Linux: Conceptos basicos

Gonzalo E. Moncada

EEST N°5 Amancio williams

7°2

Paola Flament, Martin Estanga

4 abril

**1.- Describa el sistema de archivos de Linux, estructura de directorios en forma completa.**

El sistema de archivos de linux es la forma en que el sistema operativo estructura los datos en la unidad de almacenamiento.

En Linux los datos se ordenan en archivos y directorios, la diferencia con otros sistemas operativos es que, generalmente, los programas no son almacenados cada uno en su propio directorio, sino que sus diferentes componentes están dispersos por todo el sistema de archivos. Así contamos con un directorio específico para todos los ejecutables, otro para la documentación, otro para las librerías, etc. Por ejemplo, para ejecutar un programa en GNU/Linux, generalmente se busca su archivo ejecutable en el directorio "/usr/bin". Este directorio contiene todos los ejecutables que están disponibles para todos los usuarios del sistema.

Sin embargo, no todos los componentes de un programa están almacenados en el directorio "/usr/bin". Por ejemplo, la documentación de un programa puede estar almacenada en el directorio "/usr/share/doc". Las librerías necesarias para que un programa funcione pueden estar almacenadas en el directorio "/usr/lib".

La ventaja de esta estructura es que permite una mayor flexibilidad y modularidad al momento de instalar y gestionar programas en GNU/Linux. Por ejemplo, si se necesita una determinada librería, se puede instalar en el directorio "/usr/lib" y cualquier programa que necesite esa librería podrá encontrarla allí.

Por lo tanto, la estructura de directorios de GNU/Linux es muy detallada y cuenta con una gran cantidad de directorios, cada uno con una función específica. La siguiente lista muestra la inmensa cantidad de directorios que componen el árbol de linux:

* / (root): Es el nivel superior de la estructura de directorios de Linux, y es el único nivel que no está dentro de ningún otro directorio. Contiene todos los demás directorios del sistema, y es el punto de partida para cualquier ruta de archivo o directorio en el sistema.
* /bin: Contiene los archivos binarios esenciales del sistema, como comandos y programas que se utilizan con frecuencia en el arranque y funcionamiento del sistema.
* /boot: Contiene los archivos necesarios para el arranque del sistema, incluyendo los archivos del kernel, los archivos de configuración de inicio y los archivos de carga inicial.
* /dev: Contiene los dispositivos de hardware del sistema, como unidades de disco, dispositivos de entrada y salida, y otros dispositivos de hardware conectados al sistema.
* /etc: Contiene los archivos de configuración del sistema, incluyendo los archivos de configuración de los servicios del sistema, como los archivos de configuración de red, los archivos de configuración de los servicios de usuario y los archivos de configuración de seguridad.
* /home: Contiene los directorios personales de los usuarios del sistema. Cada usuario del sistema tiene un directorio dentro de /home que contiene sus archivos personales y de configuración.
* /lib: Contiene las bibliotecas compartidas del sistema, que son archivos que contienen código que se utiliza en múltiples programas y aplicaciones.
* /media: Contiene los puntos de montaje para los dispositivos de almacenamiento extraíbles, como los discos duros externos y las unidades flash USB.
* /mnt: Contiene los puntos de montaje para los sistemas de archivos temporales y los sistemas de archivos en red.
* /opt: Contiene los paquetes de software adicionales que no están incluidos en la distribución del sistema base.
* /proc: Contiene información sobre el estado actual del sistema, como la memoria del sistema, los procesos en ejecución y los controladores de dispositivos.
* /root: Es el directorio personal del usuario "root", que es el usuario administrativo del sistema.
* /run: Contiene los archivos y directorios temporales que se utilizan durante el inicio del sistema y en el funcionamiento diario del sistema.
* /sbin: Contiene los comandos y programas esenciales del sistema que sólo pueden ser ejecutados por el usuario "root" o por otros usuarios con permisos especiales.
* /srv: Contiene los archivos de datos de los servicios del sistema, como los archivos de datos de los servidores web y de los servidores de correo electrónico.
* /sys: Contiene información sobre los dispositivos y controladores del sistema, que se utiliza para interactuar con el kernel del sistema.
* /tmp: Contiene los archivos temporales del sistema, que se eliminan automáticamente cuando el sistema se reinicia.
* /usr: Contiene los archivos y programas no esenciales del sistema, como las aplicaciones y programas adicionales que se pueden instalar en el sistema.
* /var: Contiene los archivos de datos variables del sistema, como los archivos de registro del sistema, los archivos de correo electrónico y los archivos de estado del sistema.

**2.- Describa el uso del comando “mount”**

El comando "mount" en Linux se utiliza para montar sistemas de archivos en un directorio determinado. Esto permite acceder a los archivos y datos almacenados en ese sistema de archivos como si fueran parte del sistema de archivos local.

El comando "mount" tiene la siguiente sintaxis básica: **mount [-o options] device directory**

Donde:

"device" es el dispositivo o archivo que contiene el sistema de archivos que deseas montar. Puede ser un dispositivo físico, como una unidad de disco duro o una unidad flash USB, o un archivo que contiene una imagen de un sistema de archivos.

"directory" es el directorio en el que deseas montar el sistema de archivos.

"-o options" son las opciones de montaje para el sistema de archivos. Estas opciones incluyen cosas como el modo de solo lectura, el tipo de sistema de archivos, el modo de propietario y los permisos.

Este sistema se basa en la siguiente metodología de uso:

1. Si el dispositivo es removible (un disco floppy o un CD), el usuario lo ingresa en la lectora.
2. Mediante el comando mount, asigna ese dispositivo a un directorio denominado punto de montaje.
3. Desde el punto de montaje, el usuario podrá leer y escribir los datos almacenados en ese dispositivo.
4. Una vez que haya terminado de usar el dispositivo, el usuario debe eliminar el enlace entre el dispositivo y el punto de montaje haciendo uso del comando umount.

Veamos un ejemplo para aclarar las cosas. Si un usuario quiere utilizar un disco floppy, entonces debe insertarlo en la disquetera y ejecutar el siguiente comando:

**“mount /dev/floppy /mnt/floppy”**

El comando mount requiere dos parámetros. El primero de ellos es el dispositivo por montar. Recuerden que en el directorio /dev/ están todos los archivos que representan los diferentes dispositivos básicos de nuestro sistema. Por consiguiente, /dev/ floppy representa nuestra primera disquetera. El segundo parámetro que requiere el comando mount es el punto de montaje, o sea: el directorio que usaremos para acceder a ese disco flexible. El directorio /mnt/floppy que pusimos en el ejemplo puede no existir. En tal caso, el usuario puede crear su propio directorio o utilizar cualquier otro (aunque la primera opción es la más recomendable).

Algunos ejemplos de uso comunes del comando "mount" incluyen:

* Montar una unidad flash USB en un directorio de tu sistema de archivos local para acceder a los archivos en la unidad.
* Montar una partición de disco duro en un directorio de tu sistema de archivos local para acceder a los archivos en la partición.
* Montar una imagen de disco que contenga un sistema de archivos para acceder a los archivos en la imagen.

Es importante recordar que para montar un sistema de archivos, necesitas tener los permisos necesarios para hacerlo. Normalmente, solo el usuario root tiene los permisos necesarios para montar sistemas de archivos en Linux.

**3.- Explique los permisos de los archivos de Linux. Dé ejemplos.**

Una de las características de los sistemas operativos UNIX es la posibilidad de que cada archivo tenga ciertos permisos de acceso y utilización. En realidad, esta característica la provee el sistema de archivos.

Cada archivo de nuestro sistema de archivos (incluidos los directorios) posee ciertos permisos de acceso. Éstos son, básicamente, tres: Lectura (r), Escritura (w) y Ejecución (x). Estos permisos pueden ser aplicados al usuario dueño del archivo, al grupo de usuarios al cual pertenece y a todos los demás usuarios del sistema. Esto significa que podemos asignar permisos de ejecución al dueño de un archivo, pero podemos asignar permisos de sólo lectura a todos los demás usuarios.

En el comando ls existe el parámetro es -l que ofrece informacion detallada de los archivos y directorios:

**[localhost:-/temporal] hfarena% ls -1**

**drwxrwxrwx 1 hfarena staff 25 Hay 13 15:28 documento.txt**

**[localhost:-/temporal] h£arena%**

La columna que más nos interesa ahora es la primera (drwxrwxrwx). Las siglas "drwxrwxrwx" que se muestran en la salida del comando ls en Linux representan los permisos de un directorio o archivo específico. La primera letra indica si el objeto es un archivo regular (-) o un directorio (d), mientras que las siguientes tres letras (rwx) indican los permisos del propietario, las siguientes tres letras (rwx) indican los permisos del grupo y las últimas tres letras (rwx) indican los permisos de otros usuarios.

Cada letra representa un permiso específico:

r (read) - permiso de lectura

w (write) - permiso de escritura

x (execute) - permiso de ejecución

Si una letra está ausente, significa que el permiso correspondiente no está otorgado. Por ejemplo, en el siguiente listado:

**drwxr-xr-x 2 usuario grupo 4096 feb 18 14:25 carpeta**

**-rw-r--r-- 1 usuario grupo 148 feb 16 12:34 archivo.txt**

El primer objeto es un directorio (d), y los permisos son rwxr-xr-x, lo que significa que el propietario tiene permisos de lectura, escritura y ejecución, el grupo tiene permisos de lectura y ejecución, y otros usuarios tienen permisos de lectura y ejecución. El segundo objeto es un archivo regular (-), y los permisos son rw-r--r--, lo que significa que el propietario tiene permisos de lectura y escritura, el grupo tiene permisos de lectura, y otros usuarios tienen permisos de lectura.

**4.- En qué consiste el comando “tuberías” Dé ejemplos**

El comando "tuberías" en Linux es una característica que permite tomar la salida de un comando y utilizarla como entrada para otro comando, sin necesidad de guardar la salida en un archivo temporal. La tubería se representa mediante el carácter | y se utiliza para redirigir la salida de un comando a la entrada de otro.

Por ejemplo, el siguiente comando utiliza la tubería para filtrar los resultados de la lista de archivos (ls) y mostrar solo aquellos archivos que contienen la cadena "documento":

**ls | grep documento**

En este caso, la salida del comando ls se envía a grep a través de la tubería, donde se filtra para mostrar solo los archivos que contienen la cadena "documento".

Otro ejemplo sería el siguiente comando, que utiliza la tubería para contar el número de líneas de un archivo de texto:

**cat archivo.txt | wc -l**

En este caso, la salida del comando cat (que muestra el contenido del archivo) se envía a wc (que cuenta el número de líneas), y el resultado final es el número de líneas en el archivo.

**5.- ¿Qué es el BASH? ¿Para qué sirve? Dé ejemplos**

Cuando un administrador precisa realizar algún pequeño programa para automatizar ciertas tareas administrativas, piensa en el lenguaje Bash. Este completo shell posee un lenguaje de programación muy poderoso.

BASH (Bourne-Again SHell) es un intérprete de comandos de Unix y un lenguaje de scripting que se utiliza principalmente en sistemas operativos basados en Unix y Linux. BASH es una evolución del shell original de Unix, el shell Bourne (sh), y se ha convertido en el shell predeterminado para la mayoría de las distribuciones de Linux.

El principal propósito de BASH es proporcionar una interfaz de línea de comandos para los usuarios de Unix y Linux, permitiéndoles interactuar con el sistema operativo y ejecutar comandos y programas. BASH también admite la programación de scripts para automatizar tareas y realizar secuencias de comandos complejas.

Algunos ejemplos de uso de BASH son:

* Navegación de archivos y directorios utilizando comandos como ls, cd, pwd, mkdir y rm.
* Manipulación de archivos y directorios utilizando comandos como cp, mv, chmod y chown.
* Comunicación con otros programas mediante el uso de tuberías (|), redirección de entrada/salida (< y >) y variables de entorno.
* Automatización de tareas mediante la creación de scripts de shell, que pueden ejecutar comandos, realizar operaciones de archivo y directorio, y manipular variables de entorno.
* Configuración de la shell mediante la personalización de variables de entorno, la creación de alias de comando y la definición de funciones de shell personalizadas.

Un ejemplo de un script hecho con Bash podria ser el siguiente:

**^1^IBIBIMI**

**# Ejemplo 1: Hola Mundo1**

**echo “Hola Mundo!"**

Luego, le otorgamos permisos de ejecución (755).

Este pequeño programa consta de tres líneas. La primera le indica al intérprete de comandos qué intérprete debe usar para ejecutar el lenguaje que fue utilizado en el programa. La segunda es sólo un comentario (en Bash, los comentarios comienzan con el símbolo "#"). Por último, la tercera línea es el comando echo, con un valor de cadena encerrado entre comillas. Este comando imprime la cadena en la terminal activa.

**6.- Nombre y ejemplifique 3 estructuras de control de Bash**

Estructuras condicionales:

Las estructuras condicionales son las que hacen interesantes a los scripts. Utilizándolas es posible crear menús con opciones, tomar diferentes decisiones sobre la base de una situación específica, hacer preguntas, y un sinfín de acciones más. La más famosa y utilizada de estas estructuras es el if condicional. Su forma básica es:

**if [ expresión ]**

**then acción 1**

**elif [expresión]**

**then acción 2**

**else accion3 fi**

La primera línea evalúa la expresión entre corchetes. Si es verdadera, ejecuta "ac- cionl"; si es falsa, salta la ejecución hasta el else y ejecuta "accion2". Por último, se cierra la estructura con un fi.

Estructuras de bucles:

Las estructuras de bucles permiten repetir una acción un determinado número de veces o hasta que se cumpla una condición. En Bash, las estructuras de bucles más comunes son for y while. Aquí tienes un ejemplo de cada una:

**for i in 1 2 3 4 5**

**do**

**echo $i**

**done**

**while [ $i -le 5 ]**

**do**

**echo $i**

**i=$((i+1))**

**done**

Estructuras de control de señales:

Las estructuras de control de señales permiten manejar las señales que recibe un programa, por ejemplo, cuando el usuario pulsa Ctrl+C para interrumpir la ejecución del script. En Bash, la estructura de control de señales más común es trap. Aquí tienes un ejemplo:

**trap "echo Se ha pulsado Ctrl+C" SIGINT**

**while true**

**do**

**sleep 1**

**done**

En este ejemplo, el trap captura la señal SIGINT y muestra un mensaje por pantalla. El bucle infinito se utiliza para mantener el script en ejecución hasta que se pulsa Ctrl+C y se envía la señal SIGINT.

**7.- Defina qué es un proceso.**

Es una instancia en ejecución de un programa que está siendo controlado por el sistema operativo. Cada proceso tiene un identificador único (PID) y es gestionado por el kernel del sistema operativo.

Cada proceso existe (al menos hasta cierto punto) dentro de una memoria física de la máquina. Puesto que Linux (y Unix) están diseñados para ser un entorno multiusuario, la memoria asignada a un proceso está protegida y ningún otro proceso puede acceder a ésta.

Cada proceso interactúa hasta cierto punto con el sistema de archivos para leer y escribir información que existe o existirá después del ciclo de vida del proceso.

Cada proceso mantiene su propia lista de pares nombre-valor, conocida como variables de entorno o en general como el entorno del proceso.

Cada proceso se identifica con un PID o id del proceso asignado en el momento de su creación.

Cada proceso se ejecuta bajo el contexto de un determinado usuario (o más exactamente de un id de usuario determinado) y bajo el contexto de una colección de id de grupo (generalmente, todos los grupos a los que pertenezca el usuario).

Cada proceso también registra estadísticas para trazar la cantidad de recursos del sistema utilizados, el número de archivos abiertos, la cantidad de tiempo de CPU, etc.

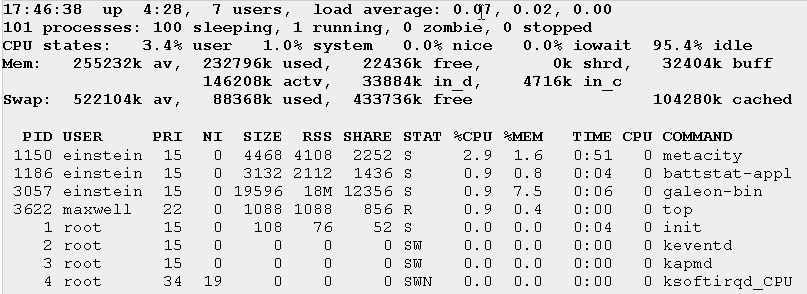
**8.-¿Qué programas se utilizan para realizar el seguimiento de los procesos?**

Los sistemas GNU/Linux cuentan varios programas para efectuar el seguimiento de los procesos que se están ejecutando en el sistema. Entre los mas usados en la interfase de texto están los programas ps y top.

* Ps: Sin ninguna opción dará la lista de procesos que están corriendo desde la terminal donde se ejecuto el ps. Puede mostrar información detallada sobre cada proceso, como su identificador de proceso (PID), estado, consumo de CPU y memoria, entre otros datos.
* Top: El comando ps nos muestra una radiografía de los procesos en el momento, pero no nos muestra los cambios que se van teniendo. Para esto contamos con el comando top. El mismo muestra en tiempo real la situación de los procesos que se están ejecutando en el sistema, ordenados por defecto según el porcentaje de la CPU que estén usando. Al ejecutarlo se podrá ver otra información adicional, como la cantidad de usuarios que están en el sistema, cuantos procesos están corriendo y de estos cuantos están activos, cuantos durmiendo, cuantos en proceso de terminar (ZOMBIE) y cuantos finalizados. Además se podrá ver la cantidad de memoria física total, la cantidad usada y la cantidad libre; así como también se podrá obtener la misma información de la memoria swap. Lo más importante es que esta información se ira actualizando automáticamente cada tanto tiempo, por defecto 5 segundos, y que podremos ir alterando lo que va mostrando.

9.- ¿Cuál es el comando que monitorea los procesos?

El comando top mostrará un cuadro de procesos actualmente en ejecución, que se actualiza cada ciertos segundos. Este comando muestra información en tiempo real sobre los procesos en ejecución en el sistema, incluyendo el uso de CPU, memoria, tiempo de ejecución y otros detalles relevantes.



10.- Defina los estados de los procesos.

En sistemas operativos Unix/Linux, un proceso puede encontrarse en uno de los siguientes cinco estados:

* Ejecutable: El proceso está listo para ser ejecutado y está esperando que el planificador del sistema le asigne tiempo de CPU para su ejecución.
* Dormido voluntario: El proceso ha solicitado esperar por algún evento o condición para continuar su ejecución, como por ejemplo una entrada de teclado o un evento de red. En este estado, el proceso está inactivo y no utiliza tiempo de CPU.
* Dormido involuntario: El proceso está esperando por algún evento externo para continuar su ejecución, como por ejemplo la finalización de una operación de entrada/salida. En este estado, el proceso está inactivo y no utiliza tiempo de CPU.
* Detenido: El proceso ha sido detenido temporalmente por algún motivo, como por ejemplo por un depurador o por un error en su ejecución. En este estado, el proceso no puede continuar su ejecución hasta que se le indique.
* Zombi: El proceso ha completado su ejecución, pero aún no ha sido eliminado de la tabla de procesos del sistema. Esto sucede cuando el proceso padre no ha esperado a que su hijo termine su ejecución, y por lo tanto no ha liberado los recursos del proceso hijo. Un proceso zombi consume muy pocos recursos del sistema, pero puede acumularse y agotar la tabla de procesos del sistema si no se maneja adecuadamente.
* Es importante destacar que estos estados son conceptos teóricos para entender cómo funcionan los procesos en el sistema operativo, ya que en la práctica un proceso puede cambiar de estado muy rápidamente o incluso estar en varios estados al mismo tiempo (por ejemplo, ejecutable y dormido voluntario).

Es importante destacar que estos estados son conceptos teóricos para entender cómo funcionan los procesos en el sistema operativo, ya que en la práctica un proceso puede cambiar de estado muy rápidamente o incluso estar en varios estados al mismo tiempo (por ejemplo, ejecutable y dormido voluntario).

**11-Desarrolle un cuadro comparativo de los comandos por consola enfocados a redes de linux y microsoft.**

| Descripción | Linux | Microsoft |
| --- | --- | --- |
| Mostrar la configuración IP de un sistema | “ip addr show” o “ifconfig” | “ipconfig” |
| Verificar la conectividad con un host remoto | “ping” | “ping” |
| Mostrar las rutas de red del sistema | “ip route show” o “route” | “route print” |
| Mostrar información de la tabla ARP del sistema | “arp” | “arp -a” |
| Verificar la resolución de nombres DNS | “nslookup” | nslookup” |
| Mostrar los puertos abiertos en un sistema | “netstat” o “ss” | “netstat” |
| Verificar la conectividad a un puerto en un host remoto | “telnet” o “nc” | “telnet” o “nc” |
| Escanear puertos en un host remoto | “nmap” | “nmap” |
| Configurar la dirección IP de una interfaz de red | “ip addr add” o “ifconfig” | “netsh interface ipv4 set address” |
| Configurar la ruta predeterminada del sistema | “ip route add default” o “route” | “route add” |
| Configurar el servidor DNS del sistema | “systemd-resolve” o “resolvectl” | “netsh interface ipv4 set dns” |
| Configurar un servidor DHCP en la red | “systemd-networkd” o “dhclient” | “ipconfig /renew” o  “ipconfig /release” |
| Mostrar estadísticas de red del sistema | “netstat” o “ss” | “netstat -s” |
| Verificar la configuración de firewall del sistema | “iptables” | “netsh advfirewall” |