Módulo 4: Búsqueda, permisos y procesos

Objetivos

- Buscar archivos en el sistema con `find` (por nombre, tipo, tamaño, tiempo de modificación).
- Buscar texto dentro de archivos de forma recursiva con `grep -r`.
- Interpretar los permisos de archivos y directorios (lectura, escritura, ejecución).
- Cambiar permisos con `chmod` (modo numérico y simbólico).
- Cambiar el propietario de un archivo con `chown`.
- Gestionar procesos: listar con `ps` y `top`, enviar señales con `kill`, y controlar trabajos en segundo plano (`&`, `jobs`, `fg`, `bg`, `Ctrl+Z`).

Contenido teórico

Búsqueda de archivos: `find`

El comando `find` permite buscar archivos y directorios según múltiples criterios.

Bash 1: Sintaxis básica

```
find [ruta] [opciones] [acción]
```

Opciones más comunes:

- `-name "patrón"` – busca por nombre (distingue mayúsculas/minúsculas). Usa comodines: `*` y `?`.
- `-iname "patrón"` – igual pero ignora mayúsculas.

- `-type f` o `-type d` – solo archivos regulares o solo directorios.
- `-size +10M` – archivos mayores a 10 MB. Se puede usar `k`, `M`, `G`.
- `-mtime -7` – modificados en los últimos 7 días. `-mtime +30` – más de 30 días.

Bash 2: Ejemplos de `find`

```
find /home -name "*.txt"       # todos los .txt en /home
find . -type d -name "backup" # directorios llamados "backup"
find /var/log -size +100M     # archivos de log mayores a 100 MB
find ~ -mtime -1              # modificados en las últimas 24hs
```

Búsqueda recursiva de texto: `grep -r`

Para buscar una cadena dentro de archivos de forma recursiva, usamos `grep -r`:

```
# busca "error" en todos los archivos de /var/log
grep -r "error" /var/log/
# solo en archivos .conf
grep -r --include="*.conf" "Listen" /etc/
```

Permisos en Linux

Cada archivo y directorio tiene permisos para tres grupos: **propietario (u)**, **grupo (g)** y **otros (o)**.

- **r** (read) – lectura.
- **w** (write) – escritura.
- **x** (execute) – ejecución (para archivos) o acceso (para directorios).

Ver permisos con `ls -l`. El primer carácter indica tipo (`-` archivo, `d` directorio). Luego tres grupos de tres letras: `rwxr-xr--`.

Usuarios y grupos: el contexto de los permisos

Cuando ejecutas `ls -l`, además de los permisos verás dos nombres: el del **propietario** y el del **grupo** al que pertenece el archivo. Por ejemplo:

```
-rwxr-xr-- 1 alicia desarrollo 1234 mar 10 10:00 procesar_info.sh
```

Aquí:

- `alicia` es el propietario (usuario).
- `desarrollo` es el grupo asociado.
- Los permisos `rwxr-xr--` significan: propietario puede leer/escribir y ejecutar; los miembros del grupo `desarrollo` pueden leer y ejecutar; los otros usuarios (resto del sistema) solo pueden leer.

Entonces debemos tener en cuenta que:

- El propietario puede cambiar los permisos con `chmod`.
- Con `chown` (necesita `sudo`) se puede cambiar el propietario: `sudo chown pedro informe.txt`.
- Con `chgrp` o `chown :grupo` se puede cambiar el grupo: `sudo chown :marketing informe.txt`.

Entender quién es el propietario y el grupo te permite dar permisos adecuados (por ejemplo, que solo el grupo pueda editar un archivo, pero otros no).

Además, cabe aclarar que todos los usuarios poseen un grupo con su mismo nombre, por eso es común ver que la mayoría de los archivos al ejecutar `ls -l` dentro del home tiene una salida como:

```
drwxrw-rw- 1 usuario usuario 4096 may 22 08:55 Documentos
```

El propietario es `usuario` y el grupo homónimo.

Tip 8: Permisos numéricos (octales)

Cada permiso tiene un valor:

- r = 4
- w = 2
- x = 1

Sumas para cada grupo. Ejemplo: `chmod 755 archivo` da al propietario rwx (4+2+1=7), al grupo r-x (4+0+1=5), a otros r-x (5).

Bash 3: Ejemplos de `chmod`

```
chmod u+x script.sh      # añade ejecución para el propietario
chmod go-w archivo.txt   # quita escritura a grupo y otros
chmod 644 config.cfg     # propietario rw-; grupo r--; otros r--
chmod -R 755 carpeta/    # recursivo (también subcarpetas)
```

Cambiar propietario: `chown`

Solo el superusuario (root) o el propietario actual (en algunos casos) puede cambiar el propietario.

```
sudo chown usuario archivo       # cambia propietario a 'usuario'
sudo chown usuario:grupo archivo # cambia propietario y grupo
```

Procesos

Un proceso es un programa en ejecución.

- `ps` – lista procesos del terminal actual. Opciones comunes: `ps aux` (todos los procesos del sistema).
- `top` – muestra procesos en tiempo real (actualiza cada 2 segundos). Presiona `q` para salir.
- `kill` – envía una señal a un proceso (por defecto `SIGTERM`, terminación amable). `kill -9 PID` (`SIGKILL`) fuerza la terminación.

- `Ctrl+C` – termina el proceso en primer plano.
- `Ctrl+Z` – suspende el proceso en primer plano.
- `jobs` – muestra trabajos suspendidos o en segundo plano.
- `fg` – reanuda un trabajo en primer plano.
- `bg` – reanuda un trabajo en segundo plano.
- `&` – ejecuta un comando en segundo plano desde el inicio.

Bash 4: Ejemplos de gestión de procesos

```
sleep 100 &              # ejecuta sleep en segundo plano
jobs                     # muestra [1]+ Running sleep 100 &
fg                       # trae el último trabajo a primer plano
Ctrl+Z                   # lo suspende
bg                       # lo reanuda en segundo plano
ps aux | grep sleep      # busca el PID del proceso sleep
kill %1                  # termina el trabajo número 1 (por jobs)
kill -9 12345            # fuerza terminación del PID 12345
```

Combinando conceptos: ejemplo integrador

Bash 5: Buscar archivos con ciertos permisos y cambiarlos

```
# Buscar archivos .sh sin permiso de ejecución y agregarlo
find . -name "*.sh" ! -executable -exec chmod +x {} \;
```

La opción `-exec` ejecuta un comando sobre cada resultado de `find`. {} se reemplaza por el nombre del archivo, y `\;` termina el comando.

Tips adicionales

Tip 9: `find -exec` con + para mayor eficiencia

En lugar de `\;` (que ejecuta el comando una vez por archivo), puedes usar `+` para pasar múltiples archivos de una sola vez:

```
find . -name "*.log" -exec rm {} +
```

Es más rápido con muchos archivos.

Advertencia: Permisos de ejecución en directorios

Para entrar a un directorio (`cd`) necesitas permiso de ejecución (`x`) en él. Sin `x`, no puedes acceder aunque tengas lectura.

Tip 10: Ver árbol de procesos con `pstree`

Si tienes instalado `pstree`, muestra los procesos en forma jerárquica. Útil para entender relaciones padre-hijo.

Ejercicios prácticos

Resuelve en tu terminal:

1. Busca todos los archivos con extensión `.conf` dentro de `/etc` (usa `find`).
2. Encuentra directorios vacíos en tu home con `find ~ -type d -empty`.
3. Busca archivos modificados en los últimos 3 días dentro de `/var/log` (puede requerir `sudo`).
4. Usa `grep -r` para buscar la palabra “error” en todos los archivos de `/var/log` (usa `sudo` si es necesario).
5. Crea un archivo `prueba.txt` y asígnale permisos `rw-r--r--` (644). Luego cambia los permisos para que el grupo pueda escribir (`chmod g+w`).
6. Ejecuta `sleep 200` en segundo plano (`sleep 200 &`). Usa `jobs` para verlo, luego tráelo a primer plano con `fg` y suspéndelo con `Ctrl+Z`. Finalmente, reanúdalo en segundo plano con `bg`.
7. Encuentra el PID de un proceso `sleep` usando `ps aux | grep sleep` y termínalo con `kill`.
8. (Desafío) Busca todos los archivos `.txt` en tu home que tengan permisos de escritura para otros (`o+w`) y cambia esos permisos a solo lectura para otros con un solo comando `find`.

Referencias

- The Linux Command Line (TLCL) – Capítulos 9 (permisos), 10 (procesos), 11 (find).
- The Bash Guide – Secciones sobre permisos y procesos.

Resumen

En este módulo aprendiste a:

- Localizar archivos con `find` usando diversos criterios.
- Buscar texto recursivamente con `grep -r`.
- Comprender y modificar permisos con `chmod` (modo numérico y simbólico), entendiendo la relevancia de usuarios y grupos.
- Gestionar procesos: listarlos, enviar señales, y controlar trabajos en primer y segundo plano.

Estás preparado para el siguiente módulo, donde escribirás tus primeros scripts en Bash.