Trabajo Práctico N°3

**Componentes de GNU: Coreutils y GCC**

**Objetivos:** Utilizar herramientas fundamentales de GNU, como **Coreutils**, **GCC**, y **GDB**, para gestionar archivos, compilar programas y depurar código, consolidando conocimientos sobre la estructura y el funcionamiento del sistema operativo GNU/Linux.

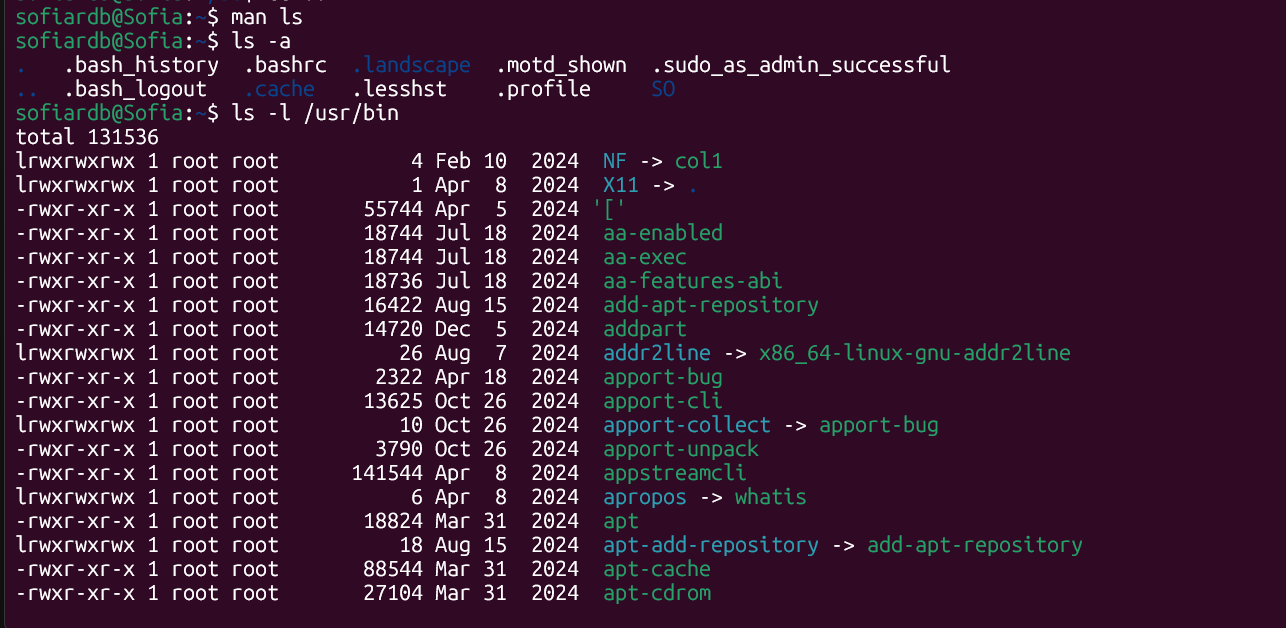
# Comandos de Coreutils.

Descarga los archivos de ejemplo del práctico en el directorio por defecto. Abre una terminal de Debian para ejecutar los comandos pertenecientes a Coreutils. Para ver la ayuda de cada uno utiliza la opción

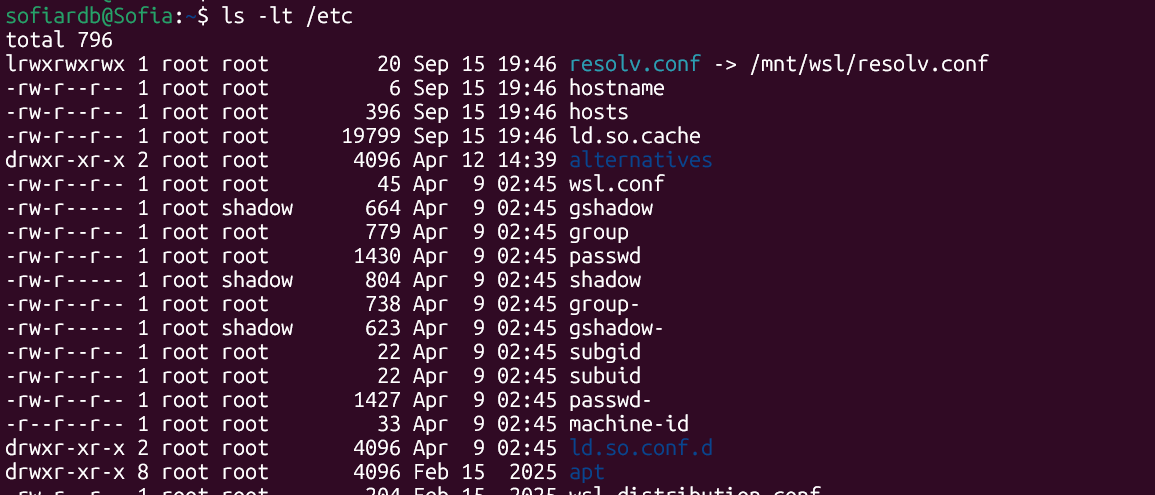
*--help.*

Comando *ls* (listar contenido de directorios):

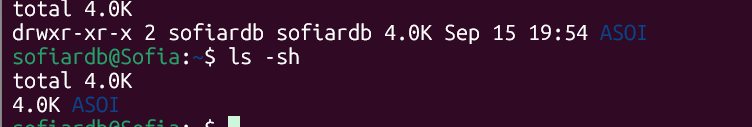
* 1. Lista todos los archivos y directorios del directorio actual, incluyendo los archivos ocultos.
  2. Muestra el contenido del directorio */usr/bin* en formato largo, con detalles como permisos, tamaño y fecha de modificación.



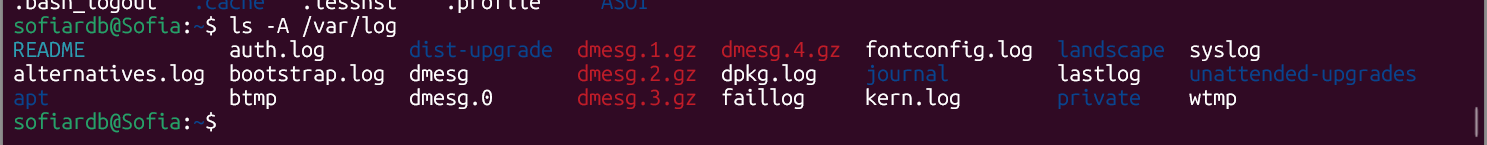
* 1. Muestra el contenido del directorio */etc* en formato largo y ordenado por fecha de modificación.



* 1. Lista los archivos del directorio actual y muestra el tamaño de los archivos en formato legible (KB, MB).



* 1. Lista el contenido del directorio */var/log* sin mostrar archivos ocultos.



Comando *mkdir* (crear directorios) y comando *cd* (cambiar de directorio):

* 1. ¿Cuál es el directorio de trabajo por defecto en GNU/Linux? ¿Qué símbolo utiliza GNU/Linux para denominarlo en los comandos?

El directorio de trabajo por defecto es el directorio del usuario logueado. Se lo representa con el símbolo ~.

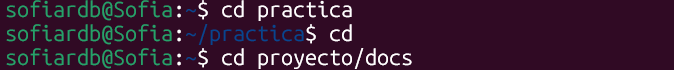
* 1. Crea un directorio llamado *practica* en el directorio personal del usuario.
  2. Cambia al directorio *practica* y muestra el camino completo desde la raíz, usando el comando *pwd*.
  3. Navega al directorio padre del actual.



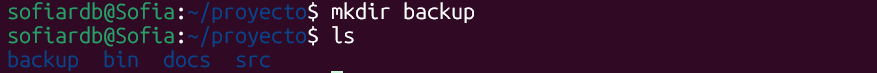
* 1. Crea una estructura de directorios donde *proyecto* sea el directorio raíz, y dentro de él haya subdirectorios llamados *src*, *bin*, y *docs*.



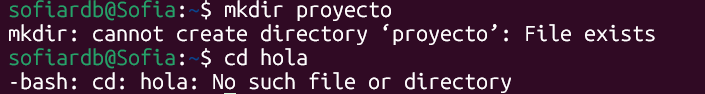
* 1. Desde el directorio por defecto, cambia al directorio *docs* dentro de *proyecto* en un solo comando.



* 1. Crea un directorio llamado *backup* dentro del directorio *proyecto* y verifica su existencia utilizando *ls*.



* 1. Cambia entre el directorio actual y el directorio anterior varias veces.
  2. Intenta crear un directorio que ya existe.
  3. Intenta cambiar a un directorio que no existe y observa el mensaje de error.

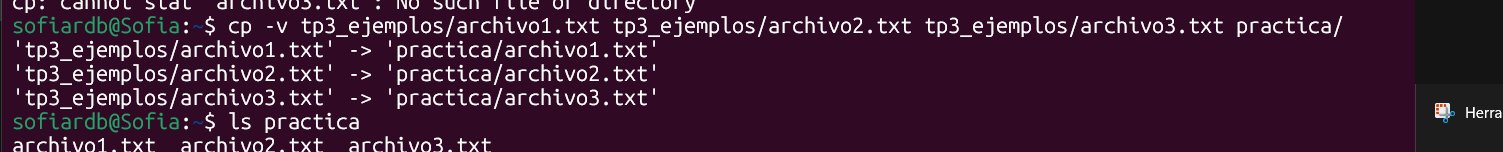


Comando *cp* (copiar archivos):

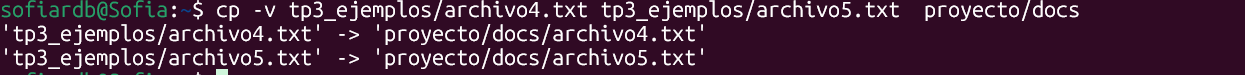
* 1. Copia el archivo */etc/hosts* al directorio personal.



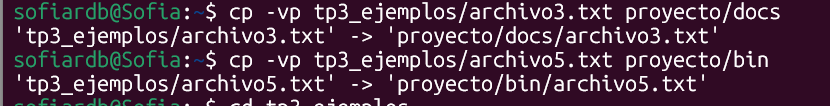
* 1. Copia los archivos *archivo1.txt* al *archivo3.txt* a la carpeta *practica* (creada anteriormente).



* 1. Copia los archivos *archivo4.txt* y *archivo5.txt* a la carpeta *docs* (creada anteriormente).

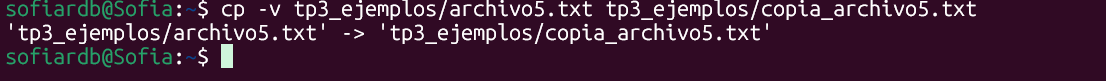


* 1. Copia el archivo *archivo3.txt* al directorio *docs*, y el archivo *archivo5.txt* al directorio *bin*, y asegúrate de preservar sus fechas de modificación original.



* 1. Copia el archivo llamado *archivo5.txt* de su directorio original y renómbralo a

*copia\_archivo5.txt* en el mismo directorio.

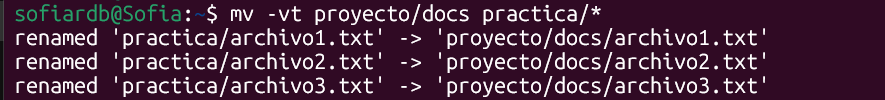


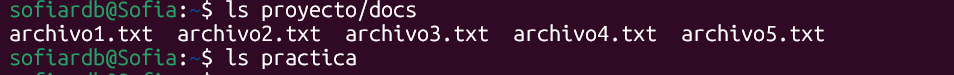
Comando mv (mover o renombrar archivos).

* 1. Mueve el archivo *archivo2.txt* desde su ubicación al directorio *src*.
  2. Renombra el archivo *archivo2.txt* como *datos2.txt*.



* 1. Mueve todos los archivos del directorio *practica* al directorio *docs*. Observa qué sucede.





Se movieron los archivos a la carpeta docs dentro de proyecto y práctica quedó vacía. Como ya existía un archivo3.txt en docs, fue reemplazado por el de práctica.

* 1. Intenta mover un archivo que no existe y observa el error que se presenta.



* 1. Renombra el directorio *docs* como *data*, sin moverlo de su ubicación.

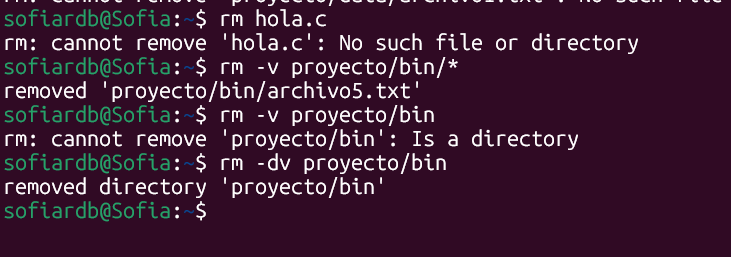


Comando rm (eliminar archivos).

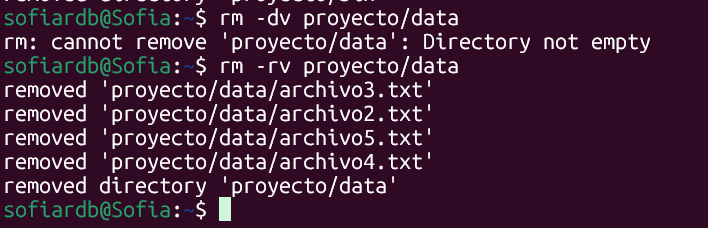
* 1. Elimina el archivo *archivo1.txt*.



* 1. Intenta eliminar un archivo que no existe y observa el mensaje de error.
  2. Elimina todos los archivos del directorio *bin*.
  3. Elimina el directorio *bin*.



* 1. Elimina el directorio *data* con todo su contenido.



# Opciones del comando gcc.

Escribe un programa que le pida al usuario ingresar un número *x* entre 1 y 20, controle que el valor ingresado es correcto, si no lo es lo pida de nuevo, y calcule los *x* primeros números de serie de Fibonacci y los presente por pantalla, finalmente que pida pulsar una tecla para salir.

Compilar un código en C:

* 1. Escribe el código descripto en lenguaje C.
  2. Compila el programa con la opción -c y observa el archivo resultado. ¿Qué contiene?

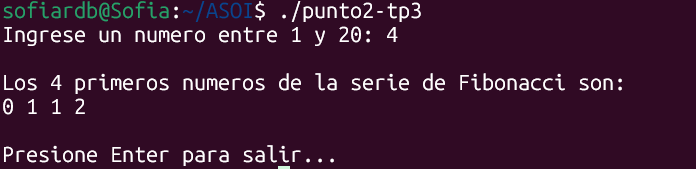


El archivo punto2-tp3.o es un objeto relocatable que contiene el código máquina parcial de tu programa, todavía no ejecutable, y que espera la etapa de linking para transformarse en binario ejecutable.

* 1. Obtén el código en lenguaje assembler.



* 1. Compila utilizando el archivo de salida anterior y ejecuta el programa.



Compilar un código en C++:

* 1. Escribe el código en C++.
  2. Compila el programa con la opción -c y observa el archivo resultado. Compáralo con el archivo de salida del ejercicio anterior.

Ambos tienen secciones .text, .data, .bss, etc. El de C++ además incluye símbolos de la librería estándar de C++ (iostream, std::vector, etc.), por lo que suele ser más grande y complejo que el de C.

* 1. Obtén el código en lenguaje assembler. Compáralo con el assembler de C.

En C el assembler es más simple (usa printf, scanf, bucles básicos).En C++ aparece mucho código extra debido al: manejo de std::cout, std::cin, plantillas de std::vector y configuración de iostream en tiempo de ejecución.

Por eso el assembler de C++ es mucho más largo y complejo.

* 1. Compila utilizando el archivo de salida anterior y ejecuta el programa.

