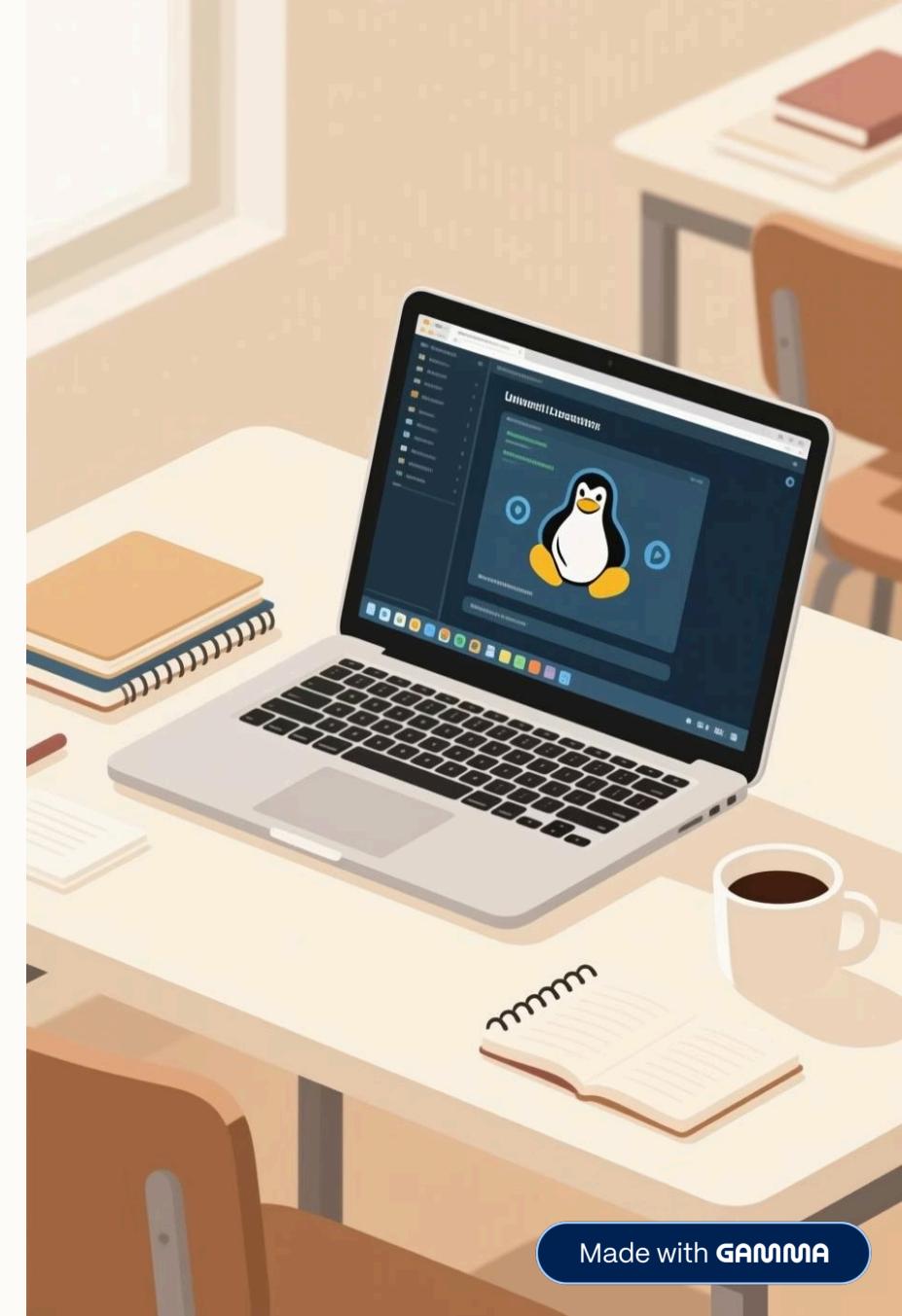


Introducción a la Ciberseguridad

Clase 1: Linux Básico



La Herramienta Esencial en Ciberseguridad



¿Por qué Linux?



Control Total

En Linux, hablás directamente con el sistema. No hay capas intermedias que oculten procesos o limiten tus acciones. Podés ver exactamente qué está ocurriendo y modificar cualquier aspecto del sistema operativo.



Transparencia

Al ser código abierto, Linux permite que cualquiera revise su código fuente. Esto significa que miles de ojos buscan constantemente vulnerabilidades, haciendo que sea más seguro por diseño.



Potencia

Linux viene con un arsenal de herramientas nativas para seguridad informática: analizadores de red, frameworks de pentesting, y utilidades de forense digital que son estándar en distribuciones como Kali Linux.

En el mundo de la ciberseguridad, Linux no es solo una opción: es una necesidad. El 90% de los servidores cloud, el 85% de los smartphones (Android está basado en Linux) y prácticamente todos los superordenadores del mundo utilizan Linux. Dominar este sistema operativo te dará una ventaja competitiva en el campo profesional y te permitirá comprender mejor cómo funcionan los sistemas que protegerás en el futuro.

Tu Centro de Comandos

La Terminal: El Corazón de Linux

La terminal (también llamada consola o línea de comandos) es una interfaz de texto donde damos órdenes directas al sistema operativo. A diferencia de las interfaces gráficas que utilizan metáforas visuales (carpetas, papelera, escritorio), la terminal te permite comunicarte con el sistema usando texto puro, lo que la hace extremadamente potente y eficiente.

Prompt (\$)

El símbolo \$ (o # para usuarios root) indica que la terminal está lista para recibir comandos. Suele mostrar también información como tu nombre de usuario, el nombre del host y el directorio actual.

```
usuario@maquina:~$
```

Comando

Es la instrucción que le das al sistema. Por ejemplo, ls para listar archivos. Los comandos en Linux suelen ser abreviaturas cortas que reflejan su función.

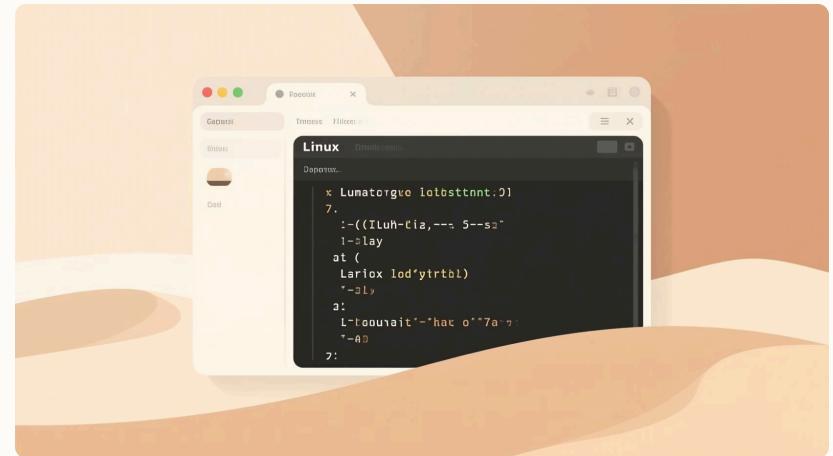
```
usuario@maquina:~$ ls
```

Argumentos y Opciones

Información adicional que le pasamos al comando para modificar su comportamiento. Las opciones suelen comenzar con guion (-) y pueden combinarse.

```
usuario@maquina:~$ ls -la /etc
```

Dominar la terminal no solo te hará más eficiente, sino que te dará acceso a herramientas y capacidades que simplemente no están disponibles a través de interfaces gráficas. En ciberseguridad, esto es crucial para realizar análisis forenses, pruebas de penetración y respuesta a incidentes.



💡 **¿Sabías que...?** La terminal que usamos hoy es descendiente directa de los teletipos (TTY) de los años 60, dispositivos electromecánicos que permitían enviar mensajes escritos a través de líneas telegráficas. Por eso muchos archivos de dispositivo en Linux comienzan con "tty".

Ubicación Actual

pwd

pwd → Print Working Directory

Este comando es tu GPS en el sistema de archivos de Linux. Te muestra la ruta completa del directorio en el que te encuentras actualmente, desde la raíz (/) hasta tu ubicación actual.

```
usuario@maquina:~$ pwd  
/home/usuario
```

Esto te indica que estás en el directorio "usuario" dentro de "/home", que es la carpeta personal del usuario. El símbolo ~ (tilde) es un atajo que representa tu directorio home.

¿Por qué es importante?

- Te ayuda a orientarte cuando trabajás con rutas relativas.
- Es esencial cuando ejecutás scripts que dependen de la ubicación.
- Evita errores al crear, mover o eliminar archivos.



El sistema de archivos de Linux

A diferencia de Windows con sus unidades C:, D:, etc., Linux tiene un único árbol de directorios que comienza en "/" (la raíz). Algunos directorios importantes:

- **/bin:** Comandos básicos del sistema
- **/etc:** Archivos de configuración
- **/home:** Directorios personales de los usuarios
- **/var:** Datos variables (logs, bases de datos)
- **/tmp:** Archivos temporales

- ❑ En ciberseguridad, conocer la estructura de directorios es crucial para encontrar logs, configuraciones y posibles vectores de ataque.

Recordá: antes de ejecutar cualquier comando potencialmente peligroso, siempre verificá dónde estás con `pwd`. ¡Ha salvado a muchos administradores de sistemas de cometer errores catastróficos!

Listando Archivos

ls

ls → List

Este comando es tu "linterna" en el sistema de archivos. Te permite ver qué contiene un directorio, mostrando archivos y subdirectorios.

```
usuario@maquina:~$ ls  
Descargas Documentos Escritorio Imágenes Música Público Vídeos
```

Por defecto, ls muestra solo los nombres de los archivos y directorios, ordenados alfabéticamente y en formato de columnas para facilitar la lectura.

Listando otros directorios

Podés especificar una ruta como argumento para ver su contenido sin tener que navegar hasta allí:

```
usuario@maquina:~$ ls /etc  
adduser.conf      hosts      passwd  
alternatives      hosts.allow  passwd-  
apache2          hosts.deny   ppp  
...  
...
```

Esto es extremadamente útil para explorar el sistema sin cambiar constantemente de directorio.



Identificando tipos de archivos

En muchas distribuciones, ls usa colores para diferenciar tipos de archivos:

- **Verde:** Archivos ejecutables
- **Azul:** Directorios
- **Cian:** Enlaces simbólicos
- **Rojo:** Archivos comprimidos
- **Blanco/Gris:** Archivos normales

⚠ En un entorno de ciberseguridad, prestar atención a los permisos y tipos de archivos puede revelar configuraciones inseguras o archivos sospechosos. Siempre observá con atención la salida de ls.

Casos de uso en ciberseguridad

- **Reconocimiento de sistema:** Identificar qué software está instalado explorando directorios como /bin, /sbin, /usr/bin.
- **Análisis forense:** Buscar archivos ocultos o con fechas sospechosas que podrían indicar una intrusión.
- **Auditoría de permisos:** Verificar qué archivos tienen permisos demasiado permisivos (veremos esto más adelante).

El comando ls es uno de los que más utilizarás en tu día a día con Linux, así que vale la pena conocer todas sus opciones.

Viendo Todos los Detalles

El comando `ls` base es útil, pero sus verdaderas capacidades se revelan cuando le agregamos opciones (flags).

`ls -l`

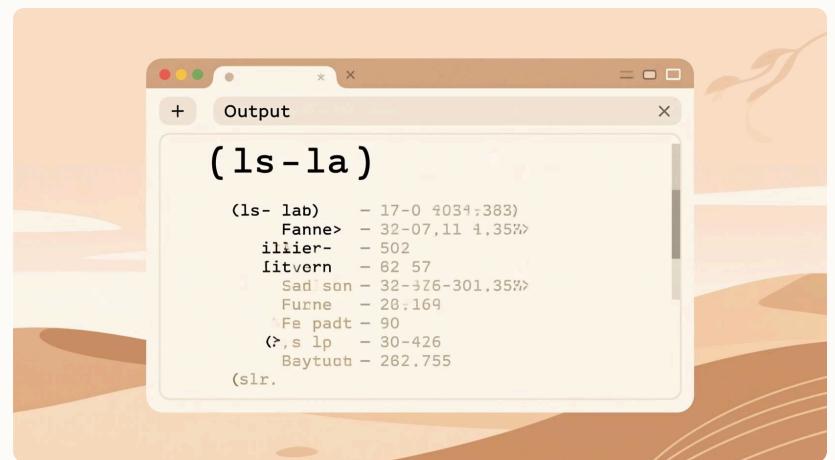
Muestra el formato de lista larga con detalles completos: permisos, propietario, grupo, tamaño, fecha de modificación y nombre.

```
-rw-r--r-- 1 usuario grupo 2048  
Sep 15 14:30 archivo.txt
```

`ls -a`

Muestra todos (**all**) los archivos, incluyendo los ocultos (que empiezan con punto). Estos archivos suelen contener configuraciones importantes.

```
usuario@maquina:~$ ls -a  
. . . .bash_history .bashrc  
Documentos .profile
```



Otras opciones útiles

- `ls -t`: Ordena por fecha de modificación (más recientes primero)
- `ls -S`: Ordena por tamaño (más grandes primero)
- `ls -R`: Lista recursivamente todos los subdirectorios
- `ls -i`: Muestra el número de inodo (útil para encontrar enlaces duros)

`ls -h`

Muestra tamaños en formato **humano** (KB, MB, GB), facilitando la lectura de tamaños de archivo.

```
-rw-r--r-- 1 usuario grupo 4.2M Sep 15 14:30 imagen.png
```

⚠ En la investigación de incidentes de seguridad, usar `ls -la` es esencial para detectar archivos ocultos maliciosos (.backdoor, .malware) que no aparecerían con un simple `ls`.

Combinación Pro: `ls -la`

Las opciones se pueden combinar. La combinación más útil es `ls -la`, que muestra todos los archivos (incluyendo ocultos) con todos sus detalles:

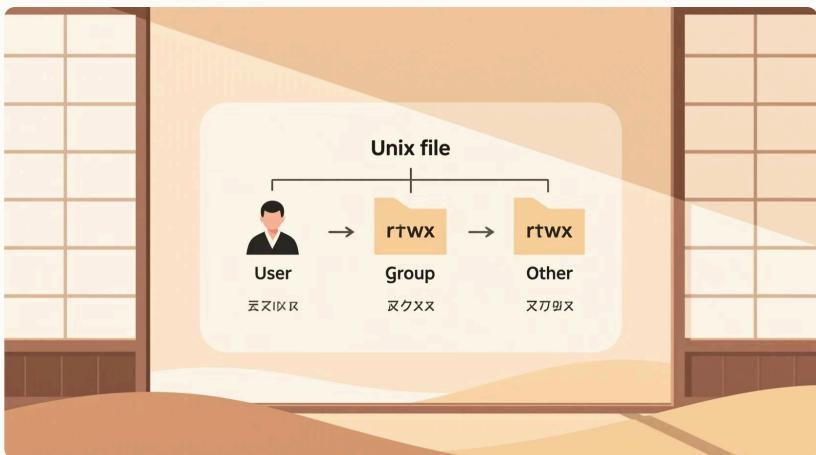
```
usuario@maquina:~$ ls -la  
total 76  
drwxr-xr-x 15 usuario grupo 4096 Sep 15 21:03 .  
drwxr-xr-x 3 root root 4096 Aug 10 09:14 ..  
-rw----- 1 usuario grupo 9807 Sep 15 20:32 .bash_history  
-rw-r--r-- 1 usuario grupo 220 Aug 10 09:14 .bashrc  
drwxr-xr-x 2 usuario grupo 4096 Sep 13 18:45 Documentos  
-rw-r--r-- 1 usuario grupo 807 Aug 10 09:14 .profile  
...
```

Los atacantes suelen usar nombres de archivos que comienzan con "." o nombres que se confunden con comandos legítimos para ocultar su presencia. Siempre verifica lo que no se ve a simple vista.

Aprendé estas opciones de memoria: te ahorrarán tiempo y te darán una visión más completa del sistema. En próximas clases, veremos cómo combinar `ls` con otros comandos para realizar búsquedas avanzadas y filtrar resultados.

Anatomía de un Archivo

Desglosando ls -la



Ejemplo: `-rwxr-xr--`



Tipo de archivo

`-`: Es un archivo regular (no un directorio u otro tipo especial)

Permisos del propietario

`rwx`: El dueño puede leer, escribir y ejecutar este archivo

Permisos del grupo

`r-x`: Los miembros del grupo pueden leer y ejecutar, pero no modificar

Permisos para otros

`r--`: El resto de usuarios solo puede leer el archivo

Las otras columnas

<code>1</code>	Número de enlaces duros al archivo
<code>usuario</code>	El propietario del archivo
<code>grupo</code>	El grupo al que pertenece el archivo
<code>4096</code>	Tamaño en bytes (4KB en este caso)
<code>Sep 15 14:30</code>	Fecha y hora de última modificación
<code>archivo.txt</code>	Nombre del archivo

Implicaciones para la seguridad

Entender los permisos de archivo es crucial en ciberseguridad:

- Privacidad de datos:** Archivos con permisos demasiado abiertos pueden exponer información sensible.
- Integridad del sistema:** Directorios críticos con permisos de escritura para "otros" son vulnerables a modificaciones no autorizadas.
- Ejecución de código:** Archivos ejecutables por cualquier usuario pueden ser vectores de ataque.
- Escalada de privilegios:** Binarios con el bit SUID configurado pueden permitir a atacantes ejecutar comandos como otros usuarios (incluso root).

En la tercera parte de la clase, aprenderemos a modificar estos permisos para mejorar la seguridad del sistema.

Moviéndose por el Sistema

cd

cd → Change Directory

Este comando te permite navegar entre directorios, cambiando tu ubicación actual en el sistema de archivos.

Navegación básica

```
usuario@maquina:~$ cd /var/log  
usuario@maquina:/var/log$ pwd  
/var/log
```

En este ejemplo, nos movemos a la carpeta donde se guardan la mayoría de los logs del sistema.

Subir un nivel

```
usuario@maquina:/var/log$ cd ..  
usuario@maquina:/var$ pwd  
/var
```

El doble punto (..) es una referencia al directorio padre, permitiéndonos subir en la jerarquía de directorios.

Volver al directorio personal

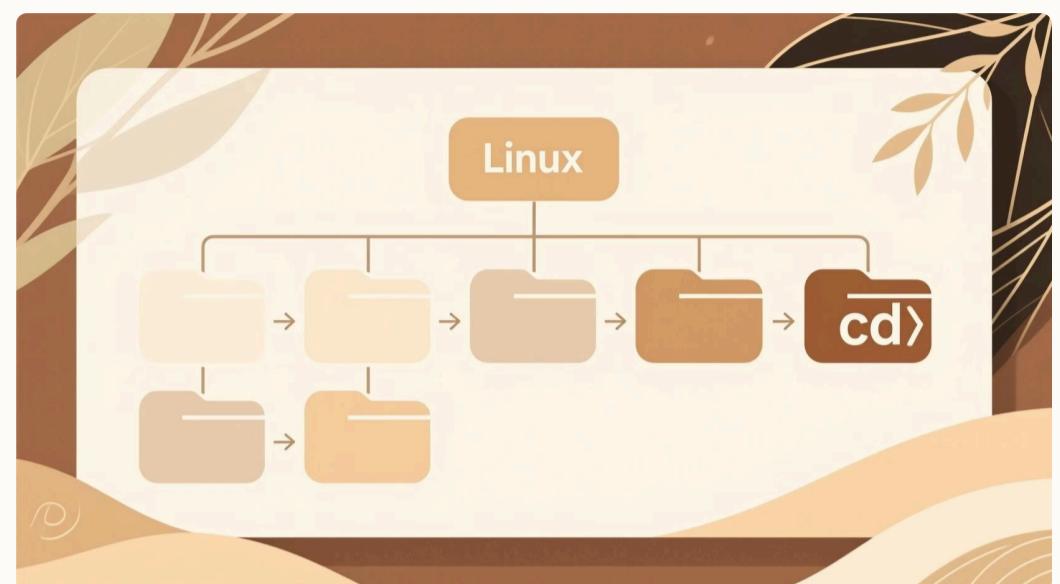
```
usuario@maquina:/var$ cd ~  
usuario@maquina:~$ pwd  
/home/usuario
```

El símbolo ~ (tilde) es un atajo para tu directorio home. También podés usar simplemente cd sin argumentos.

Rutas relativas vs. absolutas

Ruta absoluta: Comienza desde la raíz (/)
cd /home/usuario/Documentos

Ruta relativa: Desde tu ubicación actual
cd Documentos (si estás en /home/usuario)



Trucos avanzados

- `cd -`: Vuelve al directorio anterior (donde estabas antes de tu último cd). Es extremadamente útil para alternar entre dos directorios.
- `cd ~usuario2`: Va al directorio home de otro usuario (si tienes permisos).
- `cd $(find / -name "archivo.txt" -type f -printf "%h\n" 2>/dev/null)`: Cambia al directorio donde se encuentra "archivo.txt" (técnica avanzada).

💡 **Para investigadores de seguridad:** Al explorar un sistema potencialmente comprometido, usa siempre rutas absolutas para los comandos (como `/bin/ls` en lugar de `ls`) para evitar ejecutar versiones maliciosas de comandos comunes que podrían estar en el PATH.

Directarios importantes para ciberseguridad

/var/log

Contiene los archivos de registro del sistema. Esencial para investigación de incidentes y análisis forense.

- `auth.log/secure`: Intentos de inicio de sesión
- `syslog`: Mensajes generales del sistema
- `apache2/access.log`: Accesos al servidor web

/etc

Archivos de configuración del sistema. Un análisis de estos archivos puede revelar configuraciones inseguras o modificaciones maliciosas.

- `passwd, shadow`: Información de usuarios
- `ssh/sshd_config`: Configuración SSH
- `crontab`: Tareas programadas

/tmp

Archivos temporales. Muchos malwares utilizan este directorio para almacenar archivos temporales durante la ejecución. Este directorio suele tener permisos muy abiertos, lo que lo hace atractivo para atacantes.

Combinando cd con pwd y ls, ya tenés las herramientas básicas para explorar cualquier sistema Linux. La práctica constante hará que te muevas por el sistema con fluidez y confianza.

Práctica de Navegación

La tecla Tab

La tecla **Tab** es la herramienta más importante para ser rápido y eficiente en la terminal. Permite autocompletar comandos, nombres de archivo y rutas, ahorrando tiempo y evitando errores tipográficos.

01

Autocompletar comandos

Escribí los primeros caracteres de un comando y presioná Tab:

```
$ pw[TAB] → pwd
```

02

Autocompletar rutas

Escribí parte de una ruta y presioná Tab:

```
$ cd /ho[TAB] → $ cd /home/
```

03

Múltiples coincidencias

Si hay varias posibilidades, presioná Tab dos veces para ver todas las opciones:

```
$ cd Do[TAB][TAB] → Documentos/ Descargas/ Downloads/
```

Ejercicio 1:

1. Abrí una terminal en tu sistema Linux (o WSL si usás Windows).
2. Ejecutá `pwd` para ver dónde estás.
3. Listá el contenido con `ls -la` y familiarizate con lo que ves.
4. Navegá a tu carpeta de Documentos: `cd Documentos` (o `cd Documents` según tu configuración).
5. Creá un directorio para este curso: `mkdir CiberseguridadUNLP`.
6. Entrá en el nuevo directorio: `cd CiberseguridadUNLP`.
7. Creá un archivo de texto: `touch practica1.txt`.

Desafío extra

Para quienes quieran practicar más antes de la próxima clase, intentá estos ejercicios:

1. Explorá el directorio `/etc` y listá su contenido. ¿Cuántos archivos de configuración podés identificar?
2. Navegá a `/var/log` (podés necesitar usar `sudo` para algunos archivos). ¿Qué tipos de logs encontrás?
3. Volvé a tu directorio home usando tres métodos diferentes (`cd ~`, `cd` sin argumentos, y usando una ruta absoluta).
4. Investigá la estructura de directorios en `/proc` y descubrí qué tipo de información del sistema podés encontrar allí.



Tips para la vida real

- **Historial de comandos:** Usá las flechas arriba/abajo para navegar por comandos anteriores.
- **Búsqueda en historial:** Presioná `Ctrl+R` y escribí parte de un comando para buscarlo en tu historial.
- **Detener un comando:** `Ctrl+C` interrumpe la ejecución de un comando.
- **Limpiar pantalla:** `clear` o `Ctrl+L` limpia la terminal.

✓ **Para el próximo encuentro:** Practicá estos comandos hasta que te sientas cómodo/a usándolos. En la próxima clase, aprenderemos a manipular archivos con comandos como `touch`, `cp`, `mv` y `rm`, y a ver su contenido con `cat`, `less` y `head`.

La Estructura con tree

El comando `tree` es una herramienta invaluable para cualquier usuario de Linux, ya que ofrece una representación gráfica y fácil de entender de la jerarquía de archivos y directorios. En lugar de una lista plana, `tree` muestra la estructura como un árbol ramificado, lo que facilita la comprensión de la organización de un proyecto, la ubicación de archivos específicos o la identificación de directorios vacíos.

Es particularmente útil para:

- Entender la arquitectura de proyectos complejos.
- Ver rápidamente el contenido de subdirectorios sin tener que navegar por cada uno.
- Documentar la estructura de directorios para otros usuarios o para referencia futura.

 **Nota de Instalación:** Si `tree` no está disponible en tu sistema, puedes instalarlo fácilmente. En distribuciones basadas en Debian/Ubuntu, usa:
`sudo apt install tree`.



Construyendo tu Entorno: mkdir y touch

mkdir <nombre_carpeta>

Make Directory. Este comando es tu constructor principal en la línea de comandos. Te permite crear nuevas carpetas o directorios donde almacenar tus archivos. Puedes crear múltiples directorios a la vez, o usar la opción `-p` para crear directorios padres si no existen.

Ejemplos prácticos:

- `mkdir proyectos`: Crea un nuevo directorio llamado 'proyectos'.
- `mkdir -p docs/manuales/referencia`: Crea 'docs', y dentro 'manuales', y dentro 'referencia'.

touch <nombre_archivo>

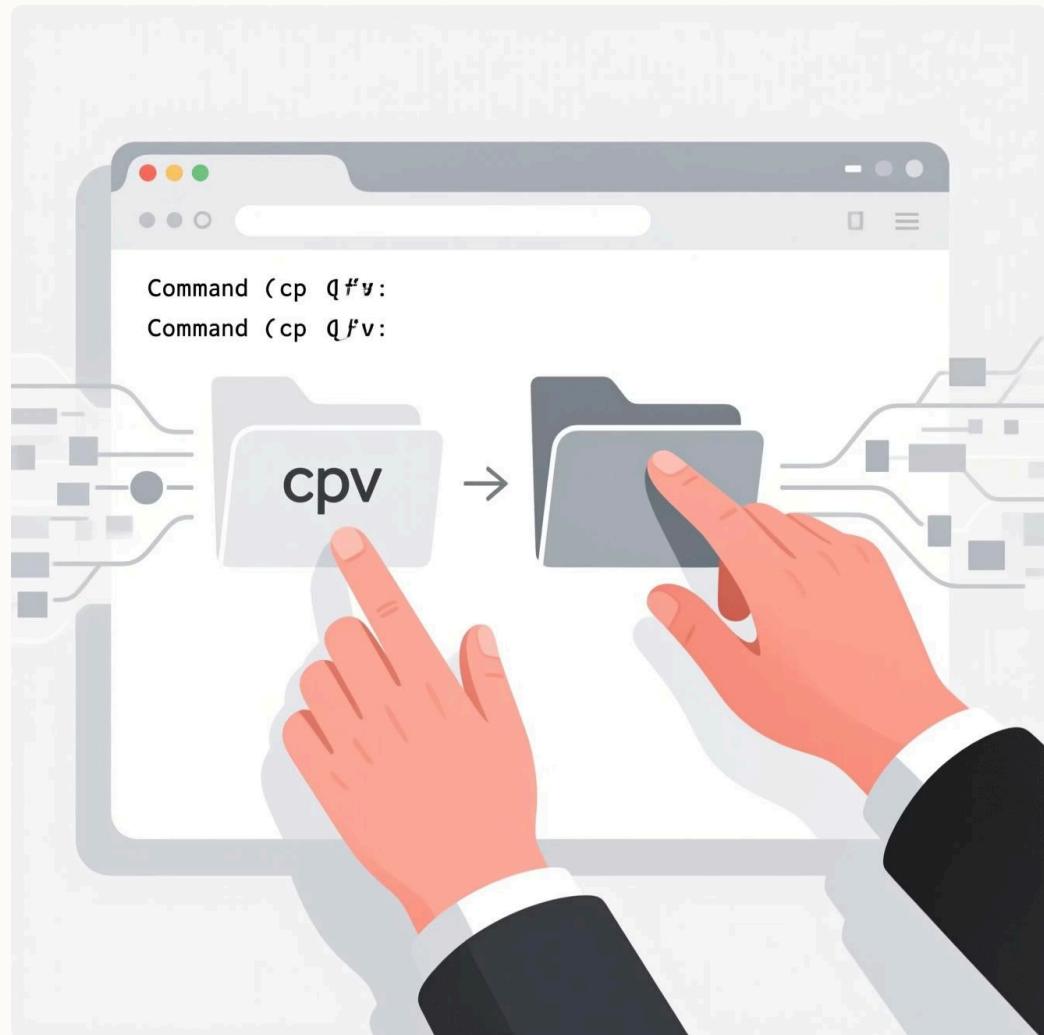
El comando `touch` tiene dos funciones principales: crear un archivo vacío si no existe, o actualizar la fecha y hora de la última modificación de un archivo existente. Es extremadamente útil para scripts que dependen de estas marcas de tiempo.

Ejemplos prácticos:

- `touch mi_documento.txt`: Crea un archivo vacío llamado 'mi_documento.txt'.
- `touch informe.pdf`: Si 'informe.pdf' ya existe, actualiza su fecha de modificación.

Dominar `mkdir` y `touch` te proporciona los cimientos para organizar eficazmente tu espacio de trabajo en el terminal, permitiéndote estructurar tus datos de forma lógica desde el principio.

Copiar, Mover y Renombrar: cp y mv



cp <origen> <destino>

Copy. Este comando te permite duplicar archivos o directorios. Es fundamental para hacer copias de seguridad o para trabajar con versiones diferentes de un mismo archivo sin alterar el original.

- `cp archivo.txt copia_archivo.txt`: Crea una copia del archivo.
- `cp -r carpeta/ nueva_carpeta/`: Copia directorios recursivamente.

mv <origen> <destino>

Move. El comando mv tiene una doble funcionalidad. Primero, te permite mover archivos y directorios de una ubicación a otra. Segundo, si el destino es un nuevo nombre en la misma ubicación, actúa como un comando para renombrar.

- `mv documento.pdf informes/`: Mueve 'documento.pdf' a la carpeta 'informes'.
- `mv foto_vieja.jpg foto_nueva.jpg`: Renombra un archivo.

Estos comandos son el pan de cada día para cualquier tarea de gestión de archivos, asegurando que puedas organizar tu información de la manera más eficiente y segura posible.

Eliminación Permanente con rm

¡ADVERTENCIA MÁXIMA!

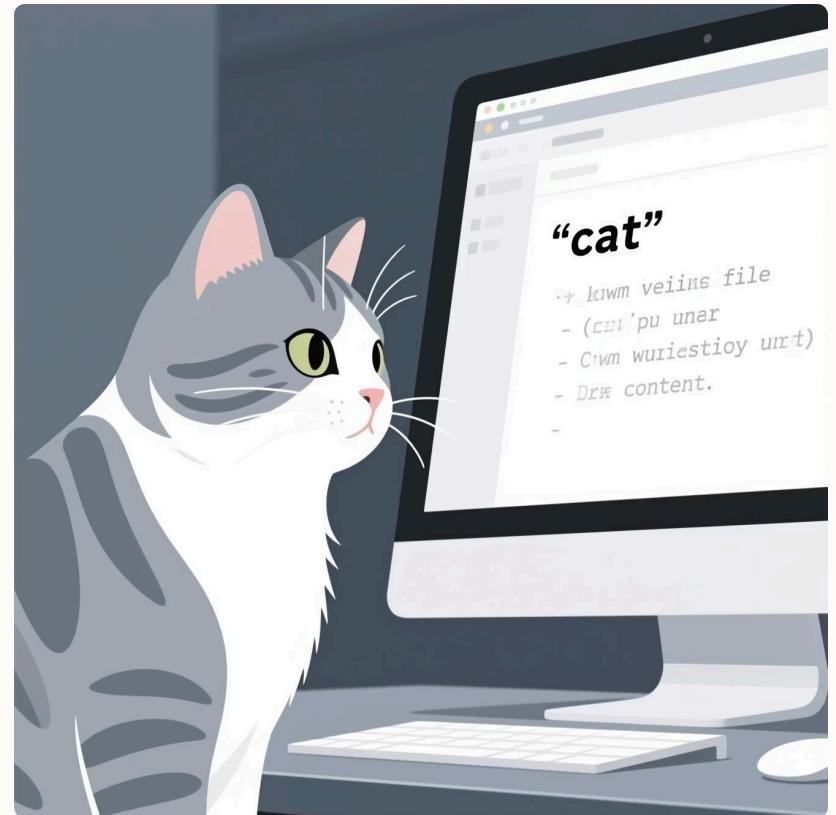
El comando `rm` (del inglés **remove**) se utiliza para eliminar archivos y directorios. Si bien es una herramienta poderosa para liberar espacio y mantener tu sistema limpio, su uso debe ser extremadamente cauteloso. A diferencia de los entornos gráficos, la línea de comandos de Linux no cuenta con una "papelera de reciclaje" por defecto para los archivos eliminados con `rm`. Una vez que un archivo es borrado con este comando, su recuperación puede ser muy difícil o incluso imposible.

Considera siempre las siguientes precauciones:

- **Verificación:** Antes de ejecutar `rm`, verifica dos veces el nombre del archivo o la ruta para asegurarte de que estás eliminando lo correcto.
- **Opción -i:** Utiliza `rm -i <archivo>` para que el sistema te pida confirmación antes de eliminar cada archivo. Esto puede salvarte de errores accidentales.
- **Borrado Recursivo (-r):** Cuando necesites eliminar directorios y su contenido, usarás `rm -r <carpeta>`. Ten especial cuidado con esta opción, ya que borrará todo dentro del directorio especificado sin piedad.
- **Evitar `rm -rf` o `rm -rf *`:** ¡Nunca ejecutes estos comandos a menos que sepas exactamente lo que haces! Pueden borrar todo tu sistema o todos los archivos en tu directorio actual sin pedir confirmación, respectivamente.

La irreversibilidad de `rm` subraya la importancia de la precisión y el cuidado al trabajar en la línea de comandos. Siempre piensa dos veces antes de eliminar.

Leyendo Archivos con cat



El comando `cat` (abreviatura de **concatenate**) es una de las utilidades más básicas y frecuentemente usadas en Linux. Su función principal es mostrar el contenido de uno o más archivos directamente en la salida estándar (normalmente, tu terminal). Aunque su nombre sugiere la concatenación, su uso más común es simplemente para ver el contenido de un archivo.

Es ideal para:

- **Archivos Cortos:** Si el archivo es pequeño y cabe en una o dos pantallas, `cat` es perfecto para obtener un vistazo rápido sin necesidad de abrir un editor.
- **Combinar Archivos:** Puedes usar `cat` para combinar el contenido de varios archivos en uno solo, o para dirigir su salida a otro comando (lo que se conoce como 'piping').
- **Crear Archivos:** Aunque menos común hoy en día, `cat` también puede usarse para crear archivos de texto directamente desde la entrada de la terminal (Ctrl+D para terminar la entrada).

```
cat mi_informe.txt
```

Este comando es el primer paso para inspeccionar el contenido de tus archivos y entender lo que contienen antes de realizar modificaciones o procesamientos más complejos.

Lectura Controlada: Navegando Archivos Largos con less y more

Cuando te enfrentas a archivos de texto que son demasiado grandes para caber en una sola pantalla del terminal, `cat` no es la herramienta adecuada, ya que el contenido se desplazará rápidamente. Para estos casos, Linux ofrece visores de página que te permiten examinar el contenido de forma controlada, página por página o línea por línea. Los dos más comunes son `less` y `more`.

less <archivo>

`less` es el visor de página moderno y generalmente preferido. Ofrece una flexibilidad superior para navegar por archivos:

- **Navegación Intuitiva:** Puedes usar las flechas del teclado (arriba/abajo) para moverte línea por línea, la barra espaciadora para avanzar una página completa y 'b' para retroceder una página.
- **Búsqueda:** Presiona '/' seguido de un término para buscar hacia adelante, y '?' para buscar hacia atrás. Presiona 'n' para la siguiente ocurrencia.
- **Sin Carga Previa:** `less` carga el archivo a medida que lo necesitas, lo que lo hace muy eficiente para archivos extremadamente grandes.
- **Salir:** Para salir de `less`, simplemente presiona 'q'.

more <archivo>

`more` es una utilidad más antigua y básica que `less`. Su funcionalidad es más limitada, pero sigue siendo útil en algunos contextos:

- **Avance Página a Página:** Solo puedes avanzar hacia adelante, una pantalla a la vez, presionando la barra espaciadora.
- **Salir:** Presiona 'q' para salir. Una vez que has llegado al final del archivo, `more` se cerrará automáticamente.

Para la mayoría de los propósitos, `less` es la opción más potente y flexible para leer archivos grandes. Sin embargo, ambos comandos son herramientas esenciales en tu arsenal de la línea de comandos.

Espiando Archivos con head y tail

1 head <archivo>

El comando `head` te permite ver las primeras líneas de un archivo. Por defecto, muestra las primeras 10 líneas. Es extremadamente útil para obtener una idea rápida del formato o del contenido inicial de un archivo sin tener que leerlo por completo.

Ejemplo: `head /var/log/syslog` te mostrará el inicio del registro del sistema.

2 tail <archivo>

A diferencia de `head`, `tail` muestra las últimas líneas de un archivo (por defecto, también 10). Este comando es crucial para revisar los registros (logs) de aplicaciones, donde los mensajes más recientes suelen estar al final.

Ejemplo: `tail ~/.bash_history` te mostrará los últimos comandos que ejecutaste en tu terminal.



ⓘ Control de Líneas:

Puedes especificar el número de líneas que deseas ver usando la opción `-n <número>` con ambos comandos.

- `head -n 5 mi_documento.txt`: Muestra las primeras 5 líneas.
- `tail -n 20 archivo_log.log`: Muestra las últimas 20 líneas.

Estos comandos son particularmente valiosos en escenarios de depuración y monitoreo, permitiéndote aislar la información relevante de manera eficiente.

Monitoreo de Registros: tail -f



Una de las funcionalidades más poderosas de tail es su capacidad para monitorear archivos en tiempo real. Esto se logra con la opción **-f** (de **follow**).

```
tail -f <archivo>
```

Cuando ejecutas tail -f sobre un archivo, el comando no termina después de mostrar las últimas líneas. En su lugar, permanece activo y sigue "espiando" el archivo, mostrando cualquier nueva línea que se añada a él en tiempo real. Esto es absolutamente fundamental para:

- **Monitoreo de Logs:** Ver los registros de un servidor web, una base de datos o una aplicación mientras está en funcionamiento, permitiéndote identificar errores o comportamientos inesperados al instante.
- **Depuración:** Si estás desarrollando una aplicación, puedes usar tail -f en el archivo de log para ver los mensajes de depuración a medida que tu código los genera.
- **Seguimiento de Eventos:** Observar la actividad del sistema o de la red en vivo.

Para detener el monitoreo y salir del comando tail -f, simplemente presiona **Ctrl + C**.

Esta característica convierte a tail -f en una herramienta indispensable para administradores de sistemas y desarrolladores, proporcionando una ventana directa a la actividad continua de tus aplicaciones y servicios.

nano



nano es un editor de texto basado en la terminal que se distingue por su simplicidad y facilidad de uso, especialmente para principiantes en la línea de comandos. A diferencia de editores más complejos como Vim o Emacs, nano te permite comenzar a editar archivos de texto casi de inmediato sin una curva de aprendizaje pronunciada.

`nano <archivo>`

Cuando abres un archivo con nano (o creas uno nuevo si no existe), verás el contenido del archivo y, lo más importante, una lista de atajos de teclado útiles en la parte inferior de la pantalla. Estos atajos están prefijados con `^` (que significa la tecla Ctrl) o `M-` (que significa la tecla Alt).

Atajos Esenciales:

- `^O (Ctrl+O)`: Guardar el archivo.
- `^X (Ctrl+X)`: Salir del editor. Te preguntará si quieras guardar los cambios si no lo has hecho ya.
- `^K (Ctrl+K)`: Cortar línea.
- `^U (Ctrl+U)`: Pegar línea.
- `^W (Ctrl+W)`: Buscar texto.

Ejercicio 2: Manipulación de archivos



¡Hora de practicar!

Aplica lo aprendido con el siguiente ejercicio, consolidando tus conocimientos en la línea de comandos:

1. **Crea un archivo:** Abre nano y crea un nuevo archivo llamado `mi_primer_archivo.txt`.
2. **Escribe y guarda:** Escribe algunas líneas de texto dentro del archivo (por ejemplo, tu nombre, la fecha y "¡Esto es Linux!"). Guarda los cambios y sal de nano.
3. **Copia el archivo:** Haz una copia de `mi_primer_archivo.txt` llamándola `copia_archivo.txt`.
4. **Renombra la copia:** Renombra `copia_archivo.txt` a `archivo_renombrado.txt`.
5. **Elimina el archivo renombrado:** Borra `archivo_renombrado.txt`. (¡Con cuidado!)
6. **Verifica:** Usa `ls` para confirmar que el archivo renombrado ha desaparecido, y que `mi_primer_archivo.txt` sigue ahí.



Permisos en Linux: La Clave de la Seguridad y el Control

B

El Pilar de la Seguridad

En el corazón de la seguridad de Linux reside su robusto sistema de permisos. Cada archivo y directorio dentro del sistema de archivos tiene un conjunto de reglas que dictan quién puede interactuar con él y de qué manera.

Imagina estos permisos como las "llaves" que abren o cierran el acceso a tus datos. Sin una comprensión clara de estos, tu sistema podría ser vulnerable o, por el contrario, demasiado restrictivo, impidiendo el flujo de trabajo.

Este sistema de control de acceso es fundamental para proteger la integridad, confidencialidad y disponibilidad de la información en entornos multiusuario y servidores.



Cada archivo tiene reglas sobre quién puede leerlo, modificarlo o ejecutarlo. Este es el sistema de control de acceso fundamental de Linux.

Leer, Escribir, Ejecutar: El Modelo rwx



r (Read) - Lectura

El permiso de lectura permite ver el contenido de un archivo. Para un directorio, permite listar su contenido (ver qué archivos hay dentro).

- Acceso a datos de archivos.
- Listado de directorios.



w (Write) - Escritura

El permiso de escritura permite modificar, guardar cambios o eliminar un archivo. Para un directorio, permite crear, eliminar o renombrar archivos dentro de él (incluso si no tienes permisos de escritura sobre el archivo en sí).

- Modificación de archivos.
- Creación/Eliminación en directorios.



x (eXecute) - Ejecución

El permiso de ejecución permite ejecutar un archivo si es un programa o script. Para un directorio, permite acceder a él, atravesarlo (entrar con `cd`) y acceder a sus subdirectorios.

- Ejecución de programas/scripts.
- Navegación de directorios.

Estos tres permisos forman la base del control de acceso en Linux, permitiendo una gestión precisa sobre las interacciones con cada elemento del sistema de archivos.

Las Identidades: Usuario, Grupo y Otros

Los permisos `rwx` no se aplican universalmente, sino que se asignan a diferentes niveles o identidades de usuarios. Esta segmentación es clave para la seguridad en sistemas multiusuario.



Usuario (u)

Es el propietario del archivo o directorio. Solo puede haber un usuario propietario a la vez. Cuando creas un archivo, tú eres, por defecto, el usuario propietario.

Ejemplo: Un documento personal solo editable por ti.



Grupo (g)

Un archivo también pertenece a un grupo. Este grupo puede contener múltiples usuarios. Todos los miembros de ese grupo comparten los permisos asignados al grupo.

Ejemplo: Archivos de un proyecto compartidos entre desarrolladores.



Otros (o)

Esta categoría incluye a todos los demás usuarios en el sistema que no son el propietario del archivo ni pertenecen al grupo propietario del archivo. También se les conoce como "el resto del mundo".

Ejemplo: Un archivo público en un servidor web.

Entender estas tres identidades es fundamental para aplicar permisos de forma efectiva y segura, garantizando que solo las personas autorizadas tengan el nivel de acceso adecuado.

chmod: El Método Intuitivo (Simbólico)

El comando `chmod` es la herramienta principal para cambiar los permisos de archivos y directorios. El método simbólico es particularmente intuitivo, ya que utiliza letras y símbolos para añadir, quitar o asignar permisos.

```
chmod [identidad][operador][permiso] [archivo]
```

- **Identidad:** `u` (usuario), `g` (grupo), `o` (otros), `a` (todos).
- **Operador:**
 - `+`: Añadir permiso.
 - `-`: Quitar permiso.
 - `=`: Asignar permisos específicos (ignorando los existentes).
- **Permiso:** `r` (lectura), `w` (escritura), `x` (ejecución).

Este método es excelente para ajustes rápidos y específicos, ya que te permite pensar en términos de "qué quiero añadir" o "qué quiero quitar" para cada identidad.

Ejemplos Prácticos:

- `u+x mi_script.sh`
Al **Usuario**, **añádele** ejecución.
- `g-w documento.txt`
Al **Grupo**, **quítale** escritura.
- `o=r publico.html`
A los **Otros**, **asígnale solo** lectura.
- `a+rw carpeta/`
A todos, añádeles lectura y escritura.

chmod: El Método Rápido (Numérico/Octal)

El método numérico de `chmod` es una forma más concisa y potente de establecer permisos, utilizando números octales (base 8). Cada permiso `rwx` tiene un valor numérico asignado:

- **r (Read): 4**
- **w (Write): 2**
- **x (eXecute): 1**
- **Ningún permiso: 0**

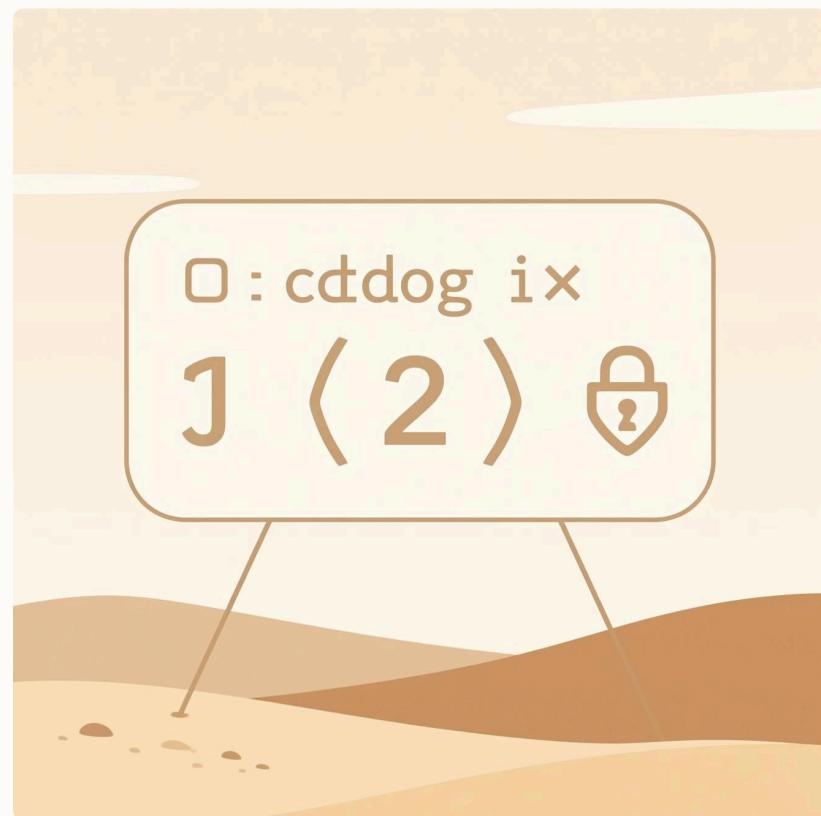
Para establecer los permisos, se suman los valores de los permisos deseados para cada una de las tres identidades (Usuario, Grupo, Otros) y se forma un número de tres dígitos. El primer dígito es para el Usuario, el segundo para el Grupo y el tercero para Otros.

Ejemplo: `chmod 755 mi_script.sh`

- **Primer dígito (Usuario): 7 (4+2+1) → rwx (lectura, escritura, ejecución)**
- **Segundo dígito (Grupo): 5 (4+0+1) → r-x (lectura, ejecución)**
- **Tercer dígito (Otros): 5 (4+0+1) → r-x (lectura, ejecución)**

Este método es ideal cuando necesitas establecer un conjunto completo de permisos de una sola vez y es muy común en scripts y automatizaciones.

Permiso	Simbólico	Valor
Lectura	r	4
Escritura	w	2
Ejecución	x	1
Ninguno	-	0



chown

Mientras que `chmod` gestiona los permisos de acceso, el comando `chown` (del inglés **c**hange **o**wner) se encarga de cambiar el usuario propietario y/o el grupo propietario de un archivo o directorio.

```
chown [nuevo_dueño] [archivo]
```

Este comando es esencial en escenarios donde los archivos necesitan ser transferidos a otros usuarios o cuando los servicios del sistema requieren que ciertos archivos sean propiedad de usuarios específicos (ej. `www-data` para archivos web).

Para cambiar el grupo propietario de un archivo, también puedes usar `chgrp [nuevo_grupo] [archivo]`. Sin embargo, `chown` ofrece una sintaxis más versátil para manejar ambos a la vez:

```
chown [dueño]:[grupo] [archivo]
```

✖ Importante!

Cambiar el propietario de un archivo o directorio es una operación que a menudo requiere privilegios de superusuario. Por lo tanto, generalmente necesitarás usar `sudo` antes del comando `chown`.

```
sudo chown miusuario:migrupo mi_archivo.txt
```



La correcta asignación de propiedad es tan crucial como los permisos en sí, ya que define quién tiene la autoridad para modificar incluso los permisos de un archivo.

Hoja de Referencia

A continuación, una tabla con los comandos clave que hemos explorado para la gestión de permisos y propietarios en Linux.

Comando	Sintaxis Principal	Descripción
ls -l	ls -l [archivo/directorio]	Muestra detalladamente los permisos, propietario, grupo, tamaño y fecha de modificación.
chmod	chmod [permisos] [archivo]	Cambia los permisos de un archivo o directorio. Puede usar método simbólico (u+x) o numérico (755).
chown	chown [usuario][:grupo] [archivo]	Cambia el usuario propietario y opcionalmente el grupo propietario de un archivo o directorio.
chgrp	chgrp [grupo] [archivo]	Cambia el grupo propietario de un archivo o directorio.
whoami	whoami	Muestra el nombre de usuario actual.
groups	groups [usuario]	Muestra los grupos a los que pertenece un usuario (o el actual si no se especifica).

¡Es lo que tenemos por hoy!



En la Próxima Clase...

- **Privilegios (sudo):** Cómo ejecutar comandos como superusuario de forma segura.
- **Procesos (htop):** Monitorizar y gestionar los programas en ejecución.
- **Búsqueda (grep, find):** Encontrar archivos y texto dentro de ellos con potentes herramientas.
- **Introducción a Redes:** Conceptos básicos de red y herramientas para diagnosticar conexiones.

Gracias :)

by Qb1t::