Introducción a la Terminal y Línea de Comandos

Introducción

La terminal (también conocida como consola, línea de comandos o shell) es una herramienta que permite interactuar directamente con el sistema operativo mediante texto. En lugar de usar una interfaz gráfica (como ventanas y botones), en la terminal escribes comandos para realizar tareas como:

- Navegar por carpetas
- Ejecutar programas
- Instalar software
- Ver y modificar archivos
- Administrar procesos del sistema

¿Dónde se usa la terminal?

Linux y macOS: Usan terminales como Bash (Born Again Shell), Zsh, etc.

Windows: Tiene el símbolo del sistema (CMD) y PowerShell. También se puede usar Windows Terminal, que permite múltiples shells.

Tipos de Shell:

- Bourne Shell
- Bash Shell - comun en linux
- Z Shell - Mac 2019 en adelante
- C Shell
- Korn Shell
- Fish Shell
- Powershell - windows

¿Qué es Windows Subsystem for Linux (WSL)?

Windows Subsystem for Linux, conocido como WSL, es una utilidad especial diseñada por Microsoft que permite ejecutar un sistema operativo Linux directamente desde Windows sin necesidad de máquinas virtuales o configuraciones complejas:

-Funciona principalmente en Windows 10 versión 2004 en adelante y Windows 11.

-Facilita la ejecución de comandos y operaciones propias de sistemas Linux mediante la terminal Bash.

- Incluye, por defecto, la distribución de Linux llamada Ubuntu, basada en Debian.

Esta comodidad ofrece la flexibilidad de experimentar y trabajar con comandos Linux directamente desde el entorno Windows, lo cual es ideal para desarrolladores y estudiantes.

¿Cómo instalar WSL en Windows paso a paso?

Instalar WSL en Windows involucra unos cuantos pasos sencillos, que requieren atención a ciertos detalles importantes: Verifica tu compatibilidad y requisitos previos

Antes de comenzar, asegúrate de contar con lo siguiente:

- Windows 10 (versión 2004 o posterior) o Windows 11 actualizado.
- Acceso a internet estable debido a la descarga de archivos que pueden ser de gran tamaño.
- Permisos de administrador para ejecutar comandos desde PowerShell.

Instalación mediante comandos

Para iniciar la instalación:

- Abre la PowerShell de Windows en modo administrador:

  Busca PowerShell desde el menú inicio.

- Haz clic derecho y selecciona Ejecutar como administrador.

  Una vez abierto PowerShell como administrador, ejecuta el comando siguiente:

```
wsl --install
```

Este comando instalará automáticamente el subsistema de Linux en tu equipo, además de descargar e instalar Ubuntu como distribución predeterminada. Configuración inicial de WSL y Ubuntu

Tras la instalación:

- Busca en el menú "WSL", identifica el icono con forma de pingüino y ábrelo.

Se iniciará una terminal especial donde se provisionará tu instancia personalizada de Ubuntu.

Aquí deberás:

- Crear un usuario nuevo (ejemplo: Code Bars).
- Establecer y memorizar una contraseña personal (es requerida en futuras operaciones).

Luego podrás verificar que Bash se instaló correctamente ejecutando el comando:

```
echo $shell
```

Si obtienes como respuesta /bin/bash, la terminal se encuentra lista y funcionando correctamente.

Opciones alternativas de instalación

Si experimentas complicaciones técnicas específicas como limitaciones de hardware o permisos que impiden la instalación con WSL, considera estas opciones:

- Instalar Ubuntu directamente en tu computadora.
- Si eres usuario de Mac, el sistema operativo ya incluye una terminal Bash por defecto, aunque con ligeras diferencias en algunos comandos por ser un sistema Unix en lugar de Linux.

Comandos

Un programa que se puede ejecutar desde la terminal. Este puede recibir parametros y opciones

Estructura básica de un comando en Bash

`comando [opciones] [argumentos]`

- comando: Es la acción que quieres ejecutar (por ejemplo, ls, cd, mkdir, grep, etc.).
- opciones o flags: Modifican el comportamiento del comando. Suelen comenzar con - o –.
- argumentos: Son los datos sobre los que el comando actúa (por ejemplo, nombres de archivos o directorios).

Ejemplo:

`ls -l /home/christian`

- `ls`: comando para listar archivos.
- `-l`: opción que muestra los archivos en formato largo (con permisos, propietario, tamaño, etc.).
- `/home/christian`: argumento que indica el directorio a listar.

¿Cómo identificar tu usuario actual en la terminal?

El comando whoami es sencillo pero esencial. Muestra con claridad el nombre de usuario con el que has iniciado sesión en tu terminal:

`whoami`

¿Dónde está tu directorio actual y cómo revisar su contenido?

Podemos conocer exactamente en qué directorio estamos utilizando el comando pwd (print working directory):

`pwd`

Esto es útil para ubicarnos rápidamente dentro del sistema de archivos. Para listar el contenido, se usa ls. En su forma simple solo muestra archivos visibles, pero al agregar opciones puedes revelar mucho más:

- Mostrar todo, incluyendo ocultos: ls -a.
- Visualizar detalles adicionales como permisos y tamaño: ls -l.
- Combinar ambas opciones para más información útil y fácilmente comprensible: `ls -la` o incluso en formato legible para humanos: `ls -lah`.

¿Cómo visualizar información específica o realizar tareas en la terminal fácilmente?

La terminal ofrece comandos útiles para diversas situaciones prácticas, tales como:

- Limpiar la pantalla completa, usando `clear` o la combinación rápida `Ctrl + L` (en Mac, Command + L también lo logra).
- Imprimir mensajes en consola con `echo`, útil en automatizaciones y scripts:

```
echo "Hola Mundo"
```

- Visualizar información del sistema operativo y fecha actual con:

```
uname -a
date
```

- Acceder al manual completo de cualquier comando usando man. Por ejemplo, para aprender más sobre el uso del comando echo:

`man echo`

Este manual proporciona descripciones detalladas, opciones adicionales, métodos de uso y autores de cada herramienta que desees explorar.

Interpretar correctamente los símbolos y colores facilita la comprensión del listado de contenidos. En general, los directorios aparecen en azul y comienzan con la letra "d", mientras que los archivos suelen ser blancos. Los caracteres adicionales, letras y números en los listados indican permisos específicos que aprenderás a interpretar más adelante.

¿Qué es navegar con rutas absolutas?

Una ruta absoluta comienza siempre desde la raíz del sistema operativo, especificada por un slash (/). Por ejemplo, usar el comando cd / nos posiciona directamente en el directorio raíz del sistema Linux, que aloja importantes carpetas del sistema tales como:

- bin
- dev
- lib64
- root
- home

Este método permite acceder directamente a cualquier directorio proporcionando su ruta completa. ¿Cómo funcionan las rutas relativas y símbolos especiales?

Mientras que las absolutas requieren especificar toda la ruta desde la raíz, las relativas funcionan desde el lugar actual en que nos encontramos. Uso frecuente de símbolos clave:

- Punto (.) para indicar el directorio actual.
- Doble punto (..) para subir un nivel de directorio.
- Virgulilla (~) para referencia rápida al directorio home del usuario actual.

Esto permite desplazarnos rápidamente usando comandos como `cd ..` para retroceder y `cd ~` para ir al home del usuario rápidamente, independientemente del directorio actual.

¿Qué son y cómo se emplean los comandos `pushd` y `popd`?

Ambos comandos facilitan almacenar temporalmente una ubicación y regresar posteriormente a ella:

- `pushd`: guarda la ubicación actual en una pila.
- `opd`: recupera la última ubicación almacenada y nos desplaza automáticamente a ella.

Este mecanismo es muy útil para navegar cómodamente cuando trabajamos en múltiples directorios simultáneamente.

¿Cómo crear archivos y directorios con comandos básicos?

Los archivos se crean fácilmente con el comando touch. Por ejemplo, generamos un archivo de texto usando:

```
touch archivo.txt
```

Para crear directorios, se utiliza `mkdir`.

```
mkdir documentos
```

Al añadir la flag `-p`, puedes crear varias subcarpetas simultáneamente:

```
mkdir -p documentos/escuela/matematicas
```

¿De qué forma mover o renombrar archivos y directorios fácilmente?

Mover elementos se hace con el comando mv. Por ejemplo, para mover la carpeta matemáticas fuera de escuela:

```
mv escuela/matematicas .
```

Para renombrar directorios o archivos también empleas mv. Por ejemplo, cambiar la carpeta escuela a colegio:

```
mv escuela colegio
```

¿Qué necesitas saber para copiar y eliminar archivos/directorios con seguridad?

Para copiar archivos utilizas el comando `cp`:

```
cp saludo.txt adios.txt
```

Cuando copias directorios completos, debes aplicar el modo recursivo con -r:

```
cp -r documentos documentos_respaldo
```

Para eliminar archivos, puedes usar rm, pero con mucho cuidado, ya que no existe papelera en la terminal:

```
rm archivo.txt
```

Eliminar directorios requiere la opción recursiva -r y la opción -f para forzar la eliminación, siendo extremadamente cuidadoso:

```
rm -rf documentos_respaldo
```

¿Cómo explorar rápidamente el contenido de archivos en terminal?

Uno de los comandos más utilizados para visualizar contenido rápidamente es cat, que muestra todo el texto dentro del archivo instantáneamente. Sin embargo, para archivos largos como los CSV, cat puede resultar poco práctico, ya que muestra mucha información de golpe.

Para una visualización más controlada y con la posibilidad de navegar dentro del archivo, existe el comando less, que presenta una interfaz interactiva donde:

- Puedes moverte con facilidad por el texto.
- Para salir de la visualización, solo debes presionar la tecla “q”.

¿Qué comandos son útiles para mostrar partes específicas de un archivo?

Cuando buscas enfocarte solo en un segmento particular del archivo, son especialmente útiles los comandos:

- head: muestra las primeras líneas de tu archivo.
- tail: presenta las últimas líneas del archivo.

Ambos tienen opciones para especificar cuántas líneas deseas visualizar. Por ejemplo, con la opción -n:

- head -n 20 nombre_archivo: muestra las primeras 20 líneas.
- tail -n 20 nombre_archivo: presenta las últimas 20 líneas.

¿Cómo obtener información detallada del archivo?

Para información adicional sobre archivos de texto, puedes emplear diversos comandos: ¿Contar líneas y palabras fácilmente?

- nl: numera las líneas directamente desde la terminal, facilitando la identificación rápida del texto dentro de un archivo.
- wc (Word Count): brinda múltiples posibilidades para obtener estadísticas textuales claves, incluyendo:

  Cantidad total de palabras (wc archivo -w). Número total de líneas (wc archivo -l).

¿Manipular archivos CSV con eficiencia?

El comando awk es una herramienta poderosa diseñada especialmente para manipular y explorar archivos CSV:

- Permite seleccionar columnas específicas con sintaxis sencilla. Por ejemplo, para imprimir la primera columna:

¿Manipular archivos CSV con eficiencia?

El comando `awk` es una herramienta poderosa diseñada especialmente para manipular y explorar archivos CSV:

Permite seleccionar columnas específicas con sintaxis sencilla. Por ejemplo, para imprimir la primera columna:

```
awk '{print $1}' archivo.csv
```

Es posible imprimir múltiples columnas indicando separadores (-F) y columnas específicas con $:

```
awk -F"," '{print $1, $3}' archivo.csv
```

Los archivos CSV suelen usar comas para separar sus valores (Comma Separated Values), facilitando tanto su exploración como análisis.

¿Qué es un wildcard y para qué sirve?

Un wildcard es un carácter especial utilizado como comodín para hacer coincidir múltiples archivos en base a un patrón determinado. Principalmente, es útil con comandos como:

- ls (listar archivos)
- cp (copiar archivos)
- mv (mover archivos)
- rm (eliminar archivos)

Y otros comandos comunes de Unix/Linux, como head, tail o grep.

¿Cuáles son los principales tipos de wildcards?

- Asterisco (*): representa cualquier combinación de caracteres.
- Signo de interrogación (?): coincide específicamente con un único carácter.
- Corchetes [ ]: agrupan caracteres específicos.
- Llaves { }: agrupan patrones o palabras.

¿Cómo usar el wildcard asterisco (*) en tus comandos?

El wildcard más común es el asterisco (*), que coincide con cualquier combinación de caracteres. ¿Cómo aplicarlo?

Para listar archivos con una extensión específica:

```
ls *.txt
```

Para listar todos los archivos que comiencen con una palabra específica:

```
ls file*
```

¿Cómo funciona el wildcard del signo de interrogación (?)?

Este comodín hace que coincida únicamente un solo carácter en la posición exacta del patrón. Ejemplo:

```
ls file?.txt
```

El comando anterior listará “file1.txt”, “file2.txt” pero no “filelargo.txt”. ¿Cómo usar corchetes para buscar por caracteres específicos?

Con corchetes [] agrupamos caracteres precisos para afinar aún más la búsqueda:

Listar archivos terminados en letra específica antes del punto:

```
ls *[o].*
```

Esto listará archivos como “archivo.ccv” o “filelargo.txt” que tienen una “o” antes del punto.

¿Cómo usar agrupación con llaves { } para patrones específicos?

Las llaves te permiten indicar diferentes patrones o palabras específicas de manera sencilla y potente:

Para listar archivos con extensiones específicas múltiples:

```
ls *.{md,log}
```

Esto mostrará todos los archivos terminados en “.md” y “.log”, como “data.log” y “fileb.md”.

Ejemplos prácticos para mover archivos en lotes

Usar wildcards también es útil para organizar archivos rápidamente:

```
mkdir backup
mv *.txt backup/
```

Con estos comandos, mueves todos los archivos .txt hacia una carpeta llamada backup.

Precauciones al usar comandos con wildcards

Aunque los wildcards son extremadamente útiles, requieren precaución para evitar acciones involuntarias como borrar archivos esenciales. Recuerda siempre verificar bien el comando antes de ejecutarlo, especialmente usando comandos como rm.

Además, toma en cuenta que los wildcards pueden variar ligeramente según la Shell que utilices (bash, ZSH, entre otras). Si algo no funciona exactamente como esperas, verifica la documentación específica de tu Shell.

Comandos GREP y FIND para búsquedas avanzadas en Linux

El comando grep (Global Regular Expression Print) es una herramienta poderosa para buscar patrones de texto en archivos.

Sintaxis básica: grep [opciones] patrón [archivo(s)]

- Opciones comunes:
  - -i: Ignora mayúsculas/minúsculas
  - -r: Búsqueda recursiva en directorios
  - -l: Solo muestra nombres de archivos (no el contenido)
  - -n: Muestra números de línea
  - -v: Muestra líneas que NO coinciden con el patrón
  - -c: Cuenta el número de coincidencias
  - -A n: Muestra n líneas después de la coincidencia
  - -B n: Muestra n líneas antes de la coincidencia
- Casos de uso comunes:

  - Búsqueda de texto en archivos: grep “error” archivo.log
  - Búsqueda recursiva en directorios: grep -r “función” /ruta/proyecto
  - Filtrar salida de otros comandos: ls -la | grep “.txt”
  - Contar ocurrencias: grep -c “warning” archivo.log

El comando find se utiliza para buscar archivos y directorios en una jerarquía de directorios basándose en diversos criterios.

Sintaxis básica: find [ruta] [expresión]

- Opciones comunes:
  - -name: Busca por nombre de archivo (acepta wildcards)
  - -type: Busca por tipo (f=archivo, d=directorio)
  - -size: Busca por tamaño
  - -mtime: Busca por tiempo de modificación
  - -user: Busca por propietario
  - -exec: Ejecuta un comando sobre los archivos encontrados
  - -not, -and, -or: Operadores lógicos
- Casos de uso comunes:
  - Buscar archivos por nombre: find /home -name “*.txt”
  - Buscar directorios: find /var -type d -name “log*”
  - Buscar archivos modificados en los últimos 7 días: find /home -mtime -7
  - Buscar archivos grandes: find /var -size +10M
  - Ejecutar comando en archivos encontrados: find . -name “*.tmp” -exec rm {} \;
  - Buscar archivos con permisos específicos: find /etc -perm 644

¿Cómo identificar qué es exactamente cada comando?

Son varias las herramientas que Linux ofrece para identificar el tipo y las características de los comandos que usamos diariamente. Aquí exploraremos algunas que debes conocer: ¿Qué es un alias en Linux y cómo reconocerlo?

Un alias funciona como un apodo en el sistema operativo Linux, permite ejecutar comandos con ciertas opciones de manera más fácil o vistosa. Por ejemplo, el comando ls que generalmente usamos para listar directorios, en realidad es un alias de ls –color=auto para mostrar resultados con colores. Para verificar si un comando es un alias, puedes ejecutar:

```
type ls
```

Esta instrucción genera una salida que indica si el comando es un alias y cuál es su composición específica.

¿Dónde se ubican los comandos originales?

Si deseas encontrar la ruta precisa del comando que estás usando, la instrucción `which` será fundamental:

```
which ls
```

Este comando te mostrará la ubicación exacta del archivo ejecutable original, distinto al alias que podrías estar usando.

Además, para obtener más ubicaciones relacionadas con un comando específico, puedes utilizar whereis. Este comando te dará un panorama más amplio:

```
whereis ls
```

Generalmente verás rutas como `/usr/bin/ls`, indicando dónde residen la mayoría de los comandos importantes y esenciales del sistema operativo.

¿Cómo obtener información rápida sobre la función de un comando?

Para obtener rápidamente información sobre qué función cumple cualquier comando, utiliza whatis. Este comando te devuelve breve y claramente la tarea principal del comando consultado:

```
whatis grep
```

¿Qué es una redirección de terminal y para qué sirve?

Una redirección permite transferir la salida estándar (standard output) de cualquier comando hacia un archivo de texto o como entrada estándar (standard input) de otro comando. Esto es fundamental porque:

- Puedes almacenar resultados que normalmente aparecen en pantalla directamente en archivos.
- Facilita el envío de datos entre comandos en una cadena o flujo.

¿Cómo almacenar resultados de comandos en archivos?

La forma más sencilla consiste en utilizar el operador de redirección mayor que (>), como en el siguiente ejemplo:

```
echo "hola mundo" > archivo_hola.txt
cat archivo_hola.txt # Resultado: hola mundo
```

Si deseas agregar contenido adicional en lugar de sobreescribir, utilizas un doble mayor que (»):

```
echo "hola personas" >> archivo_hola.txt
cat archivo_hola.txt # Resultado: hola mundo, hola personas
```

¿Cómo funciona la redirección de entradas y salidas entre comandos?

Es posible usar la salida de un comando como entrada directa (standard input) de otro utilizando el denominado pipe operator (|). Por ejemplo, los comandos LOLCAT o Cowsay usados habitualmente en ejercicios educativos, pueden recibir entradas de esta manera:

```
echo "saludo colorido" | lolcat
cat archivo_hola.txt | lolcat
cowsay "hola mundo" | lolcat
```

¿Cómo capturar errores utilizando la terminal en Linux?

Los errores que se generan al ejecutar comandos pueden ser almacenados específicamente. En Linux, el flujo de errores estándar (standard error) es referenciado con el número 2, y puede capturarse separadamente así:

```
ls archivo_inexistente 2> errores.log
cat errores.log # Contiene el mensaje de error correspondiente
```

Si quieres concatenar varios errores dentro de un mismo archivo, la redirección se realiza con:

```
# Utilizando doble mayor que,
ls archivo_inexistente 2>> errores.log
```

¿Cómo registrar salidas y errores simultáneamente?

Otra práctica frecuente es guardar tanto la información correcta generada como los errores en un único archivo.

```
sudo apt install vim &> instalación.log
```

Aquí, &> indica que se almacenarán ambas informaciones (salida y error estándar) en “instalación.log”.