Módulo 3: Ayuda, redirecciones y tuberías

Objetivos

- Obtener ayuda sobre cualquier comando con `man`, `--help`, `whatis` y `apropos`.
- Redirigir la salida de un comando a un archivo con > y >>.
- Usar la redirección de entrada con <.
- Encadenar comandos mediante tuberías (|) para procesar datos de forma eficiente.
- Filtrar líneas con `grep`, ordenar con `sort`, eliminar duplicados con `uniq` y contar con `wc`.

Contenido teórico

Ayuda integrada: cómo descubrir comandos

Antes de profundizar, es clave saber cómo obtener ayuda sin salir de la terminal.

- `comando --help` – Muestra un resumen rápido de las opciones del comando (funciona en la mayoría).
- `man comando` – Abre el manual completo (manual page) con `less`. Navega con flechas, busca con `/`, sale con `q`.
- `whatis comando` – Describe brevemente para qué sirve el comando.
- `apropos palabra` – Busca comandos relacionados con una palabra clave. Por ejemplo, `apropos copy` muestra comandos que copian.

Bash 1: Ejemplos de ayuda

```
ls --help                # Opciones de ls
man ls                   # Manual completo
whatis ls                # "list directory contents"
apropos "copy file"      # Encuentra comandos relacionados
```

Redirección de salida: > y >>

Por defecto, los comandos escriben su salida en la terminal (stdout). Podemos redirigir esa salida a un archivo.

- `comando > archivo` – Redirige la salida a `archivo`, **sobrescribiendo** su contenido si ya existe.
- `comando >> archivo` – Redirige la salida **añadiéndola** al final del archivo (sin borrar lo anterior).

Bash 2: Ejemplos de redirección de salida

```
echo "Hola" > saludo.txt   # Crea saludo.txt con "Hola"
echo "Mundo" >> saludo.txt # Añade "Mundo" al final
ls -l > lista.txt # Guarda el listado detallado en lista.txt
```

Redirección de entrada: <

Algunos comandos pueden recibir entrada desde un archivo en lugar del teclado. El símbolo < redirige el contenido de un archivo como entrada.

Bash 3: Ejemplo de redirección de entrada

```
sort < lista.txt    # Ordena las líneas de lista.txt
wc -l < archivo.txt # Cuenta líneas (sin mostrar archivo.txt)
```

Tuberías: |

Las tuberías permiten conectar la salida de un comando con la entrada del siguiente. Se construyen con el símbolo `|`.

Bash 4: Ejemplos de tuberías

```
ls -l | grep "^-"          # Solo archivos (no directorios)
ls -l | tail -3            # Últimas 3 líneas del listado
ls | grep ".txt"           # Archivos que contengan .txt
```

Filtros básicos

Estos comandos se usan muy a menudo en tuberías.

`grep`: buscar texto

`grep patrón archivo` muestra las líneas que contienen el patrón. Opciones útiles:

- `-i`: ignora mayúsculas/minúsculas.
- `-v`: muestra las líneas que **no** contienen el patrón.
- `-n`: muestra el número de línea.

```
grep "error" log.txt
ls -l | grep "^d"          # Muestra solo directorios
grep -i "usuario" /etc/passwd
```

Patrones básicos en `grep` (expresiones regulares)

`grep` entiende patrones especiales llamados **expresiones regulares** (regex). Los más útiles para empezar son:

Símbolo	Significado	Ejemplo
`^`	Inicio de línea	`grep "^error" archivo` → líneas que empiezan con "error".
`$`	Fin de línea	`grep "done$" archivo` → líneas que terminan con "done".
`.`	Un solo carácter cualquiera	`grep "c.t" archivo` → "cat", "cut", "cbt", etc.
`*`	El carácter anterior 0 o más veces	`grep "ca*t" archivo` → "ct", "cat", "caat", ...
`[abc]`	Un carácter del conjunto	`grep "c[aeiou]t" archivo` → "cat", "cit", "cot", ...

Bash 5: Ejemplos de expresiones regulares en `grep`

```
grep "^-" archivo.txt        # Líneas que empiezan con guion
grep "bash$" /etc/passwd     # Líneas que terminan en "bash"
grep "c.t" diccionario.txt   # Palabras de 3 letras: cat, cut, cit
grep "ca*t" archivo          # ct, cat, caat, caaat...
grep "[0-9]" archivo         # Líneas que contienen algún dígito
grep "^[A-Z]" archivo        # Líneas que empiezan con mayúscula
```

`sort`: ordenar líneas

`sort` ordena alfabéticamente. Opciones:

- `-r`: orden inverso.
- `-n`: orden numérico.

```
sort archivo.txt
ls -l | sort -k5 -n  # Por tamaño (columna 5) numéricamente
```

`uniq`: eliminar duplicados

`uniq` elimina líneas duplicadas **adyacentes**. Por eso suele usarse después de `sort`.

```
sort lista.txt | uniq
sort lista.txt | uniq -c   # cuenta veces que aparece cada línea
```

wc: contar

`wc` (word count) cuenta líneas, palabras y bytes.

- `wc -l`: solo líneas.
- `wc -w`: solo palabras.
- `wc -c`: solo bytes.

```
wc -l archivo.txt
ls | wc -l                # Cuenta cuántos archivos hay en el
   directorio
```

cut: extraer columnas

`cut` selecciona partes de cada línea (columnas). Muy útil para archivos CSV o datos tabulares.

- `-d` : especifica el delimitador (por defecto TAB).
- `-f` : indica qué campo(s) extraer (1-indexado).

```
cut -d "," -f 2 datos.csv   # Muestra la segunda columna (CSV)
cut -d ":" -f 1 /etc/passwd # Muestra solo los nombres de usuario
ls -l | cut -d " " -f 1     # Extrae la primera columna
```

Combinando todo

Bash 6: Ejemplos integrados

```
# Contar archivos .txt
ls *.txt | wc -l

# Líneas más frecuentes en un log
sort log.txt | uniq -c | sort -nr | head -5

# Buscar líneas que contengan "error" y mostrar las primeras 3
grep "error" archivo.log | head -3

# Extraer la columna de nombres de usuarios desde /etc/passwd
cut -d ":" -f 1 /etc/passwd | head -5
```

Tips adicionales

Tip 5: Búsqueda inversa en el historial con `Ctrl+R`

En la terminal, presiona `Ctrl+R` y empieza a escribir parte de un comando anterior. Bash mostrará el comando más reciente que coincida. Puedes presionar `Ctrl+R` repetidamente para buscar coincidencias más antiguas. Presiona `Enter` para ejecutarlo o `Ctrl+G` para cancelar.

Tip 6: `tee` – ver y guardar al mismo tiempo

Cuando usas > o >>, la salida del comando se escribe en el archivo pero **no la ves en la terminal**. Si quieres verla mientras la guardas, usa `tee`:

```
ls -l | tee listado.txt
```

Esto muestra el listado en pantalla y también lo escribe en `listado.txt`. Es como un "desvío" que copia la salida a un archivo sin interrumpir el flujo.

Diferencia práctica:

- `ls -l >listado.txt` → guarda, **no ves nada** en pantalla.
- `ls -l | tee listado.txt` → ves la salida **y además** se guarda.

Muy útil para revisar qué estás guardando, especialmente en tuberías largas.

Tip 7: Para profundizar en expresiones regulares

Las expresiones regulares son un lenguaje enorme. Con lo visto aquí es suficiente para el curso, pero si quieres dominarlas:

- TLCL – Capítulo 19 (expresiones regulares).
- Manual oficial de grep (sección "Regular Expressions").
- Comando `man 7 regex` (en tu terminal, si está instalado).

Advertencia sobre redirecciones

Cuidado con > cuando ya existe un archivo importante, porque lo sobrescribe sin preguntar. Acostúmbrate a usar >> si quieres añadir, o verifica antes con `ls`.

Ejercicios prácticos

Resuelve en tu terminal:

1. Usa `man` para averiguar qué hace la opción `-h` de `ls`. ¿Existe? ¿Cuál es la opción para mostrar tamaños legibles?

2. Redirige la salida de `ls -l` a un archivo llamado `listado.txt`. Luego añade al final la salida de `pwd`.

3. Crea un archivo `numeros.txt` con líneas desordenadas (por ejemplo, usando `echo` y >>) que contenga números del 1 al 10 desordenados y algunos repetidos. Luego, usando una tubería, muestra los números ordenados y sin duplicados.

4. Usa `grep` para mostrar solo las líneas de `/etc/passwd` que contengan el nombre de tu usuario.

5. Cuenta cuántos directorios hay en tu directorio personal. Pista: `ls -l | grep "^d"| wc -l`.

6. (Desafío) Encuentra el comando que hayas ejecutado hace unos minutos usando `Ctrl+R` (búsqueda inversa). Practica escribiendo solo unas letras.

7. (Opcional) Usa `tee` para guardar una lista de archivos en `mis_archivos.txt` mientras la ves en pantalla.

8. Crea un archivo CSV llamado `datos.csv` con al menos 3 columnas (nombre, edad, ciudad). Usa `cut` para extraer la columna de edades, ordénalas y cuenta cuántas personas hay de cada edad.

Referencias

- The Linux Command Line (TLCL) – Capítulos 7 y 8.
- The Bash Guide – Sección sobre redirecciones y tuberías.

Resumen

En este módulo aprendiste a:

- Obtener ayuda sin salir de la terminal.
- Redirigir salidas (sobrescribir o añadir) y entradas.
- Construir tuberías para combinar comandos simples y resolver tareas complejas.
- Usar filtros básicos: `grep`, `sort`, `uniq`, `wc`.
- Aplicar técnicas como `Ctrl+R` para recuperar comandos del historial.

Ahora puedes procesar texto de forma eficiente. En el próximo módulo veremos búsqueda de archivos, permisos y procesos.