💻

103.1 Trabajar en la línea de comandos

Teoría

La línea de comandos (también conocida como *shell*, *terminal* o *consola*) es la interfaz primaria para interactuar con el sistema operativo Linux de manera textual. Aunque existen interfaces gráficas, la CLI es indispensable para la automatización, administración remota y tareas avanzadas.

1. **La Shell:**
    - Es el programa que interpreta los comandos que escribes y ejecuta las acciones solicitadas. Actúa como un intérprete entre tú y el kernel.
    - Hay diferentes shells disponibles en Linux (Bash, Zsh, Ksh, Tcsh, etc.). **Bash (Bourne Again SHell)** es la shell por defecto en la mayoría de las distribuciones y es la que se espera dominar para LPIC-1.
    - Cuando abres una terminal, estás interactuando con una instancia de la shell.
2. **Estructura de un Comando:**
    - La sintaxis básica es: `comando [opciones] [argumentos]`
        - **Comando:** El nombre del programa o función que quieres ejecutar (ej: `ls`, `cd`, `cp`).
        - **Opciones (Flags o Switches):** Modifican el comportamiento del comando. Suelen empezar con uno o dos guiones.
            - Opciones cortas: Empiezan con un guion (`-`). A menudo se pueden agrupar (ej: `ls -l -a` es lo mismo que `ls -la`).
            - Opciones largas: Empiezan con dos guiones (`--`). Suelen ser más descriptivas (ej: `ls --all --long`).
            - Algunas opciones requieren un valor (ej: `grep -n 'patron' archivo` donde `-n` es la opción y `'patron'` es su argumento). Para opciones largas, a menudo es `--option=value` (ej: `sort --key=2`).
        - **Argumentos:** Son los datos sobre los que opera el comando (ej: nombres de archivos, directorios, cadenas de texto). `ls /home` (el argumento es `/home`), `cp archivo_origen archivo_destino` (hay dos argumentos).
3. **Comandos Internos vs. Externos:**
    - **Comandos Internos:** Son funcionalidades incorporadas directamente en la shell (ej: `cd`, `pwd`, `echo`, `export`, `alias`). Son ejecutados directamente por la shell, no son programas separados en el sistema de archivos. Son más rápidos porque no requieren buscar y cargar un ejecutable externo.

- **Comandos Externos:** Son programas ejecutables almacenados en el sistema de archivos (ej: `/bin/ls`, `/bin/grep`, `/usr/bin/find`). La shell los localiza (usando la variable `PATH`) y los ejecuta como procesos separados.
- Puedes usar el comando `type <comando>` para saber si un comando es interno, un alias, una función o un programa externo.

4. **Variables de Entorno y Variables de Shell:**
   - **Variables de Shell:** Almacenan información dentro de la shell actual (ej: `USER`, `HOME`, `PS1`, `PATH`). Se acceden prefijando el nombre con `$`. Se asignan sin `$`, ej: `mi_variable="hola"`.
   - **Variables de Entorno:** Son variables de shell que han sido "exportadas" para que estén disponibles para los procesos hijos que la shell inicie. Son cruciales para configurar el entorno de trabajo.
   - `printenv` o `env`: Muestran las variables de entorno.
   - `set`: Muestra *todas* las variables de shell y de entorno, funciones y aliases definidos.
   - `export <nombre_variable>`: Convierte una variable de shell en una variable de entorno.
   - `PATH`: Una variable de entorno crucial que contiene una lista de directorios donde la shell busca comandos ejecutables. Cuando escribes un comando externo, la shell busca en los directorios listados en `PATH` en orden.
5. **Historial de Comandos:**
   - La shell Bash guarda un historial de los comandos que has ejecutado.
   - Puedes navegar por el historial usando las teclas de flecha (Arriba/Abajo).
   - `history`: Muestra el historial completo.
   - `!n`: Ejecuta el comando número `n` del historial.
   - `!!`: Ejecuta el último comando.
   - `!cadena`: Ejecuta el último comando que empezó con `cadena`.
   - `!?cadena?`: Ejecuta el último comando que contenía `cadena`.
   - El historial se guarda en un archivo (típicamente `~/.bash_history`) al cerrar la shell. El número de comandos guardados está controlado por la variable `HISTSIZE`.
6. **Completado por Tabulación (Tab Completion):**
   - Presionar la tecla `Tab` una o dos veces ayuda a completar nombres de comandos, archivos, directorios, variables, etc.
   - Es una herramienta de productividad inmensa y ayuda a evitar errores tipográficos.
7. **Aliases:**
   - Atajos personalizados para comandos más largos o complejos.
   - `alias nombre='comando [opciones] [argumentos]'`: Define un alias.
   - `alias`: Muestra los aliases definidos.
   - `unalias nombre`: Elimina un alias.

- Los aliases se definen típicamente en archivos de configuración de la shell como `~/.bashrc`.

8. **Comillas y Caracteres Especiales:**
    - La shell interpreta muchos caracteres de manera especial (espacio, `*`, `?`, `>`, `<`, `|`, `&`, `$`, `!`, `"`, `'`, `\`).
    - **Comillas simples (`'`):** Protegen completamente el texto encerrado. Las variables dentro de comillas simples *no* se expanden.
    - **Comillas dobles (`"`):** Protegen la mayoría de los caracteres especiales, pero *sí* permiten la expansión de variables (`$`) y la ejecución de comandos dentro de `$()`.
    - **Barra invertida (`\`):** Escapa el siguiente carácter especial, haciendo que la shell lo trate literalmente.
9. **Obteniendo Ayuda:**
    - `man <comando>`: Muestra la página del manual (man page) del comando. Es la documentación oficial y detallada. Presiona `q` para salir del lector de man pages (`less`).
    - `apropos <palabra_clave>`: Busca páginas de manual que contienen la palabra clave en su nombre o descripción.
    - `info <comando>`: Muestra la documentación en formato "info", que a menudo es más estructurada e hiperenlazada que las man pages, pero menos universalmente disponible.
    - `<comando> --help`: Muchos comandos externos soportan la opción `--help` para mostrar un resumen rápido de su uso y opciones.
    - `help <comando_interno>`: Muestra ayuda para comandos internos de la shell (ej: `help cd`, `help export`).