

En Linux dispositivos de almacenamiento

no tiene sub

- `/dev/hda`: configurado como Master en el 1º bus IDE
- `/dev/hdb`: configurado como Slave en el 1º bus IDE
- `/dev/hdc`: configurado como Master en el 2º bus IDE
- `/dev/hdd`: configurado como Slave en el 2º bus IDE

`hda1` → primera partición del master del 1er bus.

`udev` → gestor de dispositivos

- Gestiona dinámicamente recursos de `/dev`, solo en base a hw.
- Hotplug, permite conectar disp. después de su inicio.
- persistent device naming → nombrar dispositivos, mantener referencia.

`UUID` → universal unique identifier

→ ÚNICO en el mundo

En gr. Linux no hay formato de partición

→ Case sensitive

→ Separación de entorno gráfico y texto → entornos diferentes

no es necesario tener credenciales p. acceder

en `/etc/passwd` → usuarios y atributos

Nombre, id, id grupo, contraseña hash, interprete de comando default.

`/etc/group` → grupos separados de roles los users.

permiso → `leown R y` permiso de grupo,
`groum w 2` de lectura y de otros,
`et... .. 1`

$0 \leq n < 2^8$ o $n > 2^8$ de otro
 Ejecución x 1

chmod \rightarrow p. por permisos.

Archivo que tiene, grupo, otro

chmod r s s

Bootloader \rightarrow Cargador de arranque
 \rightarrow Cargador multipaso
 \rightarrow muchos programas cargan arranque

System V (5)

- ① Bootloader por \rightarrow primeros sobre hardware
- ② BIOS lee sector de arranque (MBR) \rightarrow devuelve de arranque maestro.
- ③ Carga gestor de arranque
- ④ Bootloader carga kernel e initrd \rightarrow sistema de archivos raíz q' carga componentes esenciales
- ⑤ Kernel ejecuta init y se detiene initrd \rightarrow sistema de archivos pronto
- ⑥ init \rightarrow padre de todos los procesos. lee /etc/passwd (configuración)
- ⑦ se ejecuta scripts relacionados a runlevel 1
- ⑧ se ejecuta runlevel x detiene al final de runlevel 1

init \rightarrow padre, todos los procesos, todos los subprocesos de PID
 \rightarrow PID \rightarrow proceso 1
 \rightarrow Ejecución de montado sistemas de archivos

Runlevel \rightarrow Modo de ejecución del SO
 \rightarrow Qué servicios se ejecutan / están
 \rightarrow Permite cambiar entre la configuración de runlevels

→ que se ejecuten a lo largo del tiempo
 → en el /etc/crontab permite la configuración de runlevels

- 0: halt (parada) Scripts que hacen que apagado sea seguro, tierra todo lo prendido
- 1: single user mode (monousuario) solo un usuario puede usar la compu
- 2: multiuser, without NFS (modo multiusuario sin soporte de red)
- 3: full multiuser mode console (modo multiusuario completo por consola)
- 4: no se utiliza
- 5: X11 (modo multiusuario completo con login gráfico basado en X) todo
- 6: reboot Halt + inicio

Los scripts q se ejecutan son los de /etc/cron.d

RCLO.d → scripts de lvl 0.

se pueden agregar 7 sacar scripts a runlevels

system v → Administrar scripts prehechos como cron jobs.

→ formato específicos.

system v → blog de futurepress donde se le ha hecho actual.

(Síncrono)

systemd → Asíncrono. → se ejecuta trabajos once a hora.

→ Job de inicio en /etc/cron (.conf)

→ mask, services → inicio.

→ inicio

→ Job para tener 1 o muchos jobs.

systemd → Asíncrono

→ targets

→ manage units & targets

- **Service:** controla un servicio particular (.service)
- **Socket:** encapsula IPC, un socket del sistema o file system FIFO (.socket) → socket-based activation
- **Target:** agrupa units o establece puntos de sincronización durante el booteo (.target) → dependencia de unidades
- **Snapshot:** almacena el estado de un conjunto de unidades que puede ser restablecido más tarde (.snapshot)
- etc.

→ Activar/Desactivar unit
→ (como unit) depende a
punto de arranque.

System S → carga cosas al arranque

System MD → **sockets** → spawnear cuando se los necesita.

→ no se ejecuta todo los servicios en boot

→ **targets** → se activan y desactivan.

unit → run

→ pueden estar activos o inactivos

Socket → puerto

Groups → organizar grupo de procesos jerárquicamente.

FSMB → qué particiones se montan en arranque.

Resolución → p. cambiar salida error de la shell

→ destructivo / no destructivo → escribe al final

→ si ya existe borra y escribe

PIPE 1 → comunica 2 procesos

→ salida de los comandos con salida del otro

\$ N | more