Requisitos previos

☒Disponer de un acceso de usuario al sistema Linux.

☒Disponer de un terminal (consola o gráfico).

Objetivos

Al final de este capítulo, será capaz de:

☒Trabajar con la línea de comandos.

☒Manejar el historial de comandos.

☒Efectuar operaciones básicas en el sistema de archivos.

☒Buscar archivos en el árbol.

☒Trabajar en archivos de texto con el editor de texto vi.

☒Instalar redirecciones y tuberías.

☒Efectuar búsquedas en archivos de texto.

☒Utilizar los filtros y utilidades.

☒Gestionar los procesos.

☒Modificar la ruta de búsqueda de los ejecutables, las variables y el entorno.

☒Modificar la configuración del bash.

☒Crear scripts.

☒Efectuar pruebas y evaluar el valor lógico de una expresión.

☒Utilizar estructuras de control.

☒Crear funciones.

☒Usar un multiplexor de terminales.

A. El shell bash

1. Función del shell

Aunque todas las distribuciones ofrecen interfaces gráficas de usuario, un informático profesional que trabaje en un sistema Linux debe estar familiarizado con el funcionamiento del intérprete de comandos (shell) y los principales comandos en modo de caracteres. Por un lado, los sistemas de servidores se suelen instalar sin interfaz gráfica y, por otro lado, es fundamental poder gestionar scripts operativos y administrativos escritos en lenguaje shell y combinando comandos en modo de caracteres.

El intérprete de comandos permite ejecutar instrucciones que se escriben con el teclado o que se leen desde un archivo de script. Este intérprete suele ser un programa de tipo shell. El término shell (concha), de origen Unix, se utiliza en referencia al término **kernel** (nodo): el shell es una interfaz "alrededor" del kernel de Linux, que opera en modo de caracteres.

Hay varios programas de tipo shell, cada uno con sus propios aspectos específicos. El **Bourne Shell**, llamado así por su creador **Steve Bourne**, es el shell más antiguo escrito para Unix. Después, el shell se estandarizó bajo los estándares POSIX.

El shell de referencia para la mayoría de las distribuciones de Linux es bash (*Bourne Again Shell*), pero hay muchos otros, entre ellos:

- `sh`: Bourne Shell
- `ksh`: Korn Shell
- `csh`: C Shell
- `zsh`: Z Shell
- `ash`: A Shell
- `dash`: Debian Almquist Shell.

☞*El archivo `/etc/shells` proporciona la lista de los shells instalados en el sistema.*

2. Bash: el shell Linux por defecto

El shell bash es un derivado del Bourne Shell. Cumple con los estándares POSIX, pero añade muchas extensiones que son específicas para él. Es el shell de referencia para los exámenes de certificación LPIC-1.

☞*En las distribuciones recientes de Debian, el shell por defecto es dash, una variante muy similar al shell bash.*

a. Un shell potente y libre

El bash, que tiene licencia de código abierto GNU, se proporciona de forma predeterminada con todas las distribuciones de Linux. Incluso viene en versiones para macOS y Windows (a través de la funcionalidad Subsistema de Windows para Linux).

El shell funciona en modo de línea de comandos. Cuando se inicia desde un terminal, se inicializa desde varios archivos, muestra un mensaje de aviso al comienzo de una línea de símbolo del sistema y entra en modo de lectura de teclado. Cuando la línea de comandos se escribe y se valida con la tecla [Intro], el shell interpreta su contenido y lo ejecuta. Una vez completada la ejecución, el shell vuelve a mostrar el prompt y espera una nueva línea.

La secuencia de teclas [Ctrl] D completa la ejecución del shell. El comando `exit` también hace que el shell termine.

☞*Linux, al igual que Unix, distingue entre mayúsculas y minúsculas en los comandos, sus opciones y argumentos, así como en los nombres de archivos y directorios.*

b. Línea de comandos

El shell espera entradas por el teclado en una línea llamada línea de comandos. La cadena que se muestra al principio de esta línea se denomina **prompt**.

El prompt se puede configurar (mediante la variable de entorno `PS1`). Su contenido predeterminado varía en función de la distribución. Por lo general, muestra el nombre de la cuenta de usuario, el directorio actual y un carácter `$` (cuenta que no es de administrador) o `#` (cuenta de administrador).

Ejemplo

Prompt predeterminado del usuario `pba` en el sistema `srvrh` (distribución RHEL 9):

```
[pba@srvrh ~]$
```

Prompt predeterminado del usuario `root` (administrador) en el sistema `srvdeb` (distribución Debian 12):

```
root@srvdeb
```

3. Utilizar el shell

a. La introducción de datos en una línea de comandos

En la línea de comandos, puede mover el cursor con las teclas [Flecha derecha] y [Flecha izquierda], y eliminar caracteres con las teclas [Retroceso] o [Supr]. Para ejecutar, pulse en la tecla [Intro].

Se pueden utilizar los siguientes métodos abreviados de teclado:

- **[Ctrl] A**: ir al principio de la línea.
- **[Ctrl] E**: ir al final de la línea.
- **[Ctrl] L**: borrar el contenido de la pantalla y mostrar la línea de comandos en la parte superior.
- **[Ctrl] U**: borrar la línea hasta el principio.
- **[Ctrl] K**: borrar la línea hasta el final.

Ejemplos

Comando de visualización de la fecha.

```
$ date
lunes 15 mayo 2023 09:40:20 CEST
```

Comando de visualización de la ruta de acceso al directorio actual.

```
$ pwd
/home/pba
```

b. Sintaxis general de los comandos

La mayoría de los comandos incluidos en las distribuciones de Linux son originales de Unix, pero han sido reescritos como parte del proyecto de código abierto GNU. Su sintaxis puede variar de una versión a otra.

Los comandos GNU/Linux suelen tener la siguiente sintaxis:

```
Comando [opciones] [argumentos]
```

Un comando puede no tener ni opciones ni argumentos. Las opciones se identifican con mayor frecuencia por un carácter precedido por un guion: `-l`, `-p`, `-s`, etc. Si el comando admite varias opciones, puede especificarlas una tras otra, separadas por espacios: `-l -r -t o` agruparlas detrás de un solo guion: `-lrt`. El orden de las opciones no importa, las dos sintaxis anteriores generan el mismo resultado.

☞*Algunas opciones esperan un argumento, como por ejemplo un nombre de archivo. En este caso, los separaremos de los demás: `-lrt -f miarchivo` o los colocaremos en la última posición: `-lrtf miarchivo`.*

Los argumentos son cadenas separadas por un espacio o un carácter de tabulación. Si un argumento va a contener un espacio, debe estar entre comillas simples ' ' o dobles " ".

c. Ejemplo de comando: cal

El comando `cal` admite varias opciones y argumentos. Si se le invoca sin argumentos, muestra el calendario del mes en curso.

Ejemplo

```
$ cal

     Agosto 2023
lu ma mi ju vi sá do
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

El comando `cal` admite dos argumentos opcionales. Si se precisa solo uno, se trata del año, y se muestra el calendario de ese año en su totalidad. Si se le indican dos argumentos, el primero es el mes; el segundo, el año.

Ejemplo

```
$ cal 12 1975
    diciembre 1975
lu ma mi ju vi sá do
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

El comando recibe algunas opciones, que varían según la versión instalada.

La opción `-m` (*monday*) muestra los días de la semana empezando por el lunes.

La opción `-s` (*sunday*) muestra los días de la semana empezando por el domingo.

Ejemplo

```
$ cal -s 12 1975
    diciembre 1975
do lu ma mi ju vi sá
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

La opción `-m3` se utiliza para mostrar el mes anterior y el mes posterior al mes especificado o, sin argumentos, el mes actual.

Ejemplo

```
$ cal -m3 12 1975
cal -m3 12 1975
   noviembre 1975         diciembre 1975          enero 1976
lu ma mi ju vi sá do  lu ma mi ju vi sá do  lu ma mi ju vi sá do
                1  2   1  2  3  4  5  6  7            1  2  3  4
 3  4  5  6  7  8  9   8  9 10 11 12 13 14   5  6  7  8  9 10 11
10 11 12 13 14 15 16  15 16 17 18 19 20 21  12 13 14 15 16 17 18
17 18 19 20 21 22 23  22 23 24 25 26 27 28  19 20 21 22 23 24 25
24 25 26 27 28 29 30  29 30 31              26 27 28 29 30 31
```

Con una distribución similar a Debian, puede usar el comando similar `ncal`, que tiene una sintaxis ligeramente diferente (el comando `cal` en realidad ejecuta `ncal`).

Ejemplo

Para lograr el mismo resultado que en el ejemplo anterior:

```
$ ncal -b -M -3 7 1978
     Junio 1978            Julio 1978           Agosto 1978
lu ma mi ju vi sá do  lu ma mi ju vi sá do  lu ma mi ju vi sá do
          1  2  3  4                  1  2      1  2  3  4  5  6
 5  6  7  8  9 10 11   3  4  5  6  7  8  9   7  8  9 10 11 12 13
12 13 14 15 16 17 18  10 11 12 13 14 15 16  14 15 16 17 18 19 20
19 20 21 22 23 24 25  17 18 19 20 21 22 23  21 22 23 24 25 26 27
26 27 28 29 30        24 25 26 27 28 29 30  28 29 30 31
                      31
```

d. Encadenar los comandos

Se pueden especificar varios comandos en la misma línea de comandos, separados por un punto y coma. Se ejecutarán de forma sucesiva.

Ejemplo

```
# date;pwd;cal -m  lun. 15 mayo 2023 11:30:26 CEST
/home/pba
      mayo 2023

do lu ma mi ju vi sá
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

e. Visualizar texto

El comando `echo` muestra los argumentos que se le pasan, separados por un espacio y seguidos de un salto de línea.

Ejemplo

```
$ echo Hola amigos
Hola amigos
```

Los argumentos pueden contener caracteres especiales del lenguaje C, siempre y cuando se especifique la opción `-e`. Los más utilizados son:

Secuencia	Acción
`\n`	Salto de línea
`\t`	Tabulación horizontal
`\c`	Sin salto de línea después de la visualización de argumentos
`\b`	Retorno de un carácter atrás

Secuencia	Acción
\\	Visualiza el antislash (barra oblicua inversa)
\nnn	Visualiza el carácter con el código nnn

Ejemplo

```
$ echo -e "Hola.\tMe llamo Javier\b\b\bNadie\n"
Hola. Me llamo Nadie
```

☞*Este comando se utiliza principalmente en scripts, para mostrar comentarios, instrucciones de usuario, mensajes de error, etc.*

f. Comandos internos y externos

Existen dos tipos de comandos:

- Los **comandos externos** son programas ejecutables que se almacenan en archivos. Cuando se ejecuta el comando, el shell determina la ubicación del archivo correspondiente y, si lo encuentra, comienza su ejecución pasándole eventualmente opciones y argumentos.
- Los **comandos internos** son internos al shell y son ejecutados directamente por el shell. El código ejecutable de estos comandos es parte integrante del código de shell (primitivo de shell). Los comandos cd o pwd son dos ejemplos de ello.

Para distinguir un comando interno de otro externo, se puede utilizar el comando interno `type`.

Ejemplo

```
$ type date
date is /bin/date
$ type pwd
pwd es una orden interna del shell
```

El comando `date` es un comando externo, pero su ruta la conserva el shell en memoria (hashage).

```
$ type pwd
pwd es una primitiva del shell
```

g. Secuencias de control

`[Ctrl] C`: Hace que se interrumpa el comando actual.

`[Ctrl] D`: Al principio de la línea, termina de escribir si el comando está leyendo desde el teclado, o finaliza el shell actual si está esperando el teclado.

Ejemplo

Sin argumentos, el comando `sort` ordena las filas introducidas en el teclado. Para detener la entrada, escriba `[Ctrl] D` al principio de la línea:

```
$ sort
bbbbbbbbbb
aaaaaaa
zzzzzzz
```

```
eeeeeeee
[Ctrl]D
aaaaaaa
bbbbbbbbbbb
eeeeeeee
zzzzzzz
```

4. Historial de comandos

El shell almacena un historial de líneas de comandos, llamado `.bash_history`, en un archivo en el directorio de conexión de cada cuenta de usuario. Es posible navegar con las teclas [Flecha arriba] y [Flecha abajo], sabiendo que la flecha hacia arriba permite retroceder en el historial. Puede cambiar o no la línea de comandos mostrada y solicitar su ejecución con la tecla [Intro] (independientemente de la posición del cursor en la línea de comandos).

El comando `history` muestra las líneas de comandos introducidas más recientemente.

Ejemplo

```
$ history
...
 1000  date
 1001  pwd
 1002  uname -a
 1003  ls
 1004  fc -l -5
 1005  history
```

El comando `fc -l` muestra las últimas quince líneas de comandos, numerándolas. El número de líneas de comando que se mostrarán se puede especificar pasándolo como una opción.

Ejemplo

Ver las últimas 20 líneas del comando:

```
$ fc -l -20
109    cal -m -3 12 1975
110    date;pwd;cal
111    date;pwd;cal -m
112    echo "toto\n"
113    echo -e "toto\n"
114    type date
115    type pwd
116    type ll
117    fc -l -10
118    uname -a
```

La opción `-s` seguida del número del comando provoca que se vuelva a ejecutar.

Ejemplo

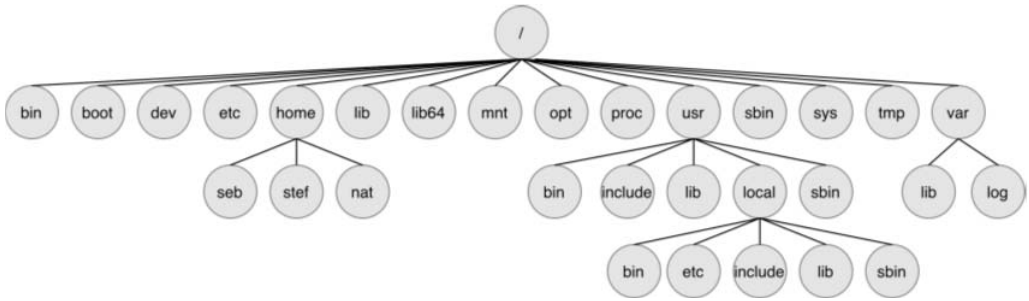
```
$ fc -s 118
uname -a
Linux srvdeb 5.10.0-22-amd64 #1 SMP Debian 5.10.178-3 (2023-04-22) x86_64 GNU/Linux
```

B. La gestión de los archivos

Linux es, al igual que Unix, un sistema operativo orientado a archivos. Todo (o casi) puede estar representado por un archivo, datos (archivos de datos de cualquier tipo, como una imagen o un programa), periféricos (terminales, ratón, teclado, tarjeta de sonido, etc.), canales de comunicación (sockets, pipes con nombre), etc.

1. El sistema de archivos

Un sistema de archivos (*File System*) define la estructura que organiza la gestión de directorios y archivos en todo o parte de un medio de almacenamiento. Todos los soportes de almacenamiento que se activan (montados) conforman el sistema global de archivos, que se presenta en forma de un único árbol, estructurado por directorios:



Ejemplo de árbol de directorio Linux

El sistema de archivos de Linux es jerárquico. Describe un árbol de directorios y subdirectorios, a partir de un elemento básico llamado **raíz** (*root directory*).

2. Los diferentes tipos de archivos

Distinguimos tres tipos de archivos: ordinario, directorio, especial.

a. Los archivos ordinarios o regulares

Los archivos ordinarios se llaman también archivos regulares, (*ordinary regular files*). Son archivos que contienen datos. El sistema los considera una secuencia de bytes. Pueden contener texto, imágenes, sonido, un programa ejecutable, etc.

El comando `file` se utiliza para obtener información sobre la naturaleza del contenido de un archivo.

Ejemplo

```
$  file /usr/bin/bash /etc /etc/passwd
/usr/bin/bash: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=
5d82a44f2a4466ff21f763af86b004d1fcb3a8f1, for GNU/Linux 3.2.0, stripped
/etc:           directory
/etc/passwd:    ASCII text
```