Anatomía del Proceso de Arranque en Linux

Del Power-On al Login

Chávez López B Alejandro

Sistemas Operativos

21 de octubre de 2025

¿Qué es bootear?

Definición

El *bootstrap* (arranque) es el proceso de cargar el Sistema Operativo en la memoria principal (RAM) para que tome el control de la computadora.

¿Qué es bootear?

Definición

El *bootstrap* (arranque) es el proceso de cargar el Sistema Operativo en la memoria principal (RAM) para que tome el control de la computadora.

- Es una cadena donde cada etapa prepara el entorno para la siguiente.

¿Qué es bootear?

Definición

El *bootstrap* (arranque) es el proceso de cargar el Sistema Operativo en la memoria principal (RAM) para que tome el control de la computadora.

- Es una cadena donde cada etapa prepara el entorno para la siguiente.
- Vamos las fases clave de este proceso en un sistema moderno.

¿Qué es bootear?

Definición

El *bootstrap* (arranque) es el proceso de cargar el Sistema Operativo en la memoria principal (RAM) para que tome el control de la computadora.

- Es una cadena donde cada etapa prepara el entorno para la siguiente.
- Vamos las fases clave de este proceso en un sistema moderno.

Flujo General

Firmware $\rightarrow$ Bootloader $\rightarrow$ Kernel $\rightarrow$ Init (PID 1) $\rightarrow$ Login

Etapa 1: El Firmware (BIOS / UEFI)

Es el primer software que se ejecuta, grabado en un chip de la tarjeta madre.

Funciones Clave:

1. **POST** (Power-On Self-Test): Realiza un chequeo rápido del hardware crítico (RAM, CPU). Si falla, emite códigos de pitido.
2. **Inicializar Hardware**: Activa los componentes básicos para el arranque.
3. **Localizar el Bootloader**: Su misión final es buscar en los discos (SSD, USB, etc.) un programa de arranque y cederle el control.

Es el puente entre el hardware puro y el software del SO.

Etapa 1: El Firmware (BIOS / UEFI)

Es el primer software que se ejecuta, grabado en un chip de la tarjeta madre.

Funciones Clave:

1. **POST** (Power-On Self-Test): Realiza un chequeo rápido del hardware crítico (RAM, CPU). Si falla, emite códigos de pitido.
2. **Inicializar Hardware**: Activa los componentes básicos para el arranque.
3. **Localizar el Bootloader**: Su misión final es buscar en los discos (SSD, USB, etc.) un programa de arranque y cederle el control.

Es el puente entre el hardware puro y el software del SO.

Comparación Crítica: BIOS vs. UEFI

BIOS (Legacy)

- Lee los primeros **512 bytes** del disco, el **MBR** (Master Boot Record).
- Limitado a 4 particiones primarias.
- No arranca desde discos $>$ 2.2 TB.
- Proceso de 16-bits, lento y menos seguro.

UEFI (Moderno)

- Es un mini-SO que lee archivos (`.efi`).
- Busca en una partición **ESP** formateada en FAT32.
- Usa particiones GPT (hasta 128 particiones).
- Habilita *Secure Boot* para prevenir rootkits.

Hoy, casi todos los sistemas operativos (Linux, Windows, macOS) usan UEFI.

Tablas de Particiones: De MBR a GPT

MBR (Master Boot Record)

- Esquema antiguo ligado a BIOS.
- Límite de 4 particiones primarias (o 3 primarias y 1 extendida).
- Tamaño máximo de disco de 2.2 TB.
- No tiene redundancia; si el MBR se corrompe, el disco es ilegible.

GPT (GUID Partition Table)

- Estándar moderno usado con UEFI.
- Permite hasta 128 particiones por defecto.
- Soporta discos de tamaño masivo (zettabytes).
- Más robusto: Guarda una copia de la tabla de particiones al final del disco.

Etapa 2: El Bootloader (GRUB 2)

Su única misión: cargar el **Kernel** del Sistema Operativo en memoria.

GRUB (GRand Unified Bootloader)

- Muestra el conocido menú de arranque.
- Lee su configuración de `/boot/grub/grub.cfg` para saber qué sistemas operativos puede cargar.
- Carga dos archivos clave en RAM:
  1. El Kernel (`vmlinuz-...`)
  2. El Disco RAM Inicial (`initramfs` o `initrd.img-...`)
- Finalmente, cede el control total al Kernel.

Profundizando en GRUB

Arranque Multietapa (Legacy BIOS)

Debido al límite de 512 bytes del MBR, GRUB se cargaba en fases:

- **Stage 1**: Código diminuto en el MBR que sabe encontrar al siguiente.
- **Stage 1.5**: Se aloja entre el MBR y la primera partición. Contiene drivers de sistema de archivos para leer el Stage 2.
- **Stage 2**: El GRUB completo, con el menú y toda su lógica, cargado desde `/boot/grub`.

(En UEFI, este proceso se simplifica cargando un único archivo `.efi`)

Profundizando en GRUB

Arranque Multietapa (Legacy BIOS)

Debido al límite de 512 bytes del MBR, GRUB se cargaba en fases:

- **Stage 1**: Código diminuto en el MBR que sabe encontrar al siguiente.
- **Stage 1.5**: Se aloja entre el MBR y la primera partición. Contiene drivers de sistema de archivos para leer el Stage 2.
- **Stage 2**: El GRUB completo, con el menú y toda su lógica, cargado desde `/boot/grub`.

(En UEFI, este proceso se simplifica cargando un único archivo `.efi`)

Edición de Parámetros del Kernel

En el menú de GRUB, la tecla **'e'** permite editar la línea de arranque.
Esto es útil para:

- Añadir `nomodeset` si hay problemas con el driver de video.
- Arrancar en modo rescate con `systemd.unit=rescue.target` para recuperar el sistema.

Etapas 3 y 4: Kernel + `initramfs`

El Problema del Huevo y la Gallina

El Kernel ya está en memoria, pero no sabe cómo leer tu disco duro (ej. NVMe, LVM, encriptado). ¡Los drivers para hacerlo están *dentro* de ese mismo disco!

Etapas 3 y 4: Kernel + `initramfs`

El Problema del Huevo y la Gallina

El Kernel ya está en memoria, pero no sabe cómo leer tu disco duro (ej. NVMe, LVM, encriptado). ¡Los drivers para hacerlo están *dentro* de ese mismo disco!

La Solución: `initramfs`

- Es un **sistema de archivos temporal que vive en la RAM**.
- Contiene los drivers y herramientas mínimas para acceder al disco real.
- El Kernel lo usa como raíz temporal, carga los módulos y monta el sistema de archivos real.
- La transición final se realiza con la llamada al sistema `pivot_root`.

Inicialización Final del Kernel

Tareas Post-`initramfs`

Una vez que el Kernel opera sobre el sistema de archivos real, realiza sus últimas tareas de inicialización:

- **Establecer Tabla de Interrupciones**: Define cómo el hardware (teclado, discos) notifica a la CPU que necesita atención.
- **Montar Sistemas de Archivos Virtuales**:
  - `/sys`: Expone información detallada sobre el hardware y los drivers.
  - `/proc`: Ofrece una vista en tiempo real de los procesos y el estado del sistema.

Inicialización Final del Kernel

Tareas Post-`initramfs`

Una vez que el Kernel opera sobre el sistema de archivos real, realiza sus últimas tareas de inicialización:

- **Establecer Tabla de Interrupciones**: Define cómo el hardware (teclado, discos) notifica a la CPU que necesita atención.
- **Montar Sistemas de Archivos Virtuales**:
  - `/sys`: Expone información detallada sobre el hardware y los drivers.
  - `/proc`: Ofrece una vista en tiempo real de los procesos y el estado del sistema.

Paso Final del Kernel

Solo cuando todo este entorno de bajo nivel está preparado, el Kernel está listo para su última acción: ejecutar el primer programa del espacio de usuario (`/sbin/init`).

Etapa 5: El Proceso `init` (PID 1)

Una vez el Kernel tiene acceso al disco, ejecuta el **primer proceso del espacio de usuario**: `/sbin/init`, que siempre tiene el **Process ID 1**.

Rol del PID 1

Es el ancestro de todos los demás procesos del sistema. Si este proceso muere, el sistema entra en pánico (Kernel Panic). Su trabajo es levantar el resto de los servicios del sistema.

Comparación Crítica: SysVinit vs. systemd

SysVinit (Antiguo)

- Basado en *Runlevels* (niveles de ejecución 0-6).
- Ejecutaba scripts de inicio de forma **secuencial**).
- Simple, pero muy lento.

systemd (Moderno)

- El estándar de facto actual.
- Basado en *Targets* y unidades (`.service`).
- Inicia servicios en **paralelo** tan pronto como sus dependencias se cumplen.
- Mucho más rápido y con más funcionalidades.

El Ecosistema systemd

`systemd` es más que un gestor de arranque; es una suite de administración.

Conceptos Clave

- **Unidades (Units)**: Archivos que describen recursos. Los más comunes son `.service`, `.mount`, `.socket` y `.target`.
- **Activación por Socket**: Un servicio (ej. SSH) no se ejecuta hasta que llega una conexión a su puerto, ahorrando recursos.
- **Journald**: Un sistema de logging centralizado y estructurado. Los logs se consultan con `journalctl`, permitiendo filtros avanzados.

Etapa 6: El Espacio de Usuario y el Login

`systemd` trabaja para alcanzar un objetivo o *Target* final.

`multi-user.target` (Modo Texto / Servidor)

Inicia todos los servicios de red y de sistema, y presenta una terminal de texto (TTY) para iniciar sesión.

`graphical.target` (Modo Gráfico / Escritorio)

Hace todo lo anterior y, adicionalmente, inicia el servidor gráfico (X11/Wayland) y el **Display Manager** (la pantalla de login gráfica que usas a diario).

¡En este punto, el proceso de arranque ha terminado y el sistema está listo para ser usado!

Etapa 6: El Espacio de Usuario y el Login

`systemd` trabaja para alcanzar un objetivo o *Target* final.

`multi-user.target` (Modo Texto / Servidor)

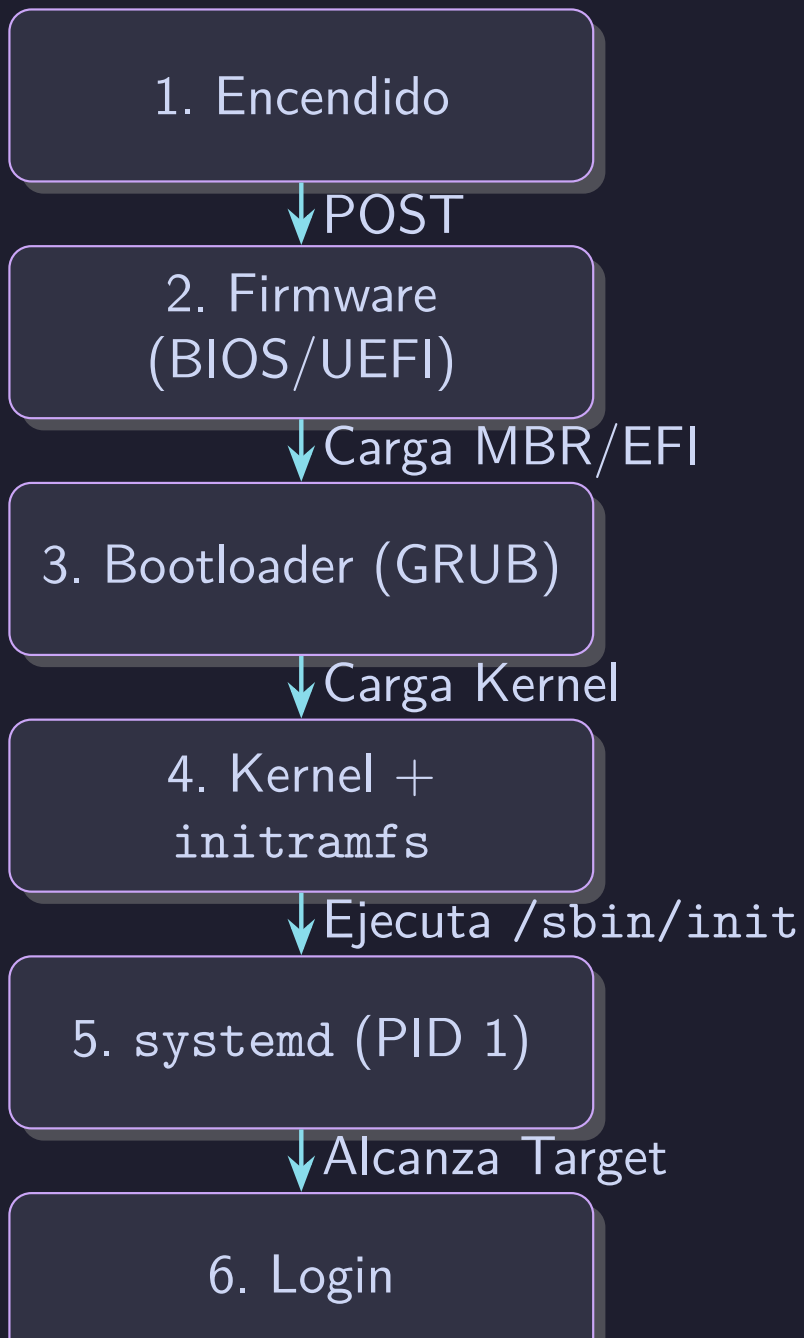
Inicia todos los servicios de red y de sistema, y presenta una terminal de texto (TTY) para iniciar sesión.

`graphical.target` (Modo Gráfico / Escritorio)

Hace todo lo anterior y, adicionalmente, inicia el servidor gráfico (X11/Wayland) y el **Display Manager** (la pantalla de login gráfica que usas a diario).

¡En este punto, el proceso de arranque ha terminado y el sistema está listo para ser usado!

Resumen del Proceso Completo



¿Y si algo sale mal? Herramientas de Diagnóstico

Cuando el proceso falla, estas son algunas situaciones y herramientas comunes:

Puntos de Falla Comunes

- **GRUB Rescue**: Una consola mínima que aparece si GRUB no puede encontrar sus archivos. Permite intentar cargar el sistema manualmente.
- **Kernel Panic**: Ocurre si el Kernel encuentra un error crítico del que no puede recuperarse (ej. no puede encontrar el sistema de archivos raíz). Detiene el sistema por completo.
- **Arranque Lento**: Si el sistema tarda mucho en iniciar, `systemd` ofrece una herramienta de diagnóstico:

  ```
  systemd-analyze blame
  ```

  Este comando lista los servicios y el tiempo que tardó cada uno en activarse.

Conclusión

Gracias por su atención.

Referencias I

Bovet, D. P., & Cesati, M. (2005). *Understanding the Linux Kernel* (3rd ed.). O'Reilly Media.

Free Software Foundation. (2023). *GNU GRUB Manual 2.12*. `https://www.gnu.org/software/grub/manual/grub/grub.html`

Intel Corporation. (2021). *Unified Extensible Firmware Interface (UEFI) Specification, Version 2.9*. UEFI Forum. `https://uefi.org/specifications`

Messmer, H-P. (2001). *The Indispensable PC Hardware Book* (4th ed.). Addison-Wesley Professional.

Nemeth, E., et al. (2017). *UNIX and Linux System Administration Handbook* (5th ed.). Prentice Hall.

Siever, E., & Love, R. (2012). *Linux in a Nutshell* (6th ed.). O'Reilly Media.