



# Módulo 4: Competencias de Línea de Comandos

---

Fundamentos de la Interfaz de Línea de Comandos (CLI) en Linux

Introducción a la interacción con el sistema operativo mediante texto

# Introducción a la CLI vs GUI

| Interfaz                              | Descripción                                           | Origen y Popularidad                                      |
|---------------------------------------|-------------------------------------------------------|-----------------------------------------------------------|
| <b>GUI</b> (Graphical User Interface) | Interfaz basada en iconos, ventanas y clics de ratón. | Introducida por Apple y popularizada por Microsoft.       |
| <b>CLI</b> (Command Line Interface)   | Interfaz basada únicamente en la entrada por teclado. | Forma preferida antes de la popularidad masiva de la GUI. |

- ✓

Control más preciso del sistema
- ✓

Capacidad de automatización (Scripting)
- ✓

Mayor velocidad de ejecución
- ✓

Flexibilidad entre distribuciones

"Aunque Linux tiene muchos entornos GUI, podrás controlar Linux mucho más eficazmente mediante la Interfaz de Línea de Comandos."

# ¿Cómo funciona la CLI?

---



El **terminal** acepta lo que el usuario escribe y lo pasa a un **shell**. El shell actúa como un intérprete que traduce las instrucciones del usuario en acciones que el sistema operativo puede ejecutar. Si el comando produce una salida, se muestra en el terminal; de lo contrario, se muestra un mensaje de error.

# Acceso a la Terminal

| Tipo de Acceso   | Descripción                                                                         | Uso Común                                                               |
|------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Servidores       | Arrancan directamente en terminal para ahorrar recursos del sistema.                | Servidores Web, Bases de Datos y entornos de alto rendimiento.          |
| Terminal de GUI  | Emulador de terminal que se ejecuta como una aplicación dentro del entorno gráfico. | Uso diario en estaciones de trabajo y desarrollo de software.           |
| Terminal Virtual | Ejecución paralela a la GUI, requiere inicio de sesión independiente.               | Recuperación de sistemas, administración avanzada o múltiples sesiones. |

 En sistemas con interfaz gráfica, se puede acceder a la terminal virtual comúnmente pulsando la combinación de teclas **Ctrl-Alt-F1**.

# El Prompt y el Shell BASH

---

## El Prompt del Sistema

```
sysadmin@localhost:~$
```

El **prompt** indica que el sistema está listo para recibir comandos. Su estructura común incluye:

- **Usuario:** sysadmin
- **Sistema:** localhost
- **Directorio:** ~ (abreviatura de /home/usuario)

## Características de BASH



### Historial de comandos

Permite volver a ejecutar comandos previos fácilmente.



### Scripting

Capacidad de automatizar tareas mediante archivos de comandos.



### Alias

Creación de sobrenombres cortos para comandos largos.

### Variables

Almacenamiento de información vital para el funcionamiento del shell.

# Formato de los Comandos

`comando [opciones] [argumentos]`

| Componente | Función                                                      | Ejemplo                |
|------------|--------------------------------------------------------------|------------------------|
| Comando    | La acción o programa que se desea ejecutar.                  | ls , cd , mkdir        |
| Opciones   | Modifican el comportamiento básico del comando.              | -l , -h , --help       |
| Argumentos | Información adicional necesaria (archivos, rutas, usuarios). | /etc/ppp , archivo.txt |



**Nota importante:** Linux es **sensible a mayúsculas y minúsculas** (*case-sensitive*). Los comandos, opciones y rutas deben escribirse exactamente como se muestran.

# Ejemplos con el comando ls

---

## >\_ Uso Básico

Lista los archivos y directorios en el directorio de trabajo actual.

```
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music
```

## 📁 Con Argumentos

Especifica el contenido de qué directorio hay que listar.

```
sysadmin@localhost:~$ ls /etc/ppp
ip-down.d ip-up.d
```

## ☰ Con Opciones (-l)

Muestra información detallada (permisos, tamaño, fecha).

```
sysadmin@localhost:~$ ls -l
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 Desktop
```

## 👤 Legibilidad Humana

Combina opciones para mostrar tamaños legibles (KB, MB).

```
sysadmin@localhost:~$ ls -lh
-rwxr-xr-x 2 root root 11K Feb 4 /usr/bin/perl
```

---

**i** Recuerda: Las opciones de una letra usan un guion ( **-** ), mientras que las de palabra completa usan dos ( **--** ). El orden de las opciones combinadas (como **-lh**) no es importante.

# Historial de Comandos

El comando **history** muestra la lista de comandos ejecutados anteriormente, permitiendo ahorrar tiempo al no tener que escribir comandos largos repetidamente.

```
$ history
1 date
2 ls /etc/ppp
3 cal 5 2015
4 history
```

- ↑ Flecha Arriba: Ver comando anterior
- ↓ Flecha Abajo: Ver comando siguiente
- ✎ Teclas Inicio/Fin: Editar el comando

| Acción                          | Comando / Atajo |
|---------------------------------|-----------------|
| Ejecutar último comando         | !!              |
| Ejecutar comando número n       | ! <b>n</b>      |
| Ejecutar <b>ls</b> más reciente | ! <b>ls</b>     |
| Ejecutar 5º desde el final      | ! <b>-5</b>     |

*Nota: Los comandos se almacenan en una lista temporal en la memoria del shell.*

# Variables del Shell y PATH

---

## Variables del Shell

Las variables permiten almacenar datos temporalmente en la memoria para proporcionar información del sistema o cambiar su comportamiento.

- **Ver valor:** Usa `echo $NOMBRE`
- **Asignar valor:** Usa `NOMBRE=valor`

```
$ echo $HISTSIZE
1000
$ HISTSIZE=500
```

## La Variable PATH

Define la lista de directorios donde el shell buscará los comandos ejecutables.

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin
```

Si un comando no está en el **PATH**, el shell no podrá localizarlo.

 **Error: "command not found"**

# Identificación de Comandos

| Comando            | Propósito Principal                   | Qué busca                                                                              |
|--------------------|---------------------------------------|----------------------------------------------------------------------------------------|
| <code>which</code> | Localizar la ruta del ejecutable.     | Busca archivos en los directorios del <code>PATH</code> .                              |
| <code>type</code>  | Determinar la naturaleza del comando. | Identifica si es un <code>builtin</code> , <code>alias</code> o <code>archivo</code> . |

## Uso de which

```
sysadmin@localhost:~$ which date
/bin/date
sysadmin@localhost:~$ which cal
/usr/bin/cal
```

## Uso de type

```
sysadmin@localhost:~$ type echo
echo is a shell builtin
sysadmin@localhost:~$ type ll
ll is aliased to `ls -aF`
```

# Los Alias: Atajos Personalizados

---

## ¿Qué es un Alias?

Un **alias** permite asignar comandos largos o complejos a secuencias de texto más cortas para aumentar la productividad y rapidez en la terminal.

Para ver los alias definidos actualmente en tu shell, simplemente escribe:

```
$ alias
alias ll='ls -aLF'
alias la='ls -A'
alias l='ls -CF'
```

## Creación de Alias

Puedes crear tus propios alias utilizando la siguiente sintaxis:

```
alias nombre='comando'
```

Ejemplo práctico para listar archivos ordenados por tamaño en formato legible:

```
$ alias lh='ls -Shl'
$ lh /etc/ppp
```



**Persistencia:** Los alias creados así solo duran mientras el shell esté abierto. Para que sean permanentes, deben guardarse en archivos de configuración.

# Globber (Comodines)

Los caracteres de **globber** o "comodines" son símbolos interpretados por el shell para coincidir con patrones de nombres de archivos antes de ejecutar un comando.

| Comodín | Significado                                                   | Ejemplo de Uso                                              |
|---------|---------------------------------------------------------------|-------------------------------------------------------------|
| *       | Representa <b>cero o más</b> caracteres de cualquier tipo.    | echo /etc/t* (Coincide con terminfo, timezone)              |
| ?       | Representa <b>exactamente un</b> carácter único.              | echo /etc/t??????? (Exactamente 7 caracteres tras la 't')   |
| [ ]     | Representa <b>un carácter</b> dentro de un conjunto o rango.  | echo /etc/[gu]* (Archivos que empiezan por 'g' o 'u')       |
| [a-d]   | Representa un <b>rango</b> de caracteres (según tabla ASCII). | echo /etc/[a-d]* (Archivos que empiezan de la 'a' a la 'd') |
| [!]     | Se utiliza para <b>negar</b> un conjunto o rango.             | echo [!DP]* (Archivos que NO empiezan por 'D' o 'P')        |

 *Nota: Los comodines pueden combinarse para crear patrones complejos, como tres letras.*

`echo /etc/*.???` para buscar archivos con extensiones de

# El uso de Comillas

| Tipo de Comilla | Efecto en el Shell                                                         | Ejemplo                |
|-----------------|----------------------------------------------------------------------------|------------------------|
| Dobles " "      | Bloquean comodines, pero permiten variables ( \$ ) y comandos.             | echo "Path is \$PATH"  |
| Simples ' '     | Bloquean la interpretación de <b>todos</b> los caracteres especiales.      | echo 'Costs \$100'     |
| Invertidas ` `  | <b>Sustitución de comando</b> : ejecuta el contenido y devuelve la salida. | echo "Today is `date`" |

 **Barra Inversa ( \ ):**Se utiliza para "escapar" un solo carácter, tratándolo como texto literal. Es útil cuando quieres que un carácter especial (como \$ ) no sea interpretado por el shell dentro de comillas dobles.

```
$ echo The service costs \$100 and the path is $PATH
```

# Instrucciones de Control

Las instrucciones de control permiten ejecutar múltiples comandos a la vez o condicionar su ejecución según el éxito o fracaso del comando anterior.

| Operador          | Nombre               | Comportamiento                                                                         |
|-------------------|----------------------|----------------------------------------------------------------------------------------|
| <b>;</b>          | Punto y Coma         | Ejecuta comandos <b>secuencialmente</b> sin importar el resultado del anterior.        |
| <b>&amp;&amp;</b> | Ampersand Doble      | Operador "Y" lógico. Ejecuta el segundo comando <b>solo si</b> el primero tiene éxito. |
| <b>  </b>         | Línea Vertical Doble | Operador "O" lógico. Ejecuta el segundo comando <b>solo si</b> el primero falla.       |

```
# Ejemplo de éxito (&&):
$ ls /etc/xml && echo "Directorio encontrado"
catalog catalog.old xml-core.xml
Directorio encontrado

# Ejemplo de fallo (||):
$ ls /etc/junk || echo "Error: El directorio no existe"
ls: cannot access /etc/junk: No such file or directory
Error: El directorio no existe
```