Universidad Nacional Autónoma de México

Exposición - Anatomía del Proceso de Arranque en Linux

Curso: Sistemas Operativos

Profesor: Dr. Gunnar Eyal Wolf Iszaveich

Autor: Chávez López B. Alejandro

Cuenta: 421112601

Grupo: 08

Contents

1 Introducción

El proceso de arranque, o *booting*, es uno de los aspectos más básicos de cualquier sistema operativo, tanto Linux, como Windows, MacOS o cualquier otro. Se refiere a la secuencia de operaciones que ejecuta la computadora desde el momento en que se enciende hasta que el sistema operativo está completamente cargado y listo para recibir comandos del usuario.

En el contexto de los sistemas Linux, este proceso es una cadena donde cada componente de software carga y cede el control al siguiente, incrementando la complejidad y funcionalidad en cada etapa.

Este documento detalla las seis etapas principales del proceso de arranque en un sistema Linux moderno, desde la inicialización del firmware hasta la carga del entorno de usuario.

2 Etapa 1: El Firmware (BIOS y UEFI)

El primer software que se ejecuta al encender la computadora es el firmware, un programa de bajo nivel almacenado en un chip de memoria no volátil (ROM o Flash) en la tarjeta madre.

2.1 POST y Arranque Básico

Inmediatamente después de recibir energía, la CPU comienza a ejecutar las instrucciones del firmware. La primera tarea es el **POST (Power-On Self-Test)**, un diagnóstico rápido que verifica la integridad y presencia del hardware crítico, como la CPU, la memoria RAM, la tarjeta de video, etc. Si se detecta un error grave en esta fase, el sistema emitirá una serie de códigos de pitido o de parpadeos LED de la motherboard.

Una vez que el POST concluye exitosamente, el firmware inicializa el hardware esencial y consulta su configuración (almacenada en CMOS) para determinar el orden de los dispositivos de arranque, el cual normalmente es el SSD o disco duro pero también pueden ser un USB o incluso por Red.

2.2 BIOS (Legacy)

El **BIOS (Basic Input/Output System)** es el firmware tradicional. Su método de arranque consiste en leer los primeros **512 bytes** del dispositivo de arranque seleccionado. Este sector se conoce como el **MBR (Master Boot Record)**. El MBR contiene dos elementos: una pequeña pieza de código ejecutable (el cargador de arranque de etapa 1) y la tabla de particiones.

Este método tiene limitaciones muy graves para sistemas modernos, como la incapacidad de direccionar discos de más de 2.2 TB y la restricción a solo cuatro particiones primarias (Messmer, 2001).

2.3 UEFI

El **UEFI (Unified Extensible Firmware Interface)** es el sucesor moderno del BIOS y opera de forma muy diferente. En lugar de ser un simple sistema de I/O, UEFI es

un mini-sistema operativo capaz de entender sistemas de archivos y particiones (Intel Corporation, 2021).

UEFI no utiliza el MBR para arrancar. En su lugar, busca una partición específica formateada en FAT32, conocida como la **ESP (EFI System Partition)**. Dentro de esta partición, busca y ejecuta archivos de aplicación EFI (con extensión `.efi`), que son los cargadores de arranque. Este método es más robusto, soporta discos mucho más grandes mediante la tabla de particiones **GPT (GUID Partition Table)** y habilita funciones de seguridad como *Secure Boot*, que previene la ejecución de cargadores de arranque no firmados digitalmente.

3 Etapa 2: El Cargador de Arranque (Bootloader)

El Bootloader es el programa al que el firmware cede el control. Su única responsabilidad es cargar el núcleo (Kernel) del sistema operativo en la memoria.

3.1 GRUB 2 (GRand Unified Bootloader)

En la mayoría de las distribuciones Linux, el bootloader estándar es **GRUB 2** (Free Software Foundation, 2023). GRUB es un cargador de arranque flexible y potente.

- **En modo BIOS (Legacy):** GRUB utiliza un enfoque multietapa. El MBR contiene el "Stage 1" de GRUB, cuyo único trabajo es localizar y cargar un "Stage 1.5", que se aloja en los sectores entre el MBR y la primera partición. Este Stage 1.5 contiene los drivers necesarios para leer el sistema de archivos (ej. ext4) donde reside el "Stage 2" (los archivos principales de GRUB en `/boot/grub/`).
- **En modo UEFI:** El proceso es mucho más simple. El firmware UEFI carga directamente el archivo `grubx64.efi` (que es el Stage 2 completo) desde la partición ESP.

3.2 Configuración y Tareas

Una vez cargado, GRUB lee su archivo de configuración, el cual normalmente se encuentra en `/boot/grub/grub.cfg`. Este archivo define el menú de arranque que se presenta al usuario. Cuando se selecciona una entrada, GRUB ejecuta dos comandos esenciales:

1. `linux /boot/vmlinuz-...`: Carga el archivo del Kernel en memoria y le pasa parámetros (como `root=...` para indicar dónde está el sistema de archivos raíz).
2. `initrd /boot/initrd.img-...`: Carga el archivo del disco RAM inicial en otra ubicación de la memoria.

Tras cargar estos archivos, GRUB cede el control al Kernel.

4 Etapa 3: El Kernel de Linux

El Kernel es el primer componente "real" de Linux que se ejecuta.

4.1 Carga y Auto-descompresión

El archivo `vmlinuz` es una imagen del kernel comprimida. Lo primero que hace al recibir el control es descomprimirse a sí mismo en la memoria RAM.

4.2 Inicialización del Núcleo

Una vez descomprimido, el Kernel toma control total de la CPU y la memoria. Comienza a inicializar sus subsistemas internos, como la gestión de memoria (tablas de paginación), el planificador de tareas (scheduler) y el VFS (Virtual File System) (Bovet & Cesati, 2005). En este punto, también lee los parámetros pasados por GRUB.

5 Etapa 4: El Sistema de Archivos RAM Inicial (initramfs)

El Kernel se topa inmediatamente con un problema de tipo "huevo y la gallina":

5.1 El problema

El Kernel necesita montar el sistema de archivos raíz (indicado por el parámetro `root=...`), pero para hacerlo, necesita los módulos o drivers para el hardware de almacenamiento (como SATA, NVMe, LVM, RAID, o drivers de cifrado LUKS). Estos módulos se encuentran en `/lib/modules/...`, dentro del sistema de archivos que aún no se pueden leer.

5.2 La solución

La solución es el `initramfs` que GRUB cargó en memoria. Este archivo es un sistema de archivos temporal que vive enteramente en la RAM.

El Kernel monta este `initramfs` como su raíz temporal. Este sistema temporal contiene un conjunto mínimo de directorios y, lo más importante, los módulos del kernel necesarios para acceder al hardware de almacenamiento real.

5.3 El "Pivot Root"

El Kernel ejecuta un script `/init` dentro del `initramfs`. Este script carga los módulos necesarios (como nvme.ko, o ext4.ko para la lectura de SSDs y discos duros), prepara los volúmenes LVM o descifra los volúmenes LUKS (si es necesario), y monta el sistema de archivos raíz *real* en un directorio temporal (como /sysroot).

Una vez que el sistema real es accesible, el script ejecuta la llamada al sistema `pivot_root`. Esta operación "intercambia" la raíz temporal del `initramfs` por la raíz real montada en `/sysroot`. El `initramfs` es entonces desmontado y su memoria liberada.

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

La última acción del Kernel es ejecutar el primer programa del espacio de usuario: `/sbin/init`. Este programa se convierte en el **proceso con PID 1** y es el "ancestro" de todos los demás procesos que se ejecutarán en el sistema.

6.1 Comparativa: SysVinit vs. systemd

Históricamente, el programa `init` en Linux era **SysVinit**. Este sistema se basaba en "Runlevels" (niveles de ejecución) y ejecutaba una serie de scripts de shell de forma **secuencial** para iniciar servicios. Era simple pero lento, ya que cada servicio debía esperar a que el anterior terminara.

El estándar moderno en casi todas las distribuciones principales es **systemd** (Siever & Love, 2012). `systemd` es un gestor de sistema y servicios que opera de manera diferente.

- **Arranque paralelo:** `systemd` utiliza "unidades" (units) y analiza un árbol de dependencias. Inicia servicios en paralelo tan pronto como sus dependencias (otros servicios, puntos de montaje, sockets) están listas.
- **Activación por socket:** Puede iniciar servicios "bajo demanda" (por ejemplo el servicio SSH)

7 Etapa 6: El Espacio de Usuario y el Login

El trabajo de `systemd` es alcanzar un "objetivo" o `target`, que es un conjunto de unidades que definen el estado deseado del sistema (Nemeth et al., 2017).

7.1 `multi-user.target` (o modo Texto)

Para un servidor, el objetivo común es `multi-user.target`. `systemd` inicia todos los servicios de red, demonios de sistema y, finalmente, lanza servicios `getty` en las terminales virtuales (llamadas TTYs). `getty` es el programa que muestra el aviso de "login:" en una terminal de texto.

7.2 `graphical.target` (Modo Gráfico)

Para un sistema de escritorio, el objetivo es `graphical.target`. Este *target* depende del `multi-user.target`; hace todo lo anterior y, además, inicia el servicio del **Display Manager** (como GDM para GNOME o SDDM para KDE).

El Display Manager es el responsable de lanzar el servidor gráfico (X11 o Wayland) y mostrar la pantalla de inicio de sesión gráfica. Una vez que el usuario se autentica, el Display Manager inicia la sesión del entorno de escritorio del usuario.

El proceso de arranque es una secuencia de cesión de control:

1. **Encendido**: Se inicia el hardware.
2. **Firmware (BIOS/UEFI)**: Ejecuta el POST y carga el bootloader.
3. **Bootloader (GRUB)**: Carga el Kernel y el initramfs.
4. **Kernel + initramfs**: El Kernel se inicializa y usa el initramfs para montar el sistema de archivos real.
5. **systemd (PID 1)**: Inicia los servicios del sistema en paralelo.
6. **Login**: Se alcanza el target gráfico o de multi-usuario, presentando la pantalla de inicio de sesión.

Referencias

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., Snyder, G., Hein, T. R., Whaley, B., & Mackin, D. (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.