal de un usuario | Write usuario terminasi tienes muchas mensaje |
| **tr** | El comando tr se usa para traducir o eliminar caracteres en un flujo de texto. Puede hacer sustituciones, eliminar caracteres o reemplazar texto.  tr [opciones] SET1 [SET2]  Convierte a líneas, ordena, y vuelve a unir con ','  echo "$lista" | tr ',' '\n' | sort -n | paste -sd ","  lista="$lista,$num"  echo "$lista" > lista  # Guarda la lista en un solo string separado por comas | -d (elimina caracteres), echo "abc123" | tr -d 'a-z'  -s (comprime secuencias),  -c (complementa el conjunto),  -t (truncar la longitud de SET1 a la de SET2),  -C (modificar con caracteres complementarios).  **Reemplazar caracteres**: echo "abc123" | tr 'a-z' 'A-Z' |
| **paste** | El comando paste se usa para combinar líneas de archivos o datos de entrada, separándolos con tabuladores o un delimitador especificado.  paste [opciones] archivo1 archivo2 … | -d (delimitador personalizado)---paste -d ',' archivo1.txt archivo2.txt  --Combina todas las líneas de archivo1.txt en una única línea, separada por tabuladores.-->  -s (combinación secuencial de columnas),--paste -s archivo1.txt  -z (terminación de línea a cero). |
| **type** | Muestra información sobre un comando en Bash.  type [opciones] comando | -t → Devuelve el tipo (alias, builtin, file, etc.).  -a → Muestra todas las definiciones posibles.  -p → Muestra la ruta del ejecutable si existe.  -f → Ignora alias y funciones, mostrando solo ejecutables. |
| **file** | Identifica el tipo de un archivo analizando su contenido.  file [opciones] archivo | -b → Muestra solo el tipo sin el nombre del archivo.  -i → Devuelve el tipo MIME (text/plain, image/png).  -L → Sigue enlaces simbólicos.  -E → Explica cómo se determinó el tipo.  -s → Examina archivos vacíos o especiales. |
| **less** | El comando less es un programa para ver archivos de texto | **less *nombre\_del\_archivo***  ***q Cierra less*** |
|  |  |  |
|  |  |  |

**Otros comandos**

| **Comando** | **Descripción** |
| --- | --- |
| **alias** | Te permite definir alias temporales en tu sesión de la consola, pudiendo sustituir una palabra por una serie de comandos. El parámetro a usar es alias NAME="VALUE". |
| **unalias** | Te sirve para quitar el alias que hayas configurado, escribiendo el comando seguido del alias que hayas creado. |
| **htop** | Te permite ver en el terminal los procesos que tienes en funcionamiento para gestionar los recursos de tu ordenador. |
| **ps** | Con este comando puedes visualizar los procesos que se ejecutan en tu sesión del terminal. Te muestra un identificador de cada proceso e información útil sobre él. |
| **ping** | Te dice tu dirección IP, o la de una web que escribas después. |
| **echo** | Te permite mostrar un texto definido por ti en el terminal, escribiendo el comando seguido del mensaje entre comillas. |
| **passwd** | Te permite cambiar la contraseña de tu cuenta de usuario. Al escribir solo el comando, te dirá primero que escribas tu contraseña actual y luego la nueva. |
| **whoami** | Te muestra el nombre de usuario que tienes ahora mismo, por si no recuerdas en qué sesión estás. |
| **uname** | Te muestra la información sobre la distribución de GNU/Linux que estás usando y su versión, además de otra información útil del sistema operativo. Sobre todo puedes usarlo con uname -a para mostrar toda la información. |
| **neofetch** | Te muestra la información sobre tu sistema, la distribución que usas, la versión del kernel, del shell y de tu hardware. Todo ello con un logotipo ASCII de tu distro. |
| **whatis** | Este comando seguido de otro comando o el nombre de un programa te muestra su descripción, para cuando no sabes qué hace un comando o qué es un programa. |
| **exit** | Cierra la sesión y cierra el terminal. |
| **shutdown** | Te permite apagar el ordenador desde el terminal. Puedes usarlo junto a now, con el comando shutdown now para apagar ahora mismo, o cambiar now por una fecha para programar el apagado. |

|  |  |  |
| --- | --- | --- |
| **Situación** | **Uso de Comillas Dobles** | **Explicación** |
| **Espacios en nombres de archivo o variables** | **Sí** | Usar comillas dobles para que el nombre de archivo o variable se trate como un solo elemento. |
| **Nombres de archivos sin espacios** | **No** | No es necesario usar comillas dobles. Bash manejará correctamente los nombres sin espacios. |
| **Patrones de expansión (\*.$ex) en bucles** | **No** | No uses comillas dobles, ya que el patrón debe expandirse a los archivos que coinciden con la expresión. |
| **Uso de variables con listas o archivos con espacios** | **Sí** | Si la variable contiene varios elementos con espacios, usa comillas dobles para evitar la separación. |
| **Expresiones regulares (ERE) con espacios** | **Sí** | Usar comillas dobles para evitar que los espacios se interpreten incorrectamente, y mantener la expresión regular completa. |
| **Uso de grep con ERE o patrones complejos** | **Sí** | Las comillas dobles permiten pasar toda la expresión regular como un solo argumento. |

### 

| **Categoría** | **Operador** | **Descripción** | **Ejemplo** |
| --- | --- | --- | --- |
| **1. Operadores Aritméticos (Números)**  **si uso (())** | + | Suma | $(($a + $b)) → 7 |
|  | - | Resta | $(($a - $b)) → 3 |
|  | \* | Multiplicación | $(($a \* $b)) → 10 |
|  | / | División entera | $(($a / $b)) → 2 |
|  | % | Módulo (resto) | $(($a % $b)) → 1 |
|  | \*\* | Exponente | $(($a \*\* $b)) → 25 |
| **2. Operadores de Comparación (Números) si uso [[]]** | -eq | Igual a (==) | [ "$a" -eq "$b" ] → ❌ |
|  | -ne | No igual a (!=) | [ "$a" -ne "$b" ] → ✅ |
|  | -gt | Mayor que (>) | [ "$a" -gt "$b" ] → ✅ |
|  | -lt | Menor que (<) | [ "$a" -lt "$b" ] → ❌ |
|  | -ge | Mayor o igual (>=) | [ "$a" -ge "$b" ] → ✅ |
|  | -le | Menor o igual (<=) | [ "$a" -le "$b" ] → ❌ |
| **3. Operadores de Comparación (Cadenas)[[]]** | = | Igualdad (==) | [ "$x" = "$y" ] → ❌ |
|  | != | Diferente | [ "$x" != "$y" ] → ✅ |
|  | -z | Cadena vacía | [ -z "$x" ] → ❌ |
|  | -n | Cadena no vacía | [ -n "$x" ] → ✅ |
| **4. Operadores Lógicos** | && | AND (ambas deben ser verdaderas) | [ "$a" -gt 0 ] && [ "$b" -gt 0 ] |
|  | ! | NOT (negación) | ! [ "$a" -lt 0 ] |
| **5. Operadores de Archivos** | -f | Es un archivo | [ -f "$archivo" ] |
|  | -d | Es un directorio | [ -d "$archivo" ] |
|  | -r | Tiene permiso de lectura | [ -r "$archivo" ] |
|  | -w | Tiene permiso de escritura | [ -w "$archivo" ] |
|  | -x | Es ejecutable | [ -x "$archivo" ] |
|  | -s | No está vacío | [ -s "$archivo" ] |
| **6. Operadores de Redirección** | > | Redirige salida a un archivo (sobrescribe) | echo "Hola" > archivo.txt |
|  | >> | Redirige salida a un archivo (añade) | echo "Hola" >> archivo.txt |
|  | < | Redirige entrada de un archivo | cat < archivo.txt |
|  | 2> | Redirige errores a un archivo | ls no\_existe 2> error.log |
|  | &> | Redirige salida y errores | comando &> salida.log |
| **7. Operadores de Procesos** | & | Ejecutar en segundo plano | comando & |
|  | ; | Ejecutar múltiples comandos | echo "Hola"; ls |
|  | && | Ejecutar el siguiente si el anterior tuvo éxito | mkdir test && cd test |

### **Tabla: Metacaracteres con grep usando comillas dobles y -E**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metacaracter** | **Con comillas dobles** | **¿Funciona sin -E?** | **¿Funciona con -E?** | **Explicación** |
| . | Sí | Sí | Sí | Coincide con **cualquier carácter** (excepto saltos de línea). No requiere -E. |
| \* | Sí | Sí | Sí | Coincide con **cero o más repeticiones** del carácter anterior. No requiere -E. |
| + | No | No (requiere escape) | Sí | Solo funciona con -E para **uno o más** del carácter anterior. |
| ? | No | No (requiere escape) | Sí | Solo funciona con -E para **cero o una** repetición del carácter anterior. |
| ` | ` | No | No (requiere escape) | Sí |
| () | No | No (requiere escape) | Sí | Solo funciona con -E para **agrupar expresiones**. |
| [] | Sí | Sí | Sí | Coincide con **cualquier carácter dentro de los corchetes**. No necesita -E. |
| ^ | Sí | Sí | Sí | Coincide con el **inicio de la línea**. No necesita -E. |
| $ | Sí | Sí | Sí | Coincide con el **final de la línea**. No necesita -E. |

| **Carácter** | **Descripción** |
| --- | --- |
| \* | **Como comodín en nombres de archivos y usando en Case**  **Si lo usas en case o [[ ... == ... ]] → Es un comodín de Bash**   * El \* representa cualquier cantidad de caracteres (incluyendo ninguno). * Se puede usar solo o combinado con otros caracteres.   ls \*.txt # Muestra todos los archivos que terminan en .txt  ls file\* # Muestra todos los archivos que comienzan con "file"  ls \* # Muestra todos los archivos del directorio |
| \* | **En expresiones regulares con grep, sed, etc.** **Si lo usas en expresiones regulares (con grep, sed, awk) → No es un comodín, sino un cuantificador**  * En este caso, \* significa "cero o más veces el carácter anterior". * Siempre debe haber algo antes.   echo "aaa" | grep "a\*" # Coincide porque "a\*" significa "cero o más a" |
| ? | Coincide con un solo carácter. Por ejemplo,  ?.txt coincide con a.txt, b.txt, etc., pero no con ab.txt. |
| [ ] | Se utiliza para definir un conjunto de caracteres. Por ejemplo, [abc].txt coincide con a.txt, b.txt y c.txt. |
| - | El guion se utiliza para introducir opciones o parámetros en un comando. Es una convención común en muchos comandos de Linux. |
| **.** | **Representa el directorio actual, padre o archivos ocultos.**  **Directorio Actual**: Un solo punto (.) se refiere al directorio actual.   * Ejemplo: ./script.sh ejecuta un script en el directorio actual.   **Directorio Padre**: Dos puntos (..) se refieren al directorio padre, un nivel hacia arriba en la jerarquía de directorios.   * Ejemplo: cd .. cambia al directorio padre.   **Archivos Ocultos**: Los nombres de archivos y directorios que comienzan con un punto (.) se consideran archivos ocultos.  **./** Facilita la referencia al directorio actual o a archivos ocultos. **./carpeta** |
| { } | Se utiliza para definir una lista de opciones. Por ejemplo, {a,b,c}.txt coincide con a.txt, b.txt y c.txt. |
| \ | Se utiliza para escapar caracteres especiales, tratándolos como caracteres literales. Por ejemplo, \\* trata \* como un carácter normal. |
| & | Ejecuta el comando en segundo plano. Por ejemplo, comando & ejecuta comando en segundo plano. |
| ` | `Se utiliza para conectar dos comandos, pasando la salida del primer comando como entrada al segundo. Por ejemplo, `comando1 comando2`. |
| > | Redirige la salida de un comando a un archivo, sobrescribiendo el archivo si ya existe. Por ejemplo, comando > archivo.txt. |
| >> | Redirige la salida de un comando a un archivo, añadiendo la salida al final del archivo si ya existe. Por ejemplo, comando >> archivo.txt. |
| < | Redirige la entrada de un archivo a un comando. Por ejemplo, comando < archivo.txt. |
| ~ | Representa el directorio personal del usuario. Por ejemplo, ~/documentos se refiere a /home/usuario/documentos. |
| && | Ejecuta el segundo comando solo si el primer comando es exitoso. Por ejemplo,  comando1 && comando2. |
| ` | Ejecuta el segundo comando solo si el primer comando falla. Por ejemplo,  `comando1 comando2`. |
| ! | Ejecuta el último comando en el historial que coincide con el patrón. Por ejemplo,  !comando. |
| # | Se utiliza para añadir comentarios en scripts o comandos, ignorando el texto que sigue. Por ejemplo, # Esto |
| ^ | **Inicio de Línea**En expresiones regulares, el carácter ^ se utiliza para representar el inicio de una línea. Por ejemplo, ^**abc coincide con cualquier línea que comience con "abc"**  .**Negación en Conjuntos** Dentro de corchetes [], el carácter ^ se utiliza para indicar negación. Por ejemplo**, [^0-9] coincide con cualquier carácter que no sea un dígito**. |

**Conceptos Esenciales**

|  |  |
| --- | --- |
| **Concepto** | **Descripción** |
| **Ruta Absoluta** | **Especifica la ubicación completa de un archivo o directorio desde la raíz del sistema de archivos. Ejemplo: /home/usuario/documento.txt.** |
| **Ruta Relativa** | **Especifica la ubicación de un archivo o directorio en relación al directorio actual. Ejemplo: ../documentos/documento.txt.**  **…—>PARA IR SUBIEDO A PADRES../../** |
| **Permisos de Archivos** | **Lectura (r): Permite ver el contenido del archivo.<br>Escritura (w): Permite modificar el contenido del archivo.<br>Ejecución (x): Permite ejecutar el archivo como un programa.<br>Los permisos pueden establecerse para el propietario del archivo, el grupo y otros usuarios. Ejemplo: chmod 755 archivo.sh.** |
| **Propietario y Grupo** | **Propietario: El usuario que ha creado el archivo o directorio.<br>Grupo: Un conjunto de usuarios que pueden compartir permisos sobre archivos y directorios. Comando: chown usuario:grupo archivo.** |
| **Enlace Simbólico (Symlink)** | **Un archivo que apunta a otro archivo o directorio. Similar a un atajo en Windows. Ejemplo: ln -s destino enlace.** |
| **Enlace Duro (Hard Link)** | **Un segundo nombre para un archivo existente. Ambos nombres apuntan al mismo contenido en el disco. Ejemplo: ln archivo enlace.** |
| **Sistema de Archivos Montado** | **Montaje: Proceso de hacer que un sistema de archivos esté disponible para su uso en un punto de montaje del sistema. Ejemplo: mount /dev/sda1 /mnt.<br>Desmontaje: Proceso de retirar un sistema de archivos del sistema. Ejemplo: umount /mnt.** |

**Comandos Básicos**

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **clear** | Este comando limpia el terminal y deja la ventana en blanco eliminando todo lo que se había hecho y escrito antes. |
| **sudo** | Es la abreviatura de "superuser do" y es un comando que fuerza una acción como si fueras un superusuario. Esto quiere decir que harás acciones como administrador, con tareas que solo pueden realizarse si tienes credenciales de superusuario, como instalar aplicaciones, modificar archivos o configuraciones. Va antes del resto de comandos de la acción. |
| **history** | Este comando muestra una lista con el historial de comandos que has utilizado en el terminal durante la última sesión. |

| **Parámetro** | **Descripción** |
| --- | --- |
| $0 | El nombre del script o comando ejecutado. |
| $1, $2, ... | Argumentos posicionales pasados al script. $1 es el primer argumento, $2 el segundo, y así sucesivamente. |
| $# | Número de argumentos pasados al script o función. |
| $@ | Todos los argumentos pasados al script o función como una lista (cada argumento se mantiene como una cadena separada). |
| $\* | Todos los argumentos pasados al script o función como una única cadena (concatenados). |
| $$ | El PID (ID del proceso) del script o comando en ejecución. |
| $! | El PID del último proceso ejecutado en segundo plano. |
| $? | El código de salida del último comando ejecutado (0 si fue exitoso, un valor distinto de 0 si hubo un error). |
| $- | Los indicadores de las opciones del shell actual (qué opciones están activadas). |
| $\_ | El último argumento del último comando ejecutado. |

tr,xargs cp parmetor mv parametros regex