Conceptos de Sistemas Operativos

Ramiro Cabral

September 29, 2023

1 Caracteristicas de GNU/Linux

1.1 Caracteristicas mas relevantes

- Es un sistema operativo UNIX-Like
- Es multiusuario
- Es multitarea
- Es altamente portable
- Es case-sensitive
- Es de codigo abierto
- Todo es un archivo(incluso los dispositivos y directorios)

1.2 POSIX

- Conjunto de estandares que define varias interfaces de herramientas, comandos y API para Sistemas Operativos de tipo UNIX.
- Creada por la IEEE, define una interfaz estandar del sistema operativo y el entorno, inclutendo un interprete de comandos (shell).

1.3 Distribuciones

Una distribucion es una customizacion de GNU/Linux formada por una version de kernel y determinados programas con sus configuraciones.

2 Estructura de GNU/Linux

2.1 3 componentes fundamentales de GNU/Linux

- Kernel(nucleo)
- Shell (interprete de comandos)
- FileSystem (sistema de archivos)

3 Kernel de GNU/Linux

3.1 Que es el Kernel y cuales son sus funcines principales?

Es el encargado de que el software y el hardware puedan trabajar juntos. Sus funciones principales son:

- Administracion de memoria.
- Administracion de la CPU.
- Administracion de la E/S.

3.2 Arquitectura del Kernel

El kernel es un nucleo monolitico hibrido:

- Los drivers y el codigo del mismo se ejecutan en modo privilegiado.
- Lo que lo hace hibrido es la capacidad de cargar y descargar funcionalidad a traves de modulos.

4 Shell (Interprete de comandos)

4.1 Que es?

Es el modo de comunicacion que el usuario posee con el sistema operativo. El mismo ejecuta programas mediante el ingreso de comandos.

5 File System (sistema de archivos)

5.1 Que es el Fyle System?

Es una estructura de datos que utiliza el sistema operativo para controlar como se almacenan y recuperan los datos en un medio de almacenamiento.

5.2 Jerarquia de directorios en GNU/Linux

- **/** : Tope de la estructura de directorios.
- **/home** : Se almacenan los archivos de los usuarios.
- **/var** : Informacion que varia de tamanio en el tiempo.

- **/etc** : Archivos de configuracion del sistema.
- **/bin** : Archivos binarios y ejecutables.
- **/dev** : Enlace a dispositivos.
- **/usr** : Aplicaciones de usuarios.
- **/tmp** : Archivos temporales.
- **/boot** : Informacion del booteo de la maquina.

6 Bootstrap (Arranque del sistema)

6.1 BIOS(Basic I/O System)

La BIOS es un chip instalado en la placa base con un firmware que contiene una serie de subrutinas basicas del procesador para el arranque del sistema. Actua como un intermediario entre la CPU y los dispositivos de I/O.

6.2 MBR (Master Boot Record)

- Sector reservado del disco fisico (cilindro 0, cabeza 0, sector 1).
- Existe un MBR en todos los discos.
- Su tamano es de **512** bytes.
  - 446 bytes corresponden al MBC (Master Bootloader Code).
  - otros 64 bytes corresponden a la tabla de particiones del disco (partition table).
  - Los ultimos 2 bytes son libres, pero pueden usarse para firmar el MBR.
  - MBR tiene espacio acotado para la tabla de particiones, tenemos dos opciones:
    - 4 particiones primarias.
    - 3 primarias y una extendida con sus particiones logicas.

6.2.1 MBC

- Es un pequeno codigo que permite arrancar el SO.
- La ultima accion del BIOS es ejecutar el MBC. Lo lleva a memoria y lo ejecuta.
- Si se tiene un SO instalado, generalmente corresponde al bootloader por defecto del sistema. Tambien puede instalarse en el un bootloader diferente (multietapa) como por ejemplo GRUB.

6.3 GPT (Guid Partition Table)

El sistema GPT es un estandar de particiones de disco. Fue creado por Intel como reeemplazo para el estandar MBR.

- Utiliza un modo de direccionamiento logico (logical block addressing **LBA**) en lugar de la *cylinder-header-sector* utilizada por MBR.
- Funciona con sistemas UEFI.
- Se mantiene un MBR legacy para mantener la compatibilidad con el esquema BIOS.
- LBA 0: MBR legacy.
- LBA 1: GPT Header.
- LBA 2...: Tabla de particiones.
- La cabecera GPT y la tabla de particiones estan escritas al principio y al final del disco por redundancia.

6.4 Gestores de Arranque

Programas o software que se encargan de controlar el proceso de inicio de un sistema operativo. Su funcion principal es la de proporcionar al usuario la opcion de elegir que sistema operativo desea iniciar al prender la maquina.

- Carga una imagen del kernel de algun SO instalado en una particion.
- Se ejecuta luego del codigo de BIOS.
- Existen 2 modos de instalacion:
  - En el MBR (suele ubicarse en el MBC).
  - En el sector de arranque de la particion raiz o activa.

6.4.1 GRUB (Grand Unified Bootloader)

Gestor de arranque comun en sistemas Linux.

- Altamente configurable, puede administrar multiples sistemas operativos y configuraciones del sistema.
- En el MBR se encuentra la fase 1, que luego carga la fase 1.5.
- La fase 1.5 se encuentra en los siguientes 30kb de disco. Carga la fase 2.
- La fase 2 posee la interfaz de usuario, y carga el Kernel seleccionado.
- Se configura a traves del archivo */boot/grub/menu.lst*.
  En la version 2 de GRUB, el archivo de configuracion se encuentra en */boot/grub/grub.cfg*

7 Proceso de arranque del sistema

- En las arquitectuas x86, el BIOS es el responsable de iniciar la carga del SO a traves del MBC
- Carga el programa de booteo (desde el MBR).
- El gestor de arranque lanzado desde el MBC carga el Kernel.
  - Prueba y hace disponibles los dispositivos.
  - Luego pasa el control al proceso **init**.

8 UEFI (Unified Extensible Firmware Interface)

- Define la ubicacion de gestor de arranque.
- Define la interfaz entre el gestor de arranque y el firmware.
- Expone infgormacion para los gestores de arranque con:
  - Informacion de hardware y configuracion de firmware.
  - Punteros a rutinas que implementan los servicios que el firmware ofrece a los bootloaders u otras aplicaciones.
  - El bootloader ahora es un tipo de aplicacion UEFI. Ahora, para Grub, ya no es necesario el arranque en varias etapas.

8.1 Secure Boot

- Posee mecanismos para un arranque libre de codigo malicioso.
- Las aplicaciones y drivers UEFI son validadas para verificar que no fueron alteradas.
- Se utilizan pares de clabes asimetricas.
- Se almacenan en el firmware una serie de claves publicas que sirven para validar que las imagenes esten firmadas por un proveedor autorizado.
- Si la clave privada esta vencida o fue revocada la verificacion puede fallar.

9 SystemV

9.1 Proceso de Inicio en GNU/Linux

1. Se comienza a ejecutar el codigo del Bios.
2. El BIOS ejecuta el POST.
3. El bios lee el sector de arranque (MBR).
4. Se carga el gestor de arranque (MBC).
5. El bootloader carga el kernel y el initrd.
6. Se monta el initrd como sistema de archivos raiz y se inicializan componentes esenciales.
7. El kernel ejecuta el proceso init y se desmonta el initrd.
8. Se lee el /etc/inittab.
9. Se ejecutan los scripts apuntados pr el runlevel 1.
10. El final del runlevel 1 le indica que vaya al runlevel por defecto.
11. Se ejecutan los scipts apuntados por el runlevel por defetco.
12. El sistema esta listo para usarse.

9.2 Init

initrd :Es cargado como parte del procedimiento de booteo del kernel. Contiene un set minimio de ejecutables y archivos necesarios para la inicializacion del kernel del sistema.

- Carga los subprocesos necesarios para el correcto funcionamiento del SO.
- Posee el PID 1 y se encuentra usualmente en /sbin/init.
- En SystemV se lo configura a traves del archivo */etc/inittab*.
- No tiene padre y es el padre de todos los procesos (pstree).
- Es el encargado de montar los filesystems y de hacer disponible los demas dipositivos.

9.3 RunLevels

- Hacen referencia al modo en que arranca Linux.
- Define que servicios del sistema estan operando.
- El proceso de arranque es dividido en diferentes niveles.
- Se encuentran definidos en */etc/inittab*.
- Syntaxis: ***id:nivelesEjecucion:accion:proceso***.
  - **ID:** identifica la entrada en inittab.
  - **nivelesEjecucion:** el/los niveles de ejecucion en los que se realiza la accion.
  - **Accion:** describe la accion a realizar.
  - **Proceso:** proceso exacto que sera ejecutado.
- Existen 7 distintos RunLevels, y cada uno puede iniciar un conjunto de procesos al arranque o apagado del sistema.
- Los scripts que se ejecutan se encuentran en */etc/init.d*.
- En */etc/rcX.d* (con X entre 0 y 6) hay links a los archivos del */etc/init.d*.
- Formato de los links: `[S-K]<orden><nombreScript>`.

- Para administrar el orden de los enlaces simbolicos, y resolver dependencias entre ellos se utiliza **insserv**.
- insserv usa cabeceras (De tipo LSB) en los scripts del init.d para especificar la relacion con otros scripts rc.

10 Upstart

- Reemplazo propuesto para SystemV.
- Ejecucion de trabajos en forma asincronica a traves de eventos, a diferencia de SystemV que es estrictamente sincronico (*dependency-based*).
- Estos trabajos se denominan ***jobs***.
- Los jobs definen servicios o tareas a ser ejecutadas por init.
- Son definidos en */etc/init*.
- Los jobs son ejecutados ante eventos, y es posible crear nuestros propios eventos.
- Cada job posee un objetivo (goal start/stop) y un estado (state).

11 SystemD

- Sistema creado por RedHat (the devil himself) que centraliza la administracionde demons y librerias del sistema.
- Mejora el paralelismo de booteo.
- Puede ser controlado por el comando ***systemctl***.
- Compatible con SystemV.
- Los runlevels son reemplazados por ***targets***.
- Las unidades de trabajo son denominadas ***units*** de tipo:
  - ***Service:*** controla un servicio particular.
  - ***Socket:*** encapsula, un IPC, un socket del sistema o filesystem.
  - ***Target:*** agrupa units o establece puntos de sincronizacion.
  - ***Snapshot:*** almacena el estado de un conjunto de unidades que puede ser restablecido mas tarde.

11.1 Activacion por Socket

- Es un mecanismo de iniciacion bajo demanda.
- Cuando el socket recibe una conexion spawnea el servicio y le pasa el socket.
- No hay necesidad de definir dependencias entre servicios, ya que se inician todos los sockets en primier medida.

11.2 cgroups

  - Permite organizar un grupo de procesos en forma jerarquica.
  - Agrupa conjuntos de procesos realcionados.

11.3 fstab

- Define que particiones se montan al arranque.
- Su configuracion se encuentra en ***/etc/fstab***.

12 Usuarios en GNU/Linux

- Cada usuario debe poseer credenciales para acceder al sistema.
  - *root:* administrador del sistema (superuser).
  - *otros:* usuarios del sistema estandar (*/etc/sudoers*).
- Archivos de configuracion:
  - ***/etc/passwd***
    - Cada linea representa a un usuario.
    - Posee informacion general del usuario (grupo,nombre,descripcion,homedir,shell,etc).
  - ***/etc/group***
    - Muestra cada grupo del sistema.
    - Cada usuario puede tener grupos secundarios.
  - ***/etc/shadow***
    - Posee nombres y contrasenas encriptadas (solo accesible por el root).

12.1 UID y GID

- Un **UID** (User identifier) es un numero asignado por Linux a cada usuario del sistema. Este numero es usado para identificar el usuario y para determinar a que recursos puede acceder.
- Son almacenados en el archivo */etc/passwd*.
- El Root posee el UID 0.
- Las UID por debajo de 1000,por convencion, son reservadas para usuarios creados por el sistema, servicios y otras cuentas especiales.
- Al crear un nuevo usuario, se le asignara un UID mayor a 1000.
- Un **GID** (group identifier) es un numero asignado a cada grupo del sistema.
- Son almacenados en el archivo */etc/groups*.
- Los primeros 100 GIDs son reservados para el uso del sistema.
- El GID 0 corresponde al grupoo root.
- El GID 100 correponde al gruop 'users'.

13 Procesos

- Cuando un programa es ejecutado, el sistema le provee una instancia especial al proces. La instancia consiste en todos los servicios/recursos que podria utilizar el proceso en la ejecucion.
- Un numero de ID de 5 digitos es asignado al proceso, este numero es denominado **PID**. Cada proceso posee un unico PID.
- Un processo que se conecta a la terminal es llamado un **foreground job**. Es llamado asi ya que puede comunicarse con el usuario mediante la pantalla y el teclado.
- Un proceso que se desconecta de la terminal y no puede comunicarse con el usuario es llamado un **background job**.
- Si un proceso en background requiere inhteraccion con el usuario, el mismo se detiene y espera hasta establecer una conexion con la temrinal.

14 Paquetes

- Los paquetes son archivos que contienen archivos, y no utilizan ningun tipo de compresion.
- Los archivos comprimidos representan la compresion, utilizando algun tipo de algoritmo, de uno o varios archivos comunes.

15 Argumentos y valor de retorno

1. **$0** contiene la invocacion del script.
2. **$1,$2,$3...** contienen cada uno de los argumentos.
3. **$#** contiene la cantidad de argumentos recibidos.
4. **$*** contiene la lista de todos los argumentos.
5. **$?** contiene en todo momento el valor de retorno del ultimo comando ejecutado.

16 exit

1. Causa la terminacion de un script.
2. Puede devolver cualquier valor entre 0 y 255.
   (a) El valor 0 indica que el script se ejecuto de forma exitosa.
   (b) Un valor distinto indica un codigo de error.
   (c) Se puede consultar el exit status imprimiendo la variable $?.