103.4 Usar flujos, pipes y redirecciones - Ejercicios

Ejercicio 3.4.1: Redireccionando Salida a Archivos

- **Objetivo:** Guardar la salida de un comando en un archivo, sobrescribiendo o añadiendo.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal.
    2. **Redirecciona la salida de `ls` a un archivo (sobrescribir):** Ejecuta `ls -l / > lista_root.txt`.
    3. **Verifica el contenido del nuevo archivo:** Ejecuta `cat lista_root.txt`. Deberías ver el listado del directorio raíz.
    4. **Redirecciona la salida de `date` al mismo archivo (sobrescribir):** Ejecuta `date > lista_root.txt`.
    5. **Verifica el contenido de nuevo:** Ejecuta `cat lista_root.txt`. Ahora solo verás la fecha; el listado de `ls` fue sobrescrito.
    6. **Añade la salida de `pwd` al final del archivo:** Ejecuta `pwd >> lista_root.txt`.
    7. **Verifica el contenido:** Ejecuta `cat lista_root.txt`. Verás la fecha seguida de tu directorio actual en una nueva línea.
    8. **Limpia:** Ejecuta `rm lista_root.txt`.

Ejercicio 3.4.2: Redireccionando Error Estándar

- **Objetivo:** Capturar los mensajes de error de un comando en un archivo.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal.
    2. **Intenta listar un directorio inexistente y redirecciona solo el error:** Ejecuta `ls /directorio_que_no_existe 2> error_ls.txt`. No verás ningún error en la pantalla.
    3. **Verifica el contenido del archivo de error:** Ejecuta `cat error_ls.txt`. Deberías ver el mensaje de error de `ls` (ej: `ls: cannot access '/directorio_que_no_existe': No such file or directory`).
    4. **Intenta listar un directorio inexistente y redirecciona ambas salidas al mismo archivo (usando `&>`):** Ejecuta `ls /directorio_que_no_existe / > todo_ls.txt &> error_y_salida.txt`. (Nota: `ls /` irá a stdout, `ls /directorio_que_no_existe` irá a stderr). Si el archivo `todo_ls.txt` se crea, estará vacío. El archivo `error_y_salida.txt` contendrá ambas salidas.
    5. **Verifica el contenido del archivo combinado:** Ejecuta `cat error_y_salida.txt`. Deberías ver tanto el listado de `/` como el mensaje de error.
    6. **Redirecciona todo a `/dev/null` (silenciar un comando):** Ejecuta `ls /directorio_que_no_existe &> /dev/null`. No verás ninguna salida ni error en la terminal.

7. **Limpia:** Ejecuta `rm error_ls.txt error_y_salida.txt todo_ls.txt`.

Ejercicio 3.4.3: Usando Pipes para Combinar Comandos

- **Objetivo:** Enviar la salida de un comando como entrada a otro usando la tubería (`|`).
- **Desarrollo Paso a Paso:**
    1. Abre una terminal.
    2. **Lista archivos y directorios, y envía la salida a `less`:** Ejecuta `ls -l /etc | less`. Esto te permite ver la lista larga de `/etc` paginada. Presiona `q` para salir de `less`.
    3. **Lista archivos, filtra por un patrón usando `grep`:** Ejecuta `ls /bin | grep zip`. Esto listará los archivos en `/bin` que contienen la cadena "zip".
    4. **Obtén información del sistema y cuenta las líneas:** Ejecuta `uname -a | wc -l`. Esto enviará la salida del comando `uname -a` a `wc -l`, que contará cuántas líneas tiene esa salida (normalmente 1).
    5. **Muestra un archivo de log, filtra líneas con "error", y cuenta cuántas hay:** Ejecuta `cat /var/log/syslog | grep "error" | wc -l`. (Ajusta el archivo de log si es necesario). Esto muestra el log, lo filtra buscando líneas con "error", y cuenta el número de líneas resultantes.
    6. **Ordena la lista de usuarios del sistema:** Ejecuta `cat /etc/passwd | sort`. Esto muestra el archivo `/etc/passwd` (que lista usuarios) ordenado alfabéticamente por la primera columna (el nombre de usuario).
    7. **Muestra el historial de comandos, filtra los comandos que contienen "ls", y muestra solo las últimas 5 líneas:** Ejecuta `history | grep "ls" | tail -n 5`. Encadenas tres comandos.

Ejercicio 3.4.4: Redireccionando Entrada desde un Archivo (Alternativa a Argumento)

- **Objetivo:** Usar la redirección `<` para proporcionar la entrada de un archivo a un comando.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal.
    2. **Crea un archivo de prueba con varias líneas (algunas repetidas):** Ejecuta `echo -e "manzana\npera\nmanzana\uva\npera" > frutas2.txt`.
    3. **Usa `sort` leyendo desde el archivo como argumento:** Ejecuta `sort frutas2.txt`.
    4. **Usa `sort` leyendo desde el archivo a través de stdin (usando `<`):** Ejecuta `sort < frutas2.txt`. La salida es la misma. Esto demuestra que para `sort` (y muchos filtros) leer un archivo pasado como argumento o leerlo a través de stdin es equivalente en términos de procesamiento, aunque internamente el comando se implemente de forma diferente.
    5. **Usa `uniq` leyendo desde stdin (requiere ordenar antes):** Ejecuta `sort < frutas2.txt | uniq`.

6. **Limpia:** Ejecuta `rm frutas2.txt`.