Trabajo Práctico N°5

Comandos básicos y avanzados

Objetivos: Afianzar los conocimientos adquiridos sobre los comandos básicos y desarrollar nuevas habilidades con comandos avanzados de GNU/Linux, optimizando el uso de la terminal y adquiriendo un manejo más profundo y versátil del sistema operativo.

Descarga el archivo `sort.tar.gz` del aula virtual en el directorio `home` del usuario con el que iniciaste sesión. Recuerda utilizar el comando **`man <comando>`**, o bien, **`<comando> -- help`**, para obtener ayuda de los comandos.

1. Investigación del comando `sort`. En este ejercicio, deberás investigar el funcionamiento del comando `sort`, utilizado para ordenar líneas de texto. Realiza una búsqueda y responde a las siguientes preguntas:
   a) ¿Qué hace el comando `sort` y cuál es su sintaxis básica? Explica brevemente su función y cómo se utiliza en su forma más simple.
   b) ¿Cuáles son las reglas por defecto de ordenación cuando no se especifican opciones? Asegúrate de mencionar cómo se ordenan los caracteres, letras y números. Prueba el comando sin opciones con los archivos `distros.txt`, `distros.num.txt`, y `numeros.txt`. ¿Qué observas?
   c) ¿Cuáles son los criterios de valores relativos entre los caracteres? Investiga cómo se comparan números, letras mayúsculas y minúsculas para decidir el ordenamiento.
   d) ¿Con qué opción se ordena numéricamente? Identifica la opción que permite ordenar según el valor numérico de los elementos. Prueba el comando con esa opción con el archivo `distros.num.txt`. Luego con el archivo `numeros.txt`, y compara con el resultado de no usar ninguna opción. ¿Qué criterio utiliza el comando?
   e) ¿Con qué opción se ordena en orden inverso? Indica la opción que permite invertir el orden de la lista. Prueba con los archivos `distros.txt` y `distros.num.txt`.
   f) ¿Con qué opción se ordena aleatoriamente? Investiga si existe una forma de ordenar los elementos de manera aleatoria. Prueba con todos los archivos, ejecutando más de una vez la línea de comando.
   g) ¿Con qué opción se ordena por una columna específica? Averigua cómo puedes ordenar líneas basándote en una columna específica de datos.
   h) ¿Con qué opción se eliminan las líneas duplicadas? Identifica la opción que elimina las líneas duplicadas durante la ordenación. Prueba con el archivo `distros.duplicados.txt`.

2. Extrae el contenido del archivo `sort.tar.gz` en un directorio llamado `ejemplos.sort`.
   a) ¿Cuál es línea de comandos para hacer esto y qué debes tener en cuenta antes?
   b) Lista el contenido del archivo sin extraerlo y guarda el listado en un archivo llamado `contenido.sort.txt`.
   c) Empaqueta nuevamente los archivos y comprímelos, esta vez, utilizando el comando `bzip2`. ¿Qué debes tener en cuenta antes de hacerlo?

3. Con el archivo `AnalectasConfucio.txt` de la carpeta `almacen/LibrosyFragmentos` del práctico anterior:
   a) Averigua cuantas palabras hay en la línea 14 del archivo.
   b) Cuenta cuántas veces aparece la palabra exacta **Estado** en el archivo.
   c) Muestra en pantalla las líneas que contengan el texto **stado**.
   d) Abre el archivo para poder navegarlo por páginas, y busca todas las apariciones de la palabra **estado**.

4. Con el archivo `Curioso_impertinente.txt` de la carpeta `almacen/LibrosyFragmentos` del práctico anterior:
   a) Averigua cuantas palabras tiene el archivo en total.
   b) Muestra en pantalla las líneas que contengan la palabra **Anselmo**, independientemente si está con mayúsculas o minúsculas.

5. Con el archivo `alumnos.txt` de la carpeta `ejemplos.sort`:
   a) Muestra en pantalla el archivo para navegarlo por páginas. Prueba la posibilidad de avanzar y retroceder las páginas. Prueba ir al inicio y al final del archivo.
   b) Observa si hay líneas repetidas. Si es así, en una línea de comandos, muestra en pantalla el archivo ordenado y sin líneas repetidas, pero que se pueda navegar por páginas.
   c) Repite el ejercicio anterior, pero además ordenando el archivo por DNI. ¿Qué debes tener en cuenta en ese ordenamiento?

6. Con el archivo `peliculas.txt` de la carpeta `ejemplos.sort`:
   a) Lista las cinco primeras líneas para tener un pantallazo del contenido.
   b) Muestra en pantalla, en una sola línea de comandos, los distintos géneros cinematográficos.
   c) Crea, en una sola línea de comandos, una carpeta llamada `películas`, con una subcarpeta por cada género cinematográfico. Si te animas, averigua cómo hacerlo tomando como entrada el resultado de la línea de comando del apartado anterior.
   d) Genera un archivo de texto vacío con cada género cinematográfico, por ejemplo, `ficcion.txt`. Si te animas, toma como entrada para esta línea de comandos, la salida de la línea de comandos del apartado **b)**. Guárdalos en la carpeta que le corresponda a cada uno.
   e) Guarda, en cada archivo, el listado de las películas según su género. Si te animas, guarda todos los campos, incluyendo la primera línea que tiene las etiquetas de las columnas.

7. Realiza las siguientes búsquedas:
   a) Encuentra todos los archivos con extensión `.conf` en la carpeta `/etc`.
   b) En la misma carpeta, encuentra los archivos que comiencen con **sys**.
   c) Encuentra en la carpeta `/var/log` los archivos modificados en los últimos 7 días.
   d) En la misma carpeta, encuentra los archivos que no se modificaron en los últimos 30 días.

8. Ejecuta tres comandos `sleep 5000` en paralelo y en segundo plano.
   a) Visualiza todos los procesos activos en el sistema. Identifica los procesos que ejecutaste.
   b) Visualiza únicamente los procesos que pertenecen al usuario con el que te lograste.
   c) Crea un archivo de texto con el listado de los procesos activos que pertenecen al usuario *root*.

9. Verifica si los comandos `sleep` del apartado anterior ya finalizaron.
   a) Si ya finalizaron, ejecútalos de nuevo. Identifica los procesos que estén ejecutando en segundo plano.
   b) Pasa al primer plano el trabajo con número de trabajo 2.
   c) Ejecuta el comando `top`, identifica cada proceso `sleep` y envíales la señal de que terminen.

10. Escribe un programa en C que escriba cada medio segundo, en un archivo de texto, una línea con el mensaje “línea número [nro_de_linea]”. El programa debe recibir por teclado la cantidad de tiempo, especificada en segundos, que el mismo deberá realizar esa tarea. Con un comando adecuado, visualiza en tiempo real cómo va cambiando el contenido del archivo, listando las tres últimas veces que el programa escribió en el archivo.