🔧 Examen 104 - Dispositivos, Sistemas de Archivos Linux y Jerarquía Estándar

104.5 Gestionar permisos y propiedad de archivos

Teoría

Cada archivo y directorio en un sistema de archivos Linux tiene asociado un conjunto de permisos y una propiedad que determinan quién puede acceder a ellos y de qué manera. Este es el modelo de permisos Discretionary Access Control (DAC) estándar en sistemas Unix.

Conceptos Clave:

1. **Propiedad (Ownership):** Cada archivo o directorio tiene un **propietario (owner)** (un usuario) y un **grupo propietario (group owner)** (un grupo).
2. **Permisos:** Para cada archivo o directorio, los permisos se definen para tres categorías de entidades:
    - **Usuario (u):** Los permisos para el propietario del archivo.
    - **Grupo (g):** Los permisos para los miembros del grupo propietario del archivo.
    - **Otros (o):** Los permisos para todos los demás usuarios en el sistema (que no son el propietario ni miembros del grupo propietario).
    - **Todos (a):** Se usa para referirse a las tres categorías (usuario, grupo, otros) simultáneamente en comandos como `chmod`.
3. **Tipos de Permisos:** Para cada una de las tres categorías (usuario, grupo, otros), se pueden asignar tres tipos de permisos:
    - **Lectura (r):**
        - Para un **archivo:** Permite ver el contenido del archivo.
        - Para un **directorio:** Permite listar el contenido del directorio (ver los nombres de los archivos y subdirectorios que contiene).
    - **Escritura (w):**
        - Para un **archivo:** Permite modificar o eliminar el archivo.
        - Para un **directorio:** Permite crear, eliminar o renombrar archivos y subdirectorios *dentro* de ese directorio. **Importante:** Para eliminar un archivo dentro de un directorio, necesitas permiso de escritura *en el directorio*, no necesariamente en el archivo mismo (aunque los permisos del archivo pueden evitar que modifiques su contenido antes de intentar borrarlo).
    - **Ejecución (x):**
        - Para un **archivo:** Permite ejecutar el archivo como un programa (si es un ejecutable o un script).
        - Para un **directorio:** Permite entrar en el directorio, acceder a sus subdirectorios y archivos, y usar comandos como `cd` para navegar a él.

Visualizando Permisos (`ls -l`):

El comando `ls -l` muestra los permisos y la propiedad en el formato largo. La primera columna de la salida es una cadena de 10 caracteres:

```
- rwx rwx rwx | | | | | | | tipo | usuario | grupo | otros | Tipo de archivo
```

- El primer carácter indica el tipo de archivo:
  - `-`: Archivo regular.
  - `d`: Directorio.
  - `l`: Enlace simbólico.
  - `c`: Dispositivo de carácter.
  - `b`: Dispositivo de bloque.
  - `s`: Socket de dominio Unix.
  - `p`: Pipe con nombre (FIFO).
- Los siguientes 9 caracteres son los permisos:
  - Caracteres 2-4: Permisos para el propietario (u).
  - Caracteres 5-7: Permisos para el grupo propietario (g).
  - Caracteres 8-10: Permisos para otros (o).
  - `r` = Lectura, `w` = Escritura, `x` = Ejecución.
  - `-` en lugar de `r`, `w` o `x` indica que el permiso no está asignado.

Ejemplo: `-rwxr-xr--`

- Es un archivo regular (`-`).
- El propietario tiene permisos de lectura, escritura y ejecución (`rwx`).
- El grupo propietario tiene permisos de lectura y ejecución (`r-x`).
- Otros usuarios solo tienen permiso de lectura (`r--`).

Representación Numérica (Octal) de Permisos:

Los permisos también se pueden representar usando números octales (base 8). A cada tipo de permiso se le asigna un valor:

- `r` (Lectura) = 4
- `w` (Escritura) = 2
- `x` (Ejecución) = 1
- `-` (Ninguno) = 0

Se suma el valor de los permisos para cada conjunto (usuario, grupo, otros):

- `rwx` = 4 + 2 + 1 = 7
- `r-w` = 4 + 0 + 2 = 6
- `r-x` = 4 + 0 + 1 = 5
- `rw-` = 4 + 2 + 0 = 6
- `---` = 0 + 0 + 0 = 0

Los permisos de un archivo o directorio se representan con una secuencia de tres dígitos octales: el primero para el usuario, el segundo para el grupo y el tercero para otros.

Ejemplo:

- `-rwxr-xr--` = Usuario `rwx` (7), Grupo `r-x` (5), Otros `r--` (4). Representación numérica: `754`.
- `-rw-rw-r--` = Usuario `rw-` (6), Grupo `rw-` (6), Otros `r--` (4). Representación numérica: `664`.
- `drwxr-xr-x` = Directorio `rwx` (7), Grupo `r-x` (5), Otros `r-x` (5). Representación numérica: `755`.

Cambio de Permisos (`chmod`):

El comando `chmod` (change mode) cambia los permisos de archivos y directorios. Puede usar la representación simbólica o numérica.

- **Simbólica:** `chmod [quien][+-=][permiso] archivo...`
  - `quien`: `u` (usuario), `g` (grupo), `o` (otros), `a` (todos). Puedes combinar (ej: `ug`). Si se omite, por defecto es `a` (todos).
  - `+-=`: `+` (añadir permiso), `-` (quitar permiso), `=` (establecer permisos exactos, eliminando los que no se especifiquen).
  - `permiso`: `r`, `w`, `x`.
  - Ejemplos:
    - `chmod u+x script.sh`: Añade permiso de ejecución para el propietario.
    - `chmod go-w archivo.txt`: Quita permiso de escritura para el grupo y otros.
    - `chmod ug=rw,o=r archivo.txt`: Establece rw para usuario y grupo, y r para otros (equivalente a `chmod 664 archivo.txt`).
    - `chmod +x script.sh`: Añade permiso de ejecución para todos (usuario, grupo, otros).
- **Numérica:** `chmod <3_digitos_octales> archivo...`
  - Ejemplo: `chmod 755 mi_directorio`: Establece rwx para usuario, r-x para grupo, r-x para otros.
  - Ejemplo: `chmod 644 mi_archivo.txt`: Establece rw- para usuario, r-- para grupo, r-- para otros.
- `chmod -R <modo> directorio`: Cambia permisos recursivamente en un directorio y su contenido.

Cambio de Propiedad y Grupo (`chown`, `chgrp`):

- `chown <nuevo_propietario>[:<nuevo_grupo>] archivo...`: Cambia el propietario y/o el grupo propietario. Solo el usuario `root` puede cambiar el propietario de un archivo a otro usuario.

    - `sudo chown otro_usuario archivo.txt`: Cambia solo el propietario.
    - `sudo chown :nuevo_grupo archivo.txt`: Cambia solo el grupo propietario.
    - `sudo chown otro_usuario:nuevo_grupo archivo.txt`: Cambia propietario y grupo.
    - `sudo chown -R <nuevo_propietario>[:<nuevo_grupo>] directorio`: Cambia propiedad recursivamente.
- `chgrp <nuevo_grupo> archivo...`: Cambia solo el grupo propietario. Los usuarios normales pueden usar `chgrp` en archivos que poseen, siempre que sean miembros del grupo al que intentan cambiar la propiedad.
    - `chgrp mi_grupo_nuevo archivo.txt`
    - `chgrp -R <nuevo_grupo> directorio`

Permisos Especiales (SUID, SGID, Sticky Bit):

Son bits adicionales en los permisos que otorgan capacidades especiales:

- **SUID (Set User ID):** En un *archivo ejecutable*, si el bit SUID está activado para el propietario, cuando *cualquier* usuario ejecuta ese archivo, el proceso se ejecuta con los permisos efectivos del **propietario** del archivo, no del usuario que lo ejecutó. Se muestra como `s` en lugar de `x` en la posición de ejecución del propietario (`-rwsr-xr-x`). Si el propietario no tiene permiso de ejecución, se muestra como `S` mayúscula (`-rwSr-xr-x`). Útil para programas que necesitan permisos elevados para una tarea específica (ej: `passwd` necesita escribir en `/etc/shadow`, que solo root puede hacer, por lo tanto, `passwd` tiene el bit SUID de root activado).
- **SGID (Set Group ID):**
    - En un *archivo ejecutable*, si el bit SGID está activado para el grupo, el proceso se ejecuta con los permisos efectivos del **grupo propietario** del archivo. Se muestra como `s` en lugar de `x` en la posición de ejecución del grupo (`-rwxrwsr-x`). Si el grupo no tiene permiso de ejecución, se muestra como `S` mayúscula.
    - En un **directorio**, si el bit SGID está activado, los archivos y subdirectorios *recién creados* dentro de ese directorio heredarán el **grupo propietario** del directorio padre, en lugar del grupo primario del usuario que los creó. Se muestra como `s` en lugar de `x` en la posición de ejecución del grupo (`drwxrwsr-x`). Útil para directorios compartidos donde todos los usuarios quieren que los archivos pertenezcan al mismo grupo.
- **Sticky Bit:**
    - En un **directorio**, si el bit sticky está activado, los usuarios solo pueden eliminar o renombrar archivos *dentro* de ese directorio si son el **propietario** del archivo O el propietario del directorio O el usuario `root`. Se muestra como `t` en lugar de `x` en la posición de ejecución de "otros" (`drwxrwxrwt`). Si "otros" no tiene permiso de

ejecución, se muestra como `T` mayúscula. Útil para directorios donde varios usuarios pueden escribir pero no deberían poder borrar los archivos de otros (ej: `/tmp`).
- En un *archivo*, el sticky bit histórico tenía otro significado, pero hoy en día solo es relevante para directorios.

Representación Numérica (Octal) de Permisos Especiales:

Los permisos especiales también tienen valores octales que se colocan como un cuarto dígito al principio de la secuencia de permisos:

- SUID = 4
- SGID = 2
- Sticky = 1

Ejemplo:

- `-rwsr-xr-x` (SUID activado) = `4755`
- `drwxrwsr-x` (SGID en directorio activado) = `2775`
- `drwxrwxrwt` (Sticky bit en directorio activado) = `1777`
- `drwxr-xr-x` (permisos normales en directorio) = `0755` o simplemente `755`.

Puedes usar estos valores numéricos con `chmod` (ej: `chmod 4755 archivo`, `chmod 1777 directorio`).

Máscara de Creación de Archivos (`umask`):

`umask` es un valor que determina los permisos *por defecto* de los archivos y directorios recién creados. Es una *máscara* de permisos que se *resta* (en un sentido binario invertido) de los permisos máximos posibles (666 para archivos, 777 para directorios).

- `umask`: Muestra el valor actual de umask (generalmente en formato octal).
- `umask -S`: Muestra el valor de umask en formato simbólico.
- El valor por defecto de umask para un usuario normal suele ser `002` (los dos primeros ceros son para bits especiales, el `2` afecta a "otros"). Esto da permisos por defecto de `664` para archivos (666 - 002 = 664) y `775` para directorios (777 - 002 = 775).
- Para el usuario root, umask suele ser `022`, dando permisos por defecto de `644` para archivos y `755` para directorios.
- `umask <valor_octal>`: Establece un nuevo valor de umask para la shell actual.