Actividad 1: Proceso de arranque (Material complementario)

Encender una computadora y ver el sistema operativo cargarse en cuestión de segundos parece magia, pero en realidad es el resultado de una secuencia meticulosa y precisa de pasos. Este "baile" coordinado entre el hardware y el software se conoce como el **proceso de arranque (boot process)**, y cada etapa es crucial para que tu sistema operativo pueda tomar el control y funcionar correctamente.

Vamos a desentrañar este proceso, desde el momento en que pulsas el botón de encendido hasta que ves el escritorio.

1. Los Primeros Latidos: El Firmware y el POST

Todo comienza incluso antes de que el sistema operativo entre en escena.

1.1. El Encendido y el Firmware (BIOS/UEFI): El Primer Software

Cuando presionas el botón de encendido:

- La **CPU** (el procesador) es lo primero que se activa. No tiene nada cargado en su memoria RAM, así que busca instrucciones en un lugar predefinido y no volátil: la memoria **ROM, EEPROM o Flash** de la placa madre.
- Aquí es donde reside el **firmware**. El firmware es un software de bajo nivel, muy básico, que está "incrustado" en un chip de la placa madre. Sus nombres más comunes son **BIOS (Basic Input/Output System)** y su sucesor, **UEFI (Unified Extensible Firmware Interface)**.
    - **Función del Firmware:**
        - Inicializa los componentes de hardware esenciales (CPU, RAM, tarjetas gráficas básicas, controladores de almacenamiento).
        - Prepara el sistema para cargar el sistema operativo.
        - Proporciona una interfaz básica para la configuración del hardware (la pantalla de configuración a la que accedes al inicio).

1.2. Verificación de Hardware (POST - Power-On Self Test): El Chequeo de Salud

Una de las primeras tareas del firmware es ejecutar el **POST**.

- **¿Qué es?** Es una rutina de diagnóstico que comprueba el funcionamiento básico de los componentes de hardware más críticos del sistema.
- **Componentes Verificados:** Incluye la memoria RAM, el procesador, la tarjeta gráfica, el teclado y el disco duro. Se asegura de que estos componentes estén presentes y respondan correctamente.
- **Indicadores de Error:**
    - Si el POST detecta un error grave (ej. RAM defectuosa o tarjeta gráfica ausente), el sistema no podrá arrancar.
    - El firmware puede emitir una serie de **códigos sonoros (beeps)** a través del altavoz interno de la placa madre. La secuencia de estos pitidos es un código que indica la naturaleza del error.
    - En sistemas más modernos o si el error permite alguna salida, se mostrará un mensaje de error en pantalla.

2. Encontrando el Sistema Operativo: El Orden de Arranque y los Esquemas de Particiones

Una vez que el hardware básico ha sido verificado, el firmware necesita saber dónde buscar el sistema operativo.

2.1. Selección del Dispositivo de Arranque: La Hoja de Ruta

- El firmware consulta la **CMOS RAM (Complementary Metal-Oxide-Semiconductor Random Access Memory)**. Esta es una pequeña memoria que, a diferencia de la RAM principal, mantiene su contenido incluso cuando la computadora está apagada, gracias a una pequeña batería en la placa madre.
- En la CMOS RAM se almacena la **configuración del orden de arranque (boot order)**. Esto es una lista de dispositivos (disco duro, SSD, unidad USB, unidad de DVD, red) en un orden de prioridad.
- El firmware intenta cargar el sistema operativo del primer dispositivo en la lista. Si no lo encuentra o el dispositivo no es arrancable, pasa al siguiente, y así sucesivamente.

2.2. Determinación del Esquema de Particiones: El Mapa del Disco

Una vez que el firmware ha elegido un dispositivo de arranque (normalmente el disco duro o SSD), necesita entender cómo está organizado ese disco para encontrar el código de arranque. Aquí es donde entran los **esquemas de particiones**:

- **¿Por qué es necesario?** El firmware por sí mismo no "sabe" dónde buscar en un disco. Necesita un estándar que le diga:
    - Dónde están las particiones lógicas en el disco.
    - Dónde se encuentra el código que inicia el proceso de carga del sistema operativo.
    - Cómo acceder a los datos dentro de esas particiones.
- **Los Dos Estándares Principales:**
    - **MBR (Master Boot Record) - Para BIOS Legacy:**
        - **Ubicación:** El MBR se encuentra en el **primer sector del disco (sector 0)**.
        - **Contenido:**
            - **Código de Arranque (Boot Code):** Un pequeño programa (512 bytes) que es el primer paso para cargar el sistema operativo. Su tarea principal es localizar el sector de arranque de la partición activa.
            - **Tabla de Particiones (Partition Table):** Describe hasta **cuatro particiones primarias**. Esta es una limitación clave del MBR. Si necesitas más, debes usar particiones extendidas y lógicas.
            - **Firma de Arranque (Magic Number):** Los últimos dos bytes del MBR siempre deben contener el valor `0xAA55` (o `55AAh`) para indicar que el MBR es válido y el disco es arrancable.
        - **Proceso con BIOS:** El BIOS lee el MBR del disco seleccionado. Si la firma es correcta, carga el código de arranque del MBR en la RAM y transfiere el control a este código. El código del MBR luego busca la partición activa y carga su **Boot Sector (Sector de Arranque de Volumen)**.
    - **GPT (GUID Partition Table) - Para UEFI:**
        - **Ubicación y Flexibilidad:** GPT es un estándar más moderno que no tiene las limitaciones del MBR. Almacena la tabla de particiones en múltiples lugares del disco (lo que lo hace más robusto a la corrupción) y puede

soportar un número prácticamente ilimitado de particiones (hasta 128 por defecto) y discos de más de 2TB.

- **La EFI System Partition (ESP):**
  - Es una partición especial y crucial para los sistemas UEFI.
  - Normalmente usa el sistema de archivos **FAT32** (lo que la hace ampliamente compatible).
  - Contiene los **cargadores de arranque** de uno o varios sistemas operativos (ej. `bootx64.efi` para sistemas de 64 bits) y otras aplicaciones de arranque (como utilidades de firmware).
- **Proceso con UEFI:** El firmware UEFI lee la tabla GPT para localizar la ESP. Una vez encontrada, carga el cargador de arranque (`.efi` file) desde la ESP directamente en la RAM y le transfiere el control. UEFI es más inteligente que BIOS y puede ejecutar directamente estos archivos EFI.

3. El Relevo: Del Manejador de Arranque al Núcleo del SO

Una vez que el firmware ha encontrado y cargado el primer trozo de código, el proceso avanza al **manejador de arranque**.

3.1. Cargar el Manejador de Arranque (Boot Loader): El Intermediario Experto

- **¿Qué es?** El manejador de arranque (también llamado gestor de arranque o boot loader) es un programa más sofisticado que el pequeño código del MBR.
- **Ejemplos Populares:**
  - **GRUB (GRand Unified Bootloader):** Muy común en sistemas GNU/Linux.
  - **Windows Boot Manager:** El gestor de arranque de Microsoft Windows.
- **Funciones:**
  - **Menú de Arranque:** Si tienes varios sistemas operativos instalados (dual-boot), el manejador de arranque te presenta un menú para que elijas cuál iniciar.
  - **Localización del Núcleo:** Su tarea principal es localizar el **núcleo (kernel)** del sistema operativo en el sistema de archivos del disco duro. El kernel no está en un sector fijo, sino en un archivo específico dentro del sistema de archivos.
  - **Carga en RAM:** Una vez localizado, el manejador de arranque carga el kernel en la memoria RAM y le pasa el control.

3.2. Iniciar el Núcleo del Sistema Operativo: El Despertar del Cerebro

Ahora sí, el corazón del sistema operativo comienza a latir.

- El **núcleo del sistema operativo** (el `vmlinuz` en Linux o `ntoskrnl.exe` en Windows) se descomprime (si estaba comprimido) y toma el control total del hardware.
- **Inicialización del Hardware:** El kernel comienza a inicializar el resto de los componentes de hardware, detectando dispositivos, cargando controladores (drivers) y configurándolos.
- **Proceso Específico en Linux:**
    1. **`initramfs` (Initial RAM Filesystem):** El kernel carga un pequeño sistema de archivos inicial en la RAM. Este `initramfs` contiene controladores y herramientas esenciales que el kernel necesita para montar el **sistema de archivos raíz (root filesystem)** principal (donde está el resto del SO).
    2. **Montar el Sistema de Archivos Raíz:** Una vez que el `initramfs` ha proporcionado los drivers necesarios, el kernel monta el sistema de archivos principal del disco.
    3. **Ejecutar `init` o `systemd`:** El kernel lanza el primer proceso del sistema, tradicionalmente `init` (en sistemas antiguos) o `systemd` (en la mayoría de los sistemas Linux modernos). Este proceso, con PID (ID de proceso) 1, es el "padre" de todos los demás procesos y se encarga de:
        - Iniciar todos los servicios del sistema (red, sonido, etc.).
        - Preparar el entorno para los usuarios.
- **Proceso Específico en Windows:**
    1. **`ntoskrnl.exe`:** El Windows Boot Manager carga el núcleo de Windows (`ntoskrnl.exe`).
    2. **Inicialización de Controladores y Servicios:** El núcleo inicia los controladores esenciales para el hardware vital y los servicios básicos del sistema.
    3. **Registro:** Se carga el **Registro de Windows**, una base de datos central que almacena la configuración de hardware, software y usuarios.
    4. **Entorno Gráfico:** Finalmente, se inicializa el entorno gráfico (el "escritorio" de Windows) y se carga la pantalla de inicio de sesión.

4. La Transición Final: Listo para el Usuario

Una vez que el kernel ha inicializado todo el hardware, ha cargado los servicios esenciales y el entorno de usuario, el sistema está listo.

- Verás la pantalla de inicio de sesión o, si está configurado, el sistema te llevará directamente al escritorio.
- Ahora puedes interactuar con la interfaz gráfica o la consola de comandos, lanzar aplicaciones y comenzar a trabajar.

Resumen Visual del Proceso de Arranque:

1. **Encendido:** El procesador carga el **Firmware** (BIOS/UEFI) desde un chip ROM/Flash.
2. **POST:** El Firmware ejecuta el **Power-On Self Test** para verificar el hardware básico.
3. **Orden de Arranque:** El Firmware consulta la **CMOS RAM** para el orden de los dispositivos de arranque.
4. **Esquema de Particiones:** El Firmware determina el esquema de particiones del disco (MBR o GPT).
    - **BIOS:** Busca el **MBR** (sector 0, con firma `0xAA55`).
    - **UEFI:** Busca la **EFI System Partition (ESP)** en un disco GPT.
5. **Cargar Boot Loader:** El Firmware carga el **Manejador de Arranque** (ej. GRUB, Windows Boot Manager) en la RAM y le transfiere el control.
6. **Cargar Núcleo del SO:** El Manejador de Arranque localiza y carga el **Núcleo (Kernel)** del sistema operativo en la RAM.
7. **Inicialización del SO:** El Kernel se descomprime e inicia la inicialización del hardware, carga los controladores necesarios y lanza el primer proceso del sistema (init/systemd en Linux, ntoskrnl.exe y servicios en Windows).
8. **Entorno de Usuario:** El control pasa al entorno gráfico o de consola, listo para la interacción del usuario.

Este intrincado proceso, que ocurre en milisegundos o pocos segundos, es una muestra magistral de cómo el software y el hardware colaboran para dar vida a tu computadora.