PROMETEO

Unidad 2: Linux: fundamentos y comandos básicos

Vas a distinguir dos ideas clave: qué es el kernel y cómo se organiza el "sistema" alrededor de él. **Linux, estrictamente hablando, es el kernel**: el núcleo que habla con la CPU, la RAM, los discos y los dispositivos.

Sesión 4 – Kernel y estructura de Linux. Diferencias con Windows

Vas a distinguir dos ideas clave: qué es el kernel y cómo se organiza el "sistema" alrededor de él. **Linux, estrictamente hablando, es el kernel**: el núcleo que habla con la CPU, la RAM, los discos y los dispositivos. Hace de "intérprete" entre el hardware y todo lo demás: planifica procesos, asigna memoria, gestiona drivers y expone interfaces para que el software trabaje sin preocuparse del metal.

Ese kernel Linux es **monolítico con módulos cargables**: integra gran parte de la lógica en el propio núcleo, pero permite añadir o quitar funcionalidades en caliente (por ejemplo, un driver de red) mediante módulos.

A ese kernel se le suma el **espacio de usuario (userland)**: utilidades GNU (coreutils, bash, tar…), el sistema de inicio (hoy, en la mayoría de distros, systemd), bibliotecas (glibc, openssl), gestores de paquetes (apt, dnf, pacman), servicios/daemons y, si quieres, un entorno gráfico (X11/Wayland + GNOME/KDE).

El resultado es una **distribución Linux** (Ubuntu, Fedora, Debian, openSUSE…), cada una con su combinación de paquetes, repositorios y ciclos de actualización. Todo ello respeta un estándar de organización de archivos (FHS) que ubica binarios en /bin y /usr/bin, configuraciones en /etc, datos variables en /var, perfiles de usuario en /home, y expone el hardware como archivos en /dev, /proc y /sys.

¿En qué se diferencia esto de Windows? Windows se distribuye como un sistema operativo integrado y propietario con un kernel NT (arquitectura "híbrida"), su API Win32/.NET, registro (Registry) centralizado, servicios, instaladores MSI/EXE y su Microsoft Store.

Linux te ofrece:

- **Transparencia** (todo está en archivos legibles)
- **Control fino** (permisos, servicios, paquetes)
- **Modularidad** (eliges shell, entorno gráfico, servidor gráfico, etc.)

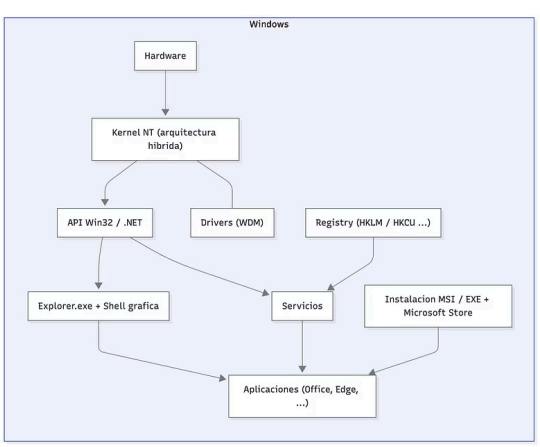
Windows te ofrece:

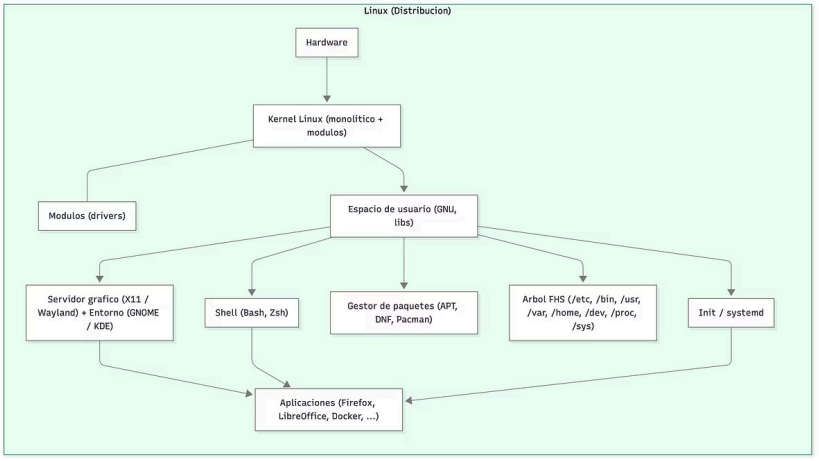
- Una experiencia unificada out-of-the-box
- Ecosistema comercial masivo (drivers y software propietario abundante)
- Herramientas administrativas integradas (Group Policy, Active Directory en entornos corporativos)

Entender ambas filosofías te permitirá elegir qué conviene para servidor, escritorio, aula o desarrollo.

2. Esquema Visual

A continuación tienes un diagrama conceptual para recrear la arquitectura y las diferencias clave. Cópialo tal cual para generarlo y luego revisa la descripción de cada elemento.





Qué representa cada elemento y cómo recrearlo fielmente:

- ## Linux / Windows

  Dos subgráficos separados para dejar clara la comparación.

- ## Hardware

  Base física. En Linux (LHW) y Windows (WHW) apunta al kernel.

- ## Kernel

  LKR (Linux monolítico con módulos) vs WKR (Windows NT híbrido). Dibuja una línea discontinua (---) hacia "Módulos/Drivers" porque se cargan dinámicamente.

- ## Espacio de usuario / API

  En Linux (LUS) agrupa GNU, bibliotecas y herramientas; en Windows (WAPI) agrupa Win32/.NET.

- ## Inicio/Servicios

  LINIT (systemd u otro init) lanza daemons y sesiones; WSRV son servicios de Windows.

- ## Shell

  LSH (Bash/Zsh) como interfaz CLI; en Windows el shell gráfico principal es Explorer.exe (WGUI).

- ## Gestión de software

  LPKG (repositorios y gestor de paquetes) vs WPKG (instaladores MSI/EXE o Store).

- ## Sistema de archivos / Registry

  LFS (FHS, archivos planos) vs WREG (registro centralizado).

- ## Gráfico

  LGUI separa servidor gráfico y entorno; WGUI integra shell y escritorio.

- ## Aplicaciones

  En ambos, capa superior de uso.

Caso de Estudio – Ubuntu (Canonical)

Contexto

Ubuntu es una distribución basada en Debian y mantenida por Canonical. Su propuesta es clara: **versiones LTS (Long Term Support) con 5 años de soporte**, un escritorio estándar (GNOME) y un ecosistema de paquetes vía APT (repositorios main, universe, multiverse) y, adicionalmente, snaps para paquetes autocontenidos. Esta combinación la ha convertido en elección frecuente para escritorios educativos, laboratorios de desarrollo y, sobre todo, servidores y cloud (imágenes oficiales en AWS, Azure y Google Cloud son habituales).

Estrategia

Modularidad real

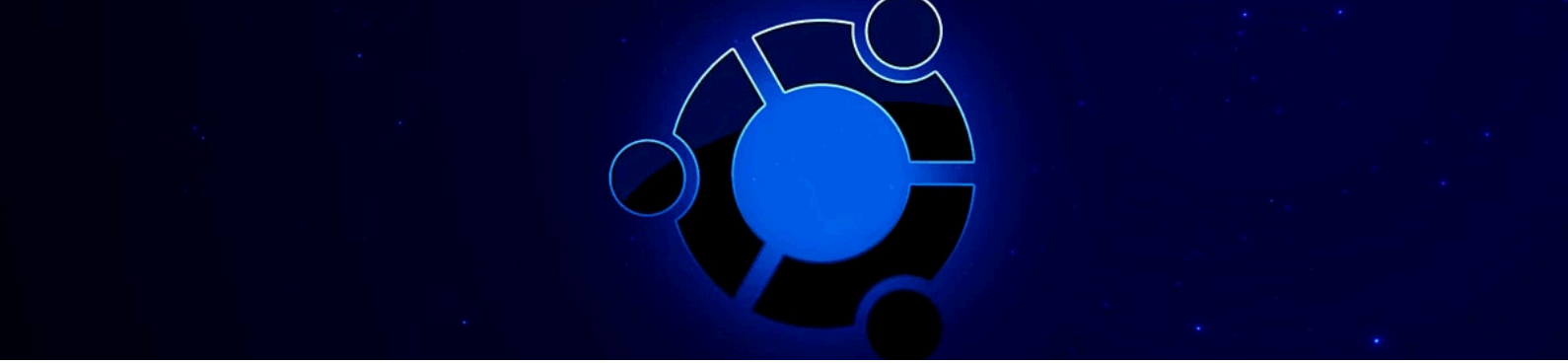
El kernel Linux de Ubuntu se distribuye con módulos, lo que permite soporte de hardware amplio. Puedes listar el kernel con `uname -a` y los módulos con `lsmod`. Si falta un driver, puedes cargarlo con `modprobe`.

Userland predecible

Los metapaquetes y tareas simplifican la instalación (por ejemplo, ubuntu-desktop para un entorno GNOME completo).

Gestión de paquetes controlada

Actualizaciones con `apt update && apt upgrade` y versiones concretas con pinning (apt preferences). El usuario decide qué repositorios activar (oficiales, PPA de terceros, snaps) y cuándo actualizar.



Arquitectura transparente

Configuraciones en /etc (texto plano), logs en /var/log, servicios gestionados con `systemctl` (systemd).

Seguridad por diseño

Permisos POSIX, sudo para privilegios temporales, y perfiles AppArmor activos por defecto en muchas instalaciones para confinar servicios.

Entornos

Mismo kernel base, pero roles distintos: servidor sin GUI (menor huella), escritorio con Wayland (o Xorg si se requiere), contenedores con LXD o Docker.

Resultado

Personalización consciente

Tú eliges: GNOME, KDE, XFCE o sin GUI; apt o snap; Wayland o Xorg.

Mantenibilidad

Con LTS reduces sorpresas y alineas ciclos de proyecto (ideal en aulas, laboratorios y servidores de prácticas).

Eficiencia y coste

Al no requerir licencias por equipo para el sistema base, los despliegues en aulas o laboratorios cierran el círculo: más máquinas, menos coste; más control, mismas competencias.

Este caso contrasta con Windows, donde la actualización (Windows Update), la gestión centralizada de políticas (GPO) y la distribución comercial aportan una experiencia unificada y soporte de software propietario sobresaliente, pero con menor capacidad de "cambiar piezas" del sistema. En Ubuntu, la modularidad es parte del día a día.

Herramientas y Consejos

Conoce tu kernel y tu distro

- `uname -a`: versión del kernel y arquitectura.
- `lsb_release -a` o lectura de `/etc/os-release`: identifica la distribución y su versión.
- `neofetch` o `fastfetch`: resumen visual (útil en aula para acelerar diagnósticos).

Explora el hardware y los módulos

- `lshw`, `lspci`, `lsusb`: inventario de hardware.
- `lsmod`, `modinfo <módulo>`, `modprobe <módulo>`: ciclo de vida de drivers en Linux.
- En Windows, lo equivalente sería Administrador de dispositivos y `pnputil`. Úsalo para comparar enfoques.

Practica sin riesgo

- VirtualBox o VMware Workstation Player: instala Ubuntu/Fedora en VM. Crea snapshots antes de grandes cambios.
- WSL (Windows Subsystem for Linux): si tu PC principal es Windows, instala Ubuntu en WSL para practicar shell, apt y estructura sin tocar particiones.
- DistroWatch (comparativo) y Ubuntu Server Live: pruebas rápidas.

Paquetes y servicios como hábito

- `apt search`/`apt show` para investigar paquetes; `apt-mark hold` para "congelar" versiones críticas.
- `systemctl status` y `journalctl -u` para depurar servicios.
- En Windows, familiarízate con `services.msc`, Event Viewer y `winget` para obtener un contraste útil.

Mapea el árbol del sistema de archivos

- Recorre `/etc` (config), `/var/log` (logs), `/home` (datos usuario), `/tmp` (temporales), `/dev` (dispositivos), `/proc` y `/sys` (info del kernel).
- Practica con `tree /etc` (instala tree) para "ver" la jerarquía. Esta visión te dará una ventaja enorme frente a la opacidad del Registry.

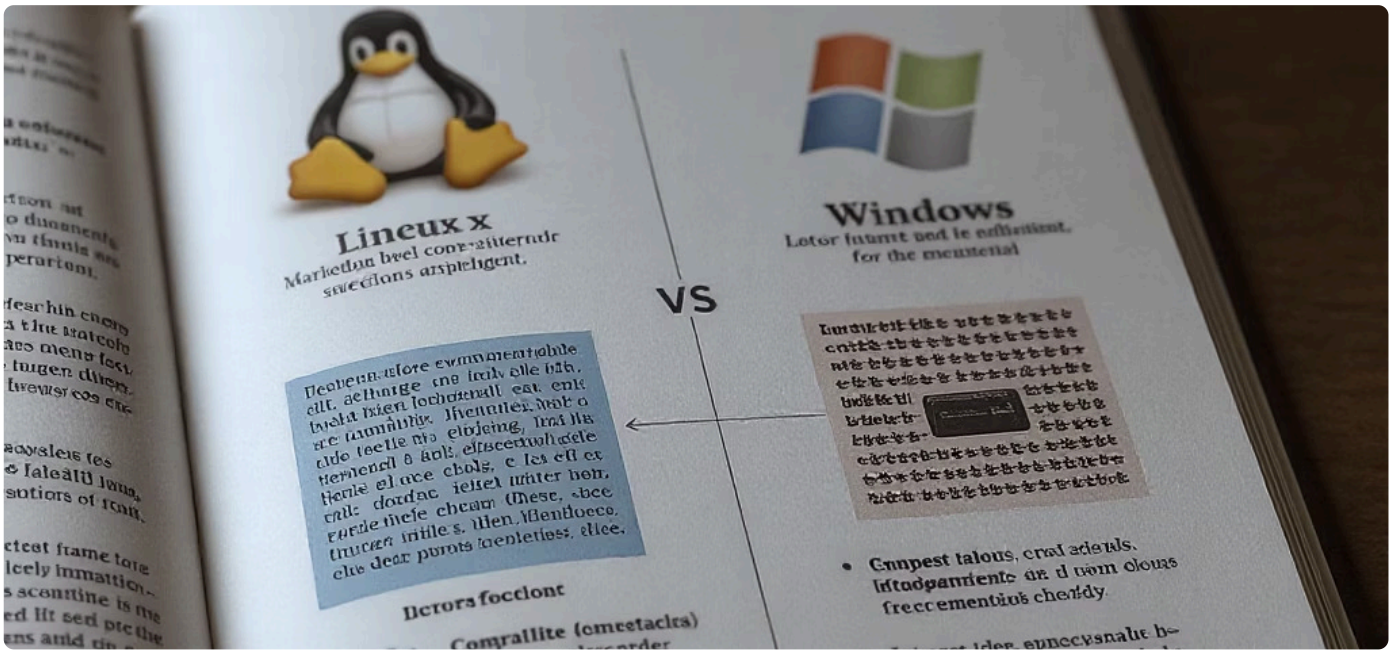
Mitos y Realidades

❌ **Mito:** "Linux es solo para programadores." → **FALSO.** Hoy puedes usar Ubuntu, Linux Mint o Fedora en escritorio con asistentes gráficos para todo (drivers, Wi-Fi, impresoras, actualizaciones). La terminal es una herramienta poderosa, no un requisito excluyente. Lo que sí ocurre es que, si aprendes algunos comandos, ganarás velocidad y control.

❌ **Mito:** "Linux no tiene interfaz gráfica." → **FALSO.** Linux puede trabajar sin GUI (ideal para servidores) o con escritorios modernos (GNOME, KDE Plasma, XFCE, Cinnamon). Puedes elegir el entorno que mejor rinda en tu hardware y cambiarlo cuando quieras, algo mucho menos flexible en Windows.

Resumen Final

- Kernel = núcleo que gestiona hardware y procesos; Linux (kernel) + userland + paquetes + GUI ⇒ **distribución**.
- Linux es **modular, abierto y transparente** (archivos de texto, FHS, paquetes, servicios); Windows es **propietario e integrado** (NT, Registry, MSI/Store).
- En Linux decides **qué** instalar y **cuándo**; en Windows prima la **experiencia unificada** y la compatibilidad comercial.
- Domina uname -a, lsb_release -a, FHS y gestores de paquetes: base práctica para moverte con solvencia entre ambos mundos.



En los sistemas Linux, la terminal o línea de comandos es una herramienta esencial que permite interactuar directamente con el sistema operativo sin necesidad de interfaz gráfica. Aunque las distribuciones modernas ofrecen entornos visuales muy completos, la verdadera potencia de Linux se encuentra en su terminal, especialmente en tareas de administración, automatización y resolución de problemas.

Los comandos de navegación son el punto de partida para todo usuario que quiera desenvolverse con soltura en Linux. Permiten desplazarte entre carpetas, listar archivos y consultar su contenido con rapidez y precisión. Son el equivalente textual al explorador de archivos, pero mucho más flexible.

El concepto de directorio actual

Cada vez que abres una terminal, te encuentras en un directorio concreto del sistema, conocido como **directorio actual** o *working directory*. Para conocer tu ubicación exacta, se utiliza el comando:

```
pwd
```

Este comando significa *print working directory* y devuelve la ruta completa desde la raíz del sistema. Por ejemplo:

```
/home/usuario/Documentos
```

Esto indica que estás dentro de la carpeta “Documentos” del usuario actual. Esta información es fundamental, porque muchas acciones —como copiar, mover o editar archivos— dependen de saber en qué directorio te encuentras.

Listar contenido con ls

El comando `ls` (list) muestra el contenido de un directorio. Por defecto, lista archivos y subcarpetas del directorio actual:

```
ls
```

Puedes especificar otra ruta (`ls /etc`) o añadir opciones para más detalle:

- `-l`: información detallada (permisos, tamaño, fecha).
- `-a`: muestra archivos ocultos (empiezan con `.`).
- `-h`: tamaños legibles (KB, MB).
- `-R`: lista de forma recursiva, incluyendo subdirectorios.

Una vista completa y legible se obtiene con:

```
ls -lha
```

Cambiar de directorio con cd

El comando `cd` (change directory) te permite moverte por la estructura de carpetas, como hacer doble clic en un entorno gráfico. Ejemplo básico:

```
cd /home/usuario/Descargas
```

Otros usos frecuentes:

- `cd ..`: sube un nivel (al directorio padre).
- `cd /`: te lleva a la raíz del sistema.
- `cd` (sin argumentos): vuelve al directorio personal.
- `cd -`: vuelve al directorio anterior.

Confirma siempre tu ubicación con `pwd`.

`cat` y `less`: Ver Archivos

Una vez localizado un archivo, puedes visualizar su contenido directamente en la terminal:

`cat` (concatenate): muestra el contenido completo de un archivo de texto. Ideal para archivos cortos o unir varios (ej.

```
cat notas.txt
```

o

```
cat p1.txt p2.txt > completo.txt
```

).

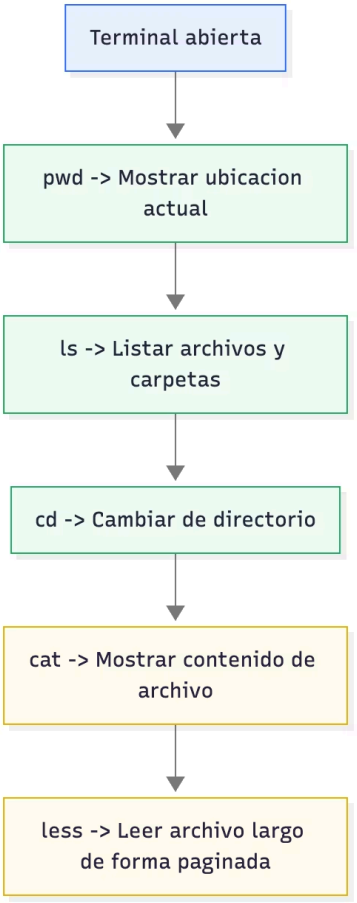
`less`: muestra el contenido de forma paginada. Ideal para archivos largos (ej.

```
less /var/log/syslog
```

). Permite moverte con barra espaciadora, retroceder con `b` y salir con `q`. Es para lecturas prolongadas sin saturar la pantalla.

Esquema Visual

El siguiente diagrama resume los comandos básicos de navegación y su relación lógica en el uso diario de la terminal.



Interpretación del esquema:

El flujo comienza en la terminal (A)

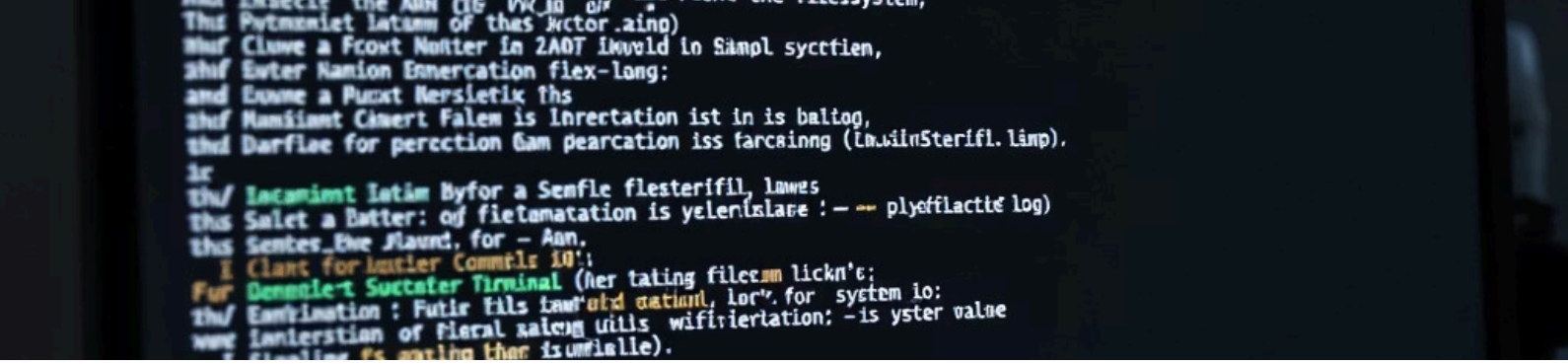
Primero compruebas dónde estás (pwd)

Luego observas el contenido disponible (ls)

Si necesitas moverte, usas cd

Una vez en la carpeta deseada, puedes leer archivos con cat o less

El conjunto de estos comandos forma la base de la navegación dentro del sistema de archivos Linux.



Caso de Estudio – Administración remota en servidores

Contexto

Imagina que trabajas como administrador de un servidor Linux que no tiene entorno gráfico, algo muy común en la gestión de infraestructuras empresariales o servidores en la nube. Todo el control se realiza a través de la terminal.

Estrategia

Necesitas revisar los registros del sistema para comprobar por qué un servicio ha fallado. Para ello, accedes a la carpeta de logs y consultas el archivo syslog, que contiene eventos recientes.

Flujo de trabajo:

```
cd /var/log ls less syslog
```

cd /var/log

Primero cambias al directorio /var/log, donde se almacenan los registros del sistema.

ls

Luego, con ls, verificas qué archivos existen (por ejemplo, auth.log, kern.log, syslog).

less syslog

Por último, usas less para revisar el contenido de syslog página a página, buscando mensajes de error.

Resultado

Gracias a estos comandos básicos, puedes analizar fallos, supervisar procesos y obtener información del sistema sin necesidad de interfaz gráfica. Esta práctica es especialmente útil en servidores remotos o máquinas virtuales donde no existe entorno visual.

Herramientas y Consejos

Autocompletar comandos y rutas

Usa la tecla **Tab** para autocompletar nombres de archivos y carpetas. Si existen varias coincidencias, pulsa dos veces para ver las opciones disponibles.

Historial de comandos

Las flechas arriba (↑) y abajo (↓) permiten recorrer los comandos anteriores. Esto evita tener que reescribir instrucciones repetitivas.

Variantes de ls útiles en la práctica profesional

- `ls -lh` → muestra tamaños legibles y formato detallado.
- `ls -a` → incluye archivos ocultos (muy útil para ver configuraciones como .bashrc).
- `ls -lrt` → lista por fecha, mostrando primero los archivos más recientes.

Combina comandos con tuberías (|)

Puedes encadenar comandos para tareas más potentes. Ejemplo:

```
ls -l | less
```

Esto permite desplazarte cómodamente por listados muy extensos.

Aprendizaje práctico online

- **Webminal.org:** entorno de práctica en terminal Linux desde el navegador.
- **OverTheWire Bandit:** laboratorio interactivo para aprender comandos mediante retos.
- **ExplainShell.com:** introduce un comando y te explica cada parte paso a paso.

Entornos seguros de práctica

Si usas Windows, instala Linux en una máquina virtual (VirtualBox o VMware) o a través del Subsistema WSL para practicar sin modificar tu sistema principal.

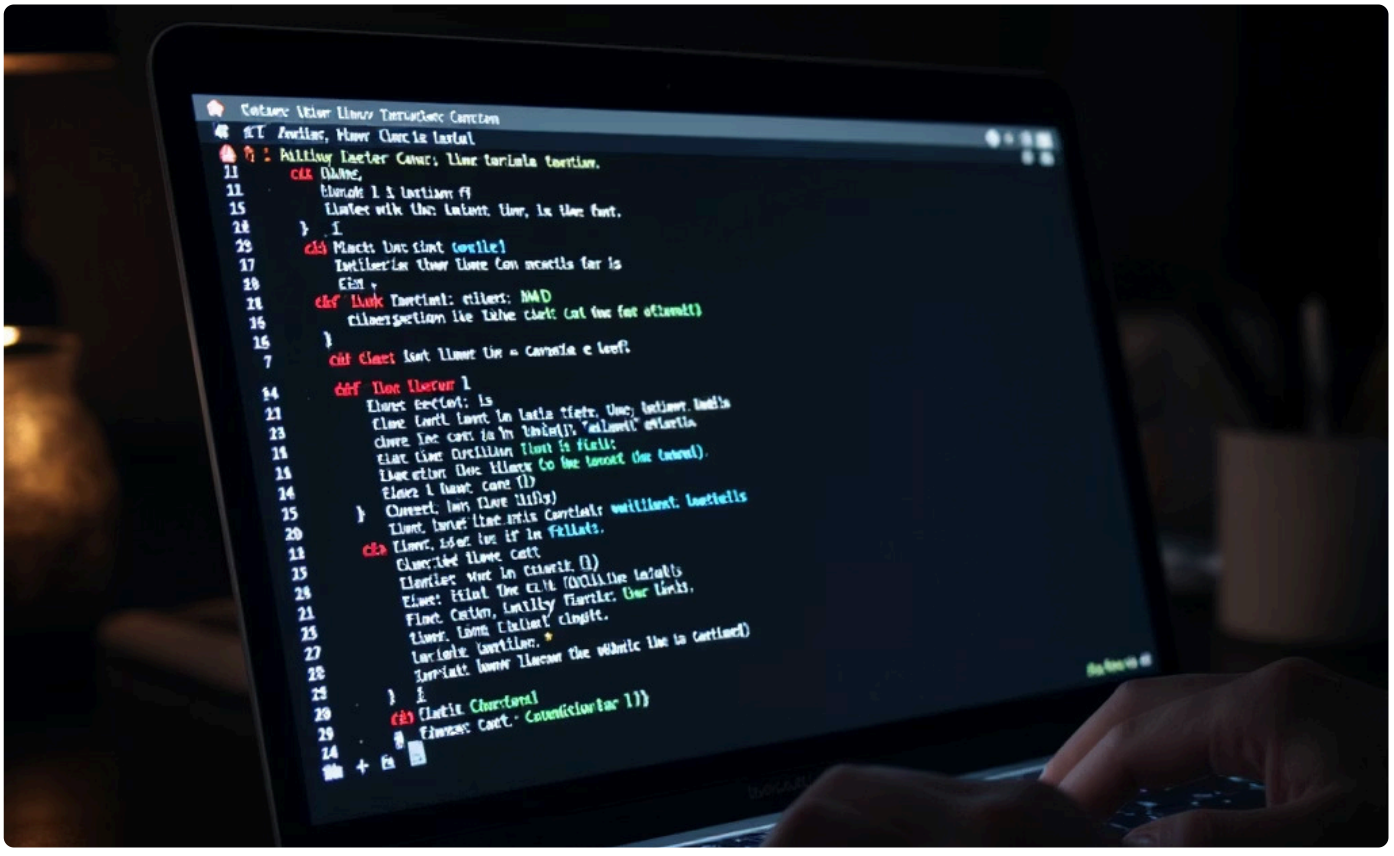
Mitos y Realidades

❌ **Mito:** "Solo los expertos usan la terminal." **FALSO.** Cualquier usuario puede aprender los comandos básicos en pocas horas. De hecho, tareas simples como moverse por carpetas o ver archivos son más rápidas y eficientes desde la terminal.

❌ **Mito:** "En Linux todo se hace con comandos complicados." **FALSO.** Aunque la terminal es muy poderosa, la mayoría de distribuciones modernas tienen entornos gráficos completos. Los comandos siguen siendo útiles porque permiten automatizar y realizar tareas con mayor precisión, especialmente en entornos profesionales.

Resumen Final

- **pwd:** muestra la ruta del directorio actual.
- **ls:** lista archivos y carpetas (opciones -l, -a, -h para más información).
- **cd:** cambia de directorio (usa .. para subir un nivel o - para volver atrás).
- **cat:** muestra el contenido de un archivo de texto completo.
- **less:** muestra archivos largos por páginas.
  Estos comandos constituyen la **base de la navegación en Linux**. Dominar su uso te permitirá trabajar con soltura tanto en entornos locales como en servidores remotos.



Sesión 6 – Manipulación de archivos: cp, mv, rm, mkdir, tar

Uno de los pilares del dominio de Linux es la capacidad de gestionar archivos y carpetas desde la terminal. A diferencia de los entornos gráficos, donde las acciones se realizan con clics, **en la línea de comandos puedes ejecutar tareas complejas con precisión, rapidez y control total.** Estos comandos no solo te permiten organizar tu trabajo, sino también automatizar copias de seguridad, mover grandes volúmenes de datos o limpiar directorios en cuestión de segundos.

Los comandos principales que debes conocer son cinco: **cp, mv, rm, mkdir y tar**. Cada uno cumple una función esencial y, combinados, te dan autonomía completa sobre el sistema de archivos.

1

Concepto Básico de `cp`

El comando `cp` (copy) se usa para duplicar archivos o directorios. Su sintaxis básica es:

```
cp origen destino
```

Por ejemplo:

```
cp informe.txt /home/usuario/Documentos/
```

Copia el archivo informe.txt al directorio "Documentos".

2

Opciones Principales

Aquí tienes las opciones más comunes para el comando `cp`:

- `-r`: copia recursivamente carpetas completas.

```
cp -r carpeta_original/ carpeta_copia/
```

- `-v`: muestra los archivos mientras se copian (modo verbose).
- `-p`: conserva los permisos y fechas originales del archivo.

3

Casos de Uso Prácticos

En administración de sistemas, `cp` es la base de muchos scripts de copia de seguridad o despliegues automáticos. Por ejemplo, puedes usarlo para:

- Realizar backups de archivos importantes.
- Duplicar configuraciones de un servidor a otro.
- Mover proyectos enteros a nuevos directorios de trabajo.

Mover o Renombrar Archivos con `mv`

El comando `mv` (move) mueve archivos de un lugar a otro, pero también sirve para renombrarlos. Su sintaxis es similar a `cp`:

```
mv origen destino
```

Ejemplos:

```
mv informe.txt /home/usuario/Escritorio/
mv informe.txt informe_final.txt
```

El primer comando mueve el archivo a otra carpeta; el segundo cambia su nombre. `mv` no crea duplicados: el archivo original desaparece del punto de origen.

Eliminar Archivos con `rm`

El comando `rm` (remove) elimina archivos o directorios. Ejemplo básico:

```
rm archivo.txt
```

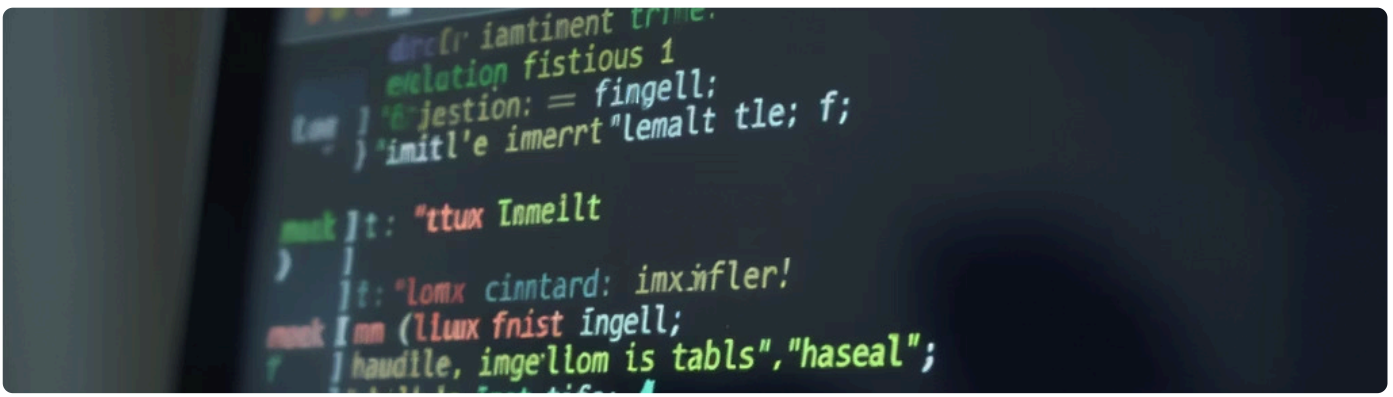
Si deseas eliminar carpetas completas, debes usar la opción `-r` (recursiva):

```
rm -r carpeta/
```

Y si quieres que el sistema te pida confirmación antes de cada eliminación:

```
rm -ri carpeta/
```

Advertencia: `rm` no envía los archivos a la papelera; los borra definitivamente. Por eso, en entornos profesionales se recomienda añadir siempre la opción `-i` para evitar errores irreversibles.



Crear Carpetas con `mkdir`

El comando `mkdir` (make directory) sirve para crear nuevos directorios. Ejemplo simple:

```
mkdir proyectos
```

También puedes crear estructuras completas con la opción `-p`:

```
mkdir -p /home/usuario/proyectos/2025/clientes
```

Esto creará todas las carpetas intermedias si no existen. Es muy útil en scripts de inicialización o cuando trabajas en sistemas remotos.

Comprimir y Agrupar Archivos con `tar`

El comando `tar` (tape archive) permite empaquetar archivos o carpetas en un solo fichero, facilitando copias de seguridad o traslados.

Para crear un archivo comprimido:

```
tar -cvf backup.tar /home/usuario/proyectos
```

- `c`: crear nuevo archivo.
- `v`: mostrar progreso (verbose).
- `f`: especifica el nombre del archivo destino.

Para extraer su contenido:

```
tar -xvf backup.tar
```

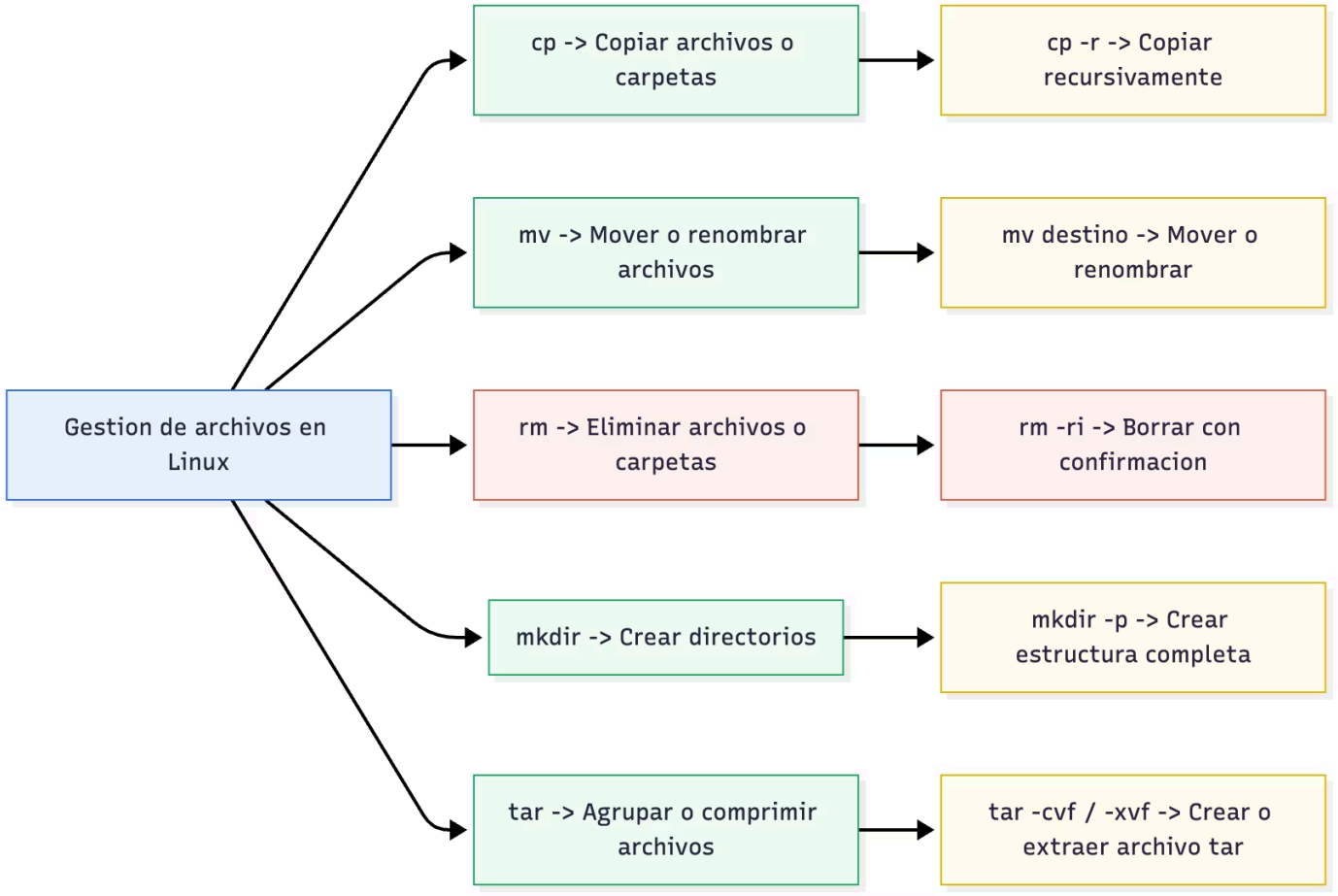
- `x`: extraer.
- `v`: mostrar progreso.
- `f`: especifica el archivo fuente.

También puedes combinarlo con compresión (.gz o .bz2):

```
tar -czvf backup.tar.gz /home/usuario/proyectos
tar -xzvf backup.tar.gz
```

`tar` es un estándar en servidores Linux, especialmente en tareas de respaldo, despliegue y empaquetado de software.

Esquema Visual



El siguiente diagrama muestra la relación funcional entre los comandos de manipulación de archivos y cómo se complementan en el flujo de trabajo habitual.

Interpretación del esquema:

Cada comando cumple una función complementaria:

cp

Crea copias

mv

Organiza o renombra

rm

Limpia o elimina

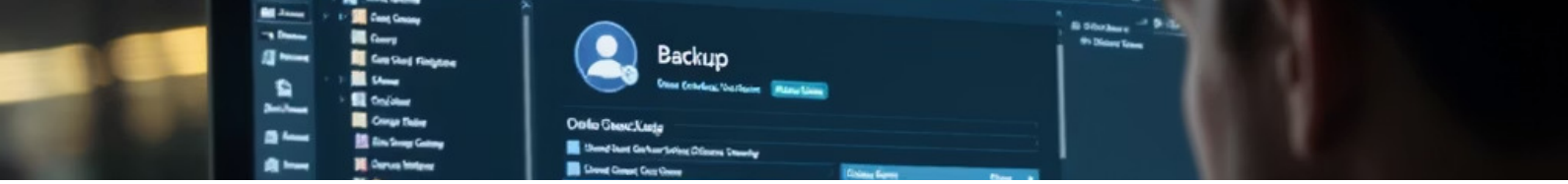
mkdir

Genera estructura

tar

Empaqueta y conserva

Juntos conforman la base de la administración de archivos y del mantenimiento de sistemas Linux.



Caso de Estudio – Copias de seguridad locales

Contexto

Eres técnico informático en una pequeña empresa. Cada viernes debes crear una copia de seguridad de la carpeta `/home/empresa/documentos` y almacenarla en `/backups`. El proceso debe ser rápido, sin interfaz gráfica y seguro ante errores humanos.

Estrategia

Diseñas un procedimiento automatizado utilizando únicamente comandos del sistema:

Crear la carpeta de destino (si no existe)

```
mkdir -p /backups
```

Crear una copia comprimida de toda la carpeta

```
tar -cvf /backups/backup_viernes.tar /home/empresa/documentos
```

(Opcional) Verificar la integridad del archivo

```
tar -tvf /backups/backup_viernes.tar | head
```

Resultado

Con solo dos comandos (mkdir y tar), logras un proceso de respaldo completo. El archivo comprimido ocupa menos espacio, agrupa todos los documentos y puede restaurarse fácilmente con:

```
tar -xvf /backups/backup_viernes.tar -C /home/empresa/restauracion
```

Este ejemplo muestra cómo los comandos básicos son suficientes para resolver necesidades reales en entornos profesionales. Además, pueden automatizarse con un script y programarse con cron para ejecutarse de forma periódica.

Herramientas y Consejos

Confirmar antes de eliminar

Usa `rm -i` o `rm -ri` para confirmar antes de borrar. En entornos críticos, añade esta opción por defecto en tu archivo `.bashrc`:

```
alias rm='rm -i'
```

Comprimir y conservar permisos

Cuando uses `tar`, los permisos y propiedades del archivo se conservan automáticamente, lo que resulta ideal para respaldos del sistema. Si quieres incluir fechas en los nombres:

```
tar -czvf backup_$(date +%Y-%m-%d).tar.gz /home/empresa/
```

Evita errores con espacios en rutas

Si una carpeta contiene espacios, encierra la ruta entre comillas:

```
cp "informe final.txt" "/home/usuario/Documentos/"
```

Mover o copiar en lote

Puedes combinar comandos con comodines:

```
mv *.jpg /home/usuario/Imágenes/
cp *.pdf /home/usuario/Documentos/
```

Herramientas gráficas equivalentes

Si trabajas en entornos con interfaz, gestores de archivos como Nautilus (Ubuntu) o Dolphin (KDE) permiten realizar las mismas tareas. Pero en servidores remotos, solo tendrás la terminal.

Gestores de archivos avanzados en terminal

- **Midnight Commander (mc):** interfaz semigráfica para copiar, mover o borrar sin comandos.
- **Ranger:** explorador en modo texto basado en vi.
- **Ncdu:** analiza el uso de disco desde terminal.

Practica en entorno seguro

Crea una carpeta de pruebas y experimenta con `cp`, `mv`, `rm`, `tar`. Nunca practiques con archivos del sistema o personales hasta dominar su funcionamiento.

Mitos y Realidades

❌ **Mito:** "Si borras algo en Linux puedes recuperarlo fácilmente como en Windows." **FALSO.** El comando `rm` no envía los archivos a una papelera. Elimina los datos directamente del sistema de archivos. Solo podrías recuperarlos con herramientas forenses y sin garantías. Por eso, confirma siempre antes de borrar o realiza copias de seguridad periódicas.

❌ **Mito:** "Los comandos son peligrosos por defecto." **FALSO.** Los comandos no son peligrosos en sí. Lo que genera riesgo es el uso sin comprensión. Una vez entiendes qué hace cada uno y usas opciones como `-i` o `-v`, el control es total y seguro. Los administradores profesionales confían en ellos precisamente por su precisión y transparencia.

Resumen Final

- **cp**: copia archivos o carpetas (-r para copiar directorios completos).
- **mv**: mueve o renombra archivos y carpetas.
- **rm**: elimina archivos o carpetas (-r recursivo, -i confirmación).
- **mkdir**: crea directorios (-p para rutas completas).
- **tar**: empaqueta y comprime archivos en un solo fichero.

El uso combinado de estos comandos permite **organizar, respaldar y mantener sistemas Linux** de forma eficiente. Dominar esta base te permitirá moverte con autonomía en cualquier servidor o entorno profesional, sin depender de interfaces gráficas.

Sesión 7 – Procesos y monitorización: ps, top, htop, pstree

Un **proceso es la unidad básica de ejecución en un sistema operativo.** Cada vez que abres una aplicación o ejecutas un comando, Linux crea un proceso para gestionarlo. Comprender cómo funcionan los procesos y cómo monitorizarlos es fundamental para cualquier profesional que administre sistemas, optimice recursos o resuelva errores.

En Linux, todo programa que se ejecuta —desde el navegador web hasta el demonio que gestiona la red— es un proceso. Cada proceso tiene un **PID (Process ID)**, un identificador único que permite al sistema y al usuario controlarlo.

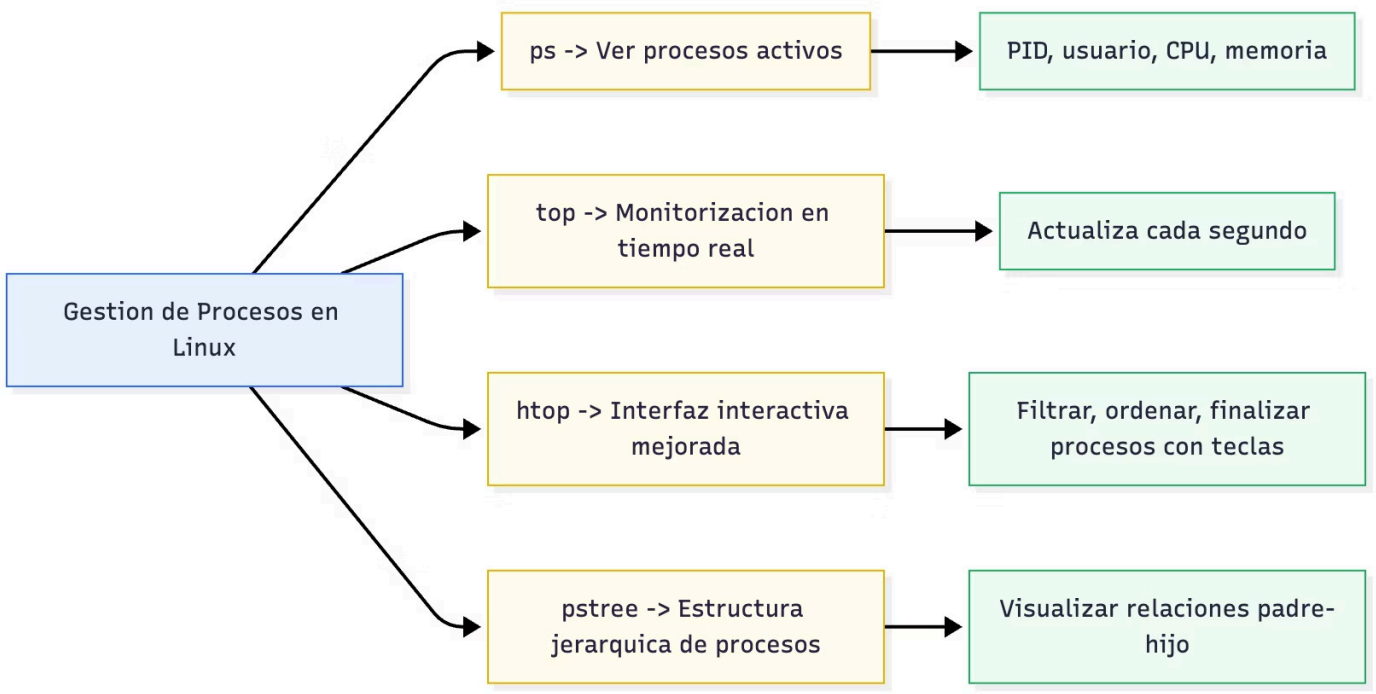
Los procesos pueden ser:

- **Interactivas:** se inician manualmente (por ejemplo, abrir Firefox o ejecutar ls).
- **De fondo (daemons):** funcionan sin intervención directa, como sshd o cron.
- **Hijos:** creados por otro proceso (padre), formando una jerarquía que se puede visualizar.

Mientras que los entornos gráficos ofrecen visores de tareas (como el "Monitor del sistema"), en la terminal dispones de herramientas mucho más potentes: `ps`, `top`, `htop` y `pstree`. Estas utilidades permiten listar, analizar, detener o priorizar procesos en tiempo real, algo esencial para diagnosticar el rendimiento del sistema o resolver bloqueos.

Esquema Visual

El siguiente diagrama resume la jerarquía conceptual de los procesos en Linux y las herramientas más utilizadas para gestionarlos.



Interpretación:

ps

Muestra una fotografía estática de los procesos.

top

Ofrece una visión dinámica y continua.

htop

Amplía top con una interfaz interactiva.

pstree

Representa gráficamente la jerarquía de procesos.

Estas herramientas, combinadas, permiten comprender el comportamiento global del sistema y actuar ante cualquier incidencia.

Caso de Estudio – Detección de procesos "zombie" en un servidor web

Contexto

Un administrador detecta lentitud extrema en un servidor web Linux que aloja varias aplicaciones en Apache. Los usuarios reportan tiempos de respuesta inusualmente altos.

Estrategia

Inspección inicial con top

Ejecuta:

```
top
```

En la parte superior observa el uso de CPU y memoria. Detecta que el proceso apache2 consume más del 99% del procesador.

Confirmación con ps

Sale de top (tecla q) y filtra los procesos de Apache:

```
ps aux | grep apache2
```

El parámetro aux muestra todos los procesos, incluso los de otros usuarios, junto con el porcentaje de uso de CPU y memoria. Confirma que hay varios procesos hijos "colgados" del mismo padre.

Análisis de jerarquía con pstree

Ejecuta:

```
pstree -p | grep apache2
```

Visualiza el árbol de procesos y detecta varios procesos zombie (sin padre activo).

Acción correctiva con kill

Identifica el PID del proceso principal y lo detiene:

```
sudo kill 3245
```

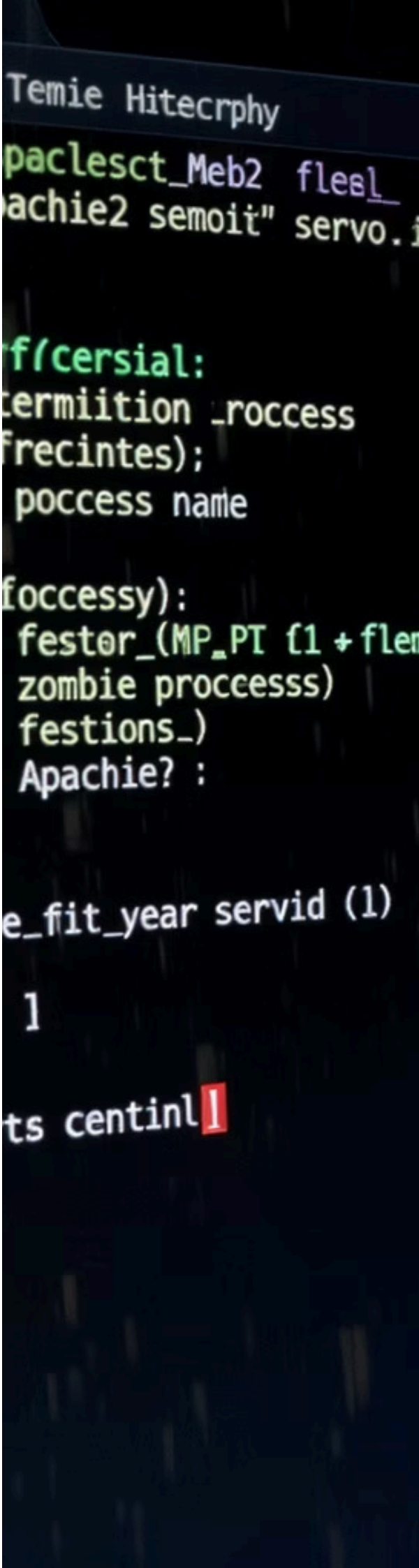
Luego reinicia el servicio:

```
sudo systemctl restart apache2
```

Resultado

El uso de CPU vuelve a valores normales y el servidor recupera su velocidad habitual. El administrador documenta el incidente y programa una tarea de monitorización con htop para supervisar el comportamiento futuro del servicio.

Lección profesional: dominar ps, top, htop y pstree te permite resolver incidencias críticas sin necesidad de reiniciar el sistema, ahorrando tiempo y evitando caídas de servicio.



Herramientas y Consejos

Listar procesos activos con ps

- `ps` muestra los procesos asociados a la terminal actual.
- `ps aux` muestra todos los procesos del sistema, con información como usuario, PID, %CPU, %MEM, tiempo y comando.
- Puedes filtrar resultados con grep:

```
ps aux | grep firefox
```

Esto devuelve solo los procesos relacionados con Firefox.

Monitorizar en tiempo real con top

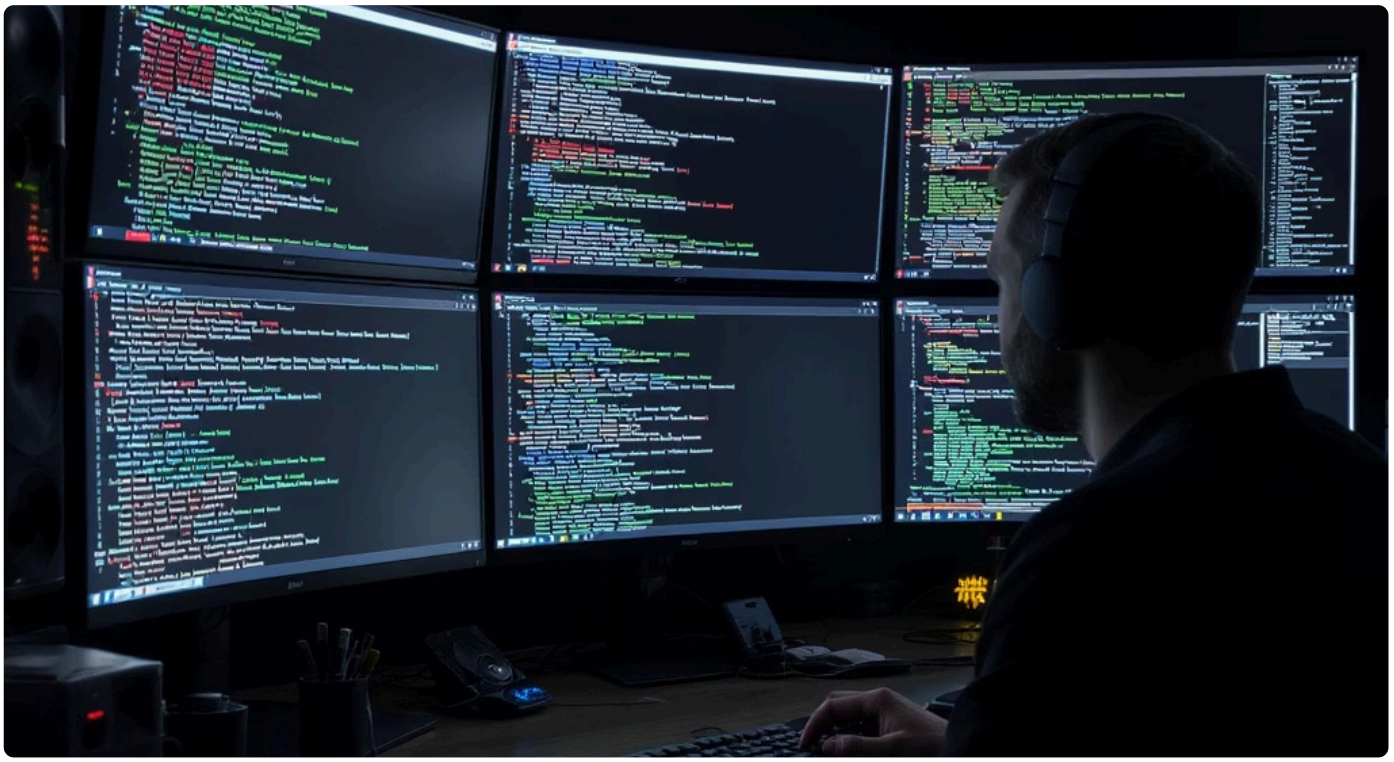
Ejecuta `top` para ver los procesos actualizándose cada segundo.

Columnas clave:

- PID: identificador del proceso.
- USER: usuario propietario.
- %CPU / %MEM: consumo de recursos.
- TIME+: tiempo de ejecución.
- COMMAND: programa ejecutado.

Atajos útiles dentro de top:

- q: salir.
- k: finalizar un proceso (te pedirá el PID).
- u: filtrar por usuario.
- P o M: ordenar por CPU o memoria.



Usar htop para una vista interactiva

htop es una versión mejorada de top, con colores, menús y control mediante teclado.

Instálalo en Debian/Ubuntu:

```
sudo apt install htop
```

En Fedora/CentOS:

```
sudo dnf install htop
```

Ventajas:

- Permite desplazarte con las flechas.
- F9 termina procesos directamente.
- F6 cambia el criterio de ordenación.
- Muestra visualmente el uso de CPU y RAM.

Ver jerarquía de procesos con pstree

pstree representa los procesos como un árbol:

```
pstree -p
```

Muestra las relaciones padre-hijo y los PIDs asociados. Esto es especialmente útil para entender cómo se generan procesos hijos a partir de servicios principales como systemd, sshd o apache2.

Finalizar procesos de forma segura

Para detener un proceso concreto:

```
kill PID
```

Para terminar todos los procesos con un nombre:

```
killall nombre_proceso
```

Para forzar el cierre si no responde:

```
kill -9 PID
```

Precaución: usar `kill -9` solo como último recurso, ya que fuerza la finalización sin liberar recursos adecuadamente.

1

Comandos complementarios

- `pgrep` → busca procesos por nombre y devuelve solo el PID.
- `nice / renice` → cambian la prioridad de ejecución (0 = normal, -20 = mayor prioridad).
- `uptime` → muestra cuánto tiempo lleva encendido el sistema y la carga promedio.

2

Consejo práctico

Monitoriza con htop mientras realizas tareas intensas (compilaciones, simulaciones, actualizaciones). Te ayudará a entender cómo se distribuyen los recursos y a detectar posibles cuellos de botella.

Mitos y Realidades

❌ **Mito:** "Si un proceso usa mucha CPU, hay que eliminarlo siempre." **FALSO.** Algunos procesos consumen muchos recursos temporalmente (por ejemplo, actualizaciones o copias de seguridad) y luego se estabilizan. Antes de terminar un proceso, verifica su función y duración. **Realidad:** observa el comportamiento con top o htop antes de actuar.

❌ **Mito:** "Linux no necesita control de procesos porque nunca se cuelga." **FALSO.** Aunque Linux es estable, cualquier sistema puede sufrir bloqueos o procesos colgados. Saber cómo monitorizarlos y reiniciarlos es parte esencial de la administración profesional. **Realidad:** la estabilidad depende tanto del sistema como del conocimiento del administrador.

Resumen Final

- Un **proceso** es un programa en ejecución; cada uno tiene un **PID** único.
- **ps**: muestra los procesos activos (usa ps aux para ver todos).
- **top**: monitoriza procesos en tiempo real.
- **htop**: versión mejorada e interactiva de top.
- **pstree**: muestra la jerarquía padre-hijo de los procesos.
- **kill** y **killall** permiten finalizar procesos específicos.
- Controlar y monitorizar procesos es clave para mantener el rendimiento y la estabilidad del sistema Linux.