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.

1. 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):

1.1. Lista todos los archivos y directorios del directorio actual, incluyendo los archivos ocultos.
1.2. Muestra el contenido del directorio */usr/bin* en formato largo, con detalles como permisos, tamaño y fecha de modificación.
1.3. Muestra el contenido del directorio */etc* en formato largo y ordenado por fecha de modificación.
1.4. Lista los archivos del directorio actual y muestra el tamaño de los archivos en formato legible (KB, MB).
1.5. Lista el contenido del directorio */var/log* sin mostrar archivos ocultos.

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

1.6. ¿Cuál es el directorio de trabajo por defecto en GNU/Linux? ¿Qué símbolo utiliza GNU/Linux para denominarlo en los comandos?
1.7. Crea un directorio llamado *practica* en el directorio personal del usuario.
1.8. Cambia al directorio *practica* y muestra el camino completo desde la raíz, usando el comando *pwd*.
1.9. Navega al directorio padre del actual.
1.10. Crea una estructura de directorios donde *proyecto* sea el directorio raíz, y dentro de él haya subdirectorios llamados *src*, *bin*, y *docs*.
1.11. Desde el directorio por defecto, cambia al directorio *docs* dentro de *proyecto* en un solo comando.
1.12. Crea un directorio llamado *backup* dentro del directorio *proyecto* y verifica su existencia utilizando *ls*.
1.13. Cambia entre el directorio actual y el directorio anterior varias veces.
1.14. Intenta crear un directorio que ya existe.
1.15. Intenta cambiar a un directorio que no existe y observa el mensaje de error.

Comando *cp* (copiar archivos):

1.16. Copia el archivo */etc/hosts* al directorio personal.
1.17. Copia los archivos *archivo1.txt* al *archivo3.txt* a la carpeta *practica* (creada anteriormente).

1.18. Copia los archivos *archivo4.txt* y *archivo5.txt* a la carpeta *docs* (creada anteriormente).
1.19. 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.
1.20. 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).

1.21. Mueve el archivo *archivo2.txt* desde su ubicación al directorio *src*.
1.22. Renombra el archivo *archivo2.txt* como *datos2.txt*.
1.23. Mueve todos los archivos del directorio *practica* al directorio *docs*. Observa qué sucede.
1.24. Intenta mover un archivo que no existe y observa el error que se presenta.
1.25. Renombra el directorio *docs* como *data*, sin moverlo de su ubicación.

Comando rm (eliminar archivos).

1.26. Elimina el archivo *archivo1.txt*.
1.27. Intenta eliminar un archivo que no existe y observa el mensaje de error.
1.28. Elimina todos los archivos del directorio *bin*.
1.29. Elimina el directorio *bin*.
1.30. Elimina el directorio *data* con todo su contenido.

2. 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:

2.1. Escribe el código descripto en lenguaje C.
2.2. Compila el programa con la opción -c y observa el archivo resultado. ¿Qué contiene?
2.3. Obtén el código en lenguaje assembler.
2.4. Compila utilizando el archivo de salida anterior y ejecuta el programa.

Compilar un código en C++:

2.5. Escribe el código en C++.
2.6. Compila el programa con la opción -c y observa el archivo resultado. Compáralo con el archivo de salida del ejercicio anterior.
2.7. Obtén el código en lenguaje assembler. Compáralo con el assembler de C.
2.8. Compila utilizando el archivo de salida anterior y ejecuta el programa.