



Sistemas Operativos 2020-2021

TEMA 1

Introducción a los Sistemas Operativos

- 1.1. Definición de un sistema operativo.
- 1.2. Evolución histórica.
- 1.3. Bloques funcionales de un sistema operativo.
- 1.4. Arranque y parada del sistema.
- 1.5. Clasificación de los sistemas operativos.



Definición, funciones y objetivos de un SO

Índice

- 1.1.1. Sistema informático o sistema de computación.
- 1.1.2. Definición de sistema operativo.
- 1.1.3. Tipos de usuarios de un sistema operativo.



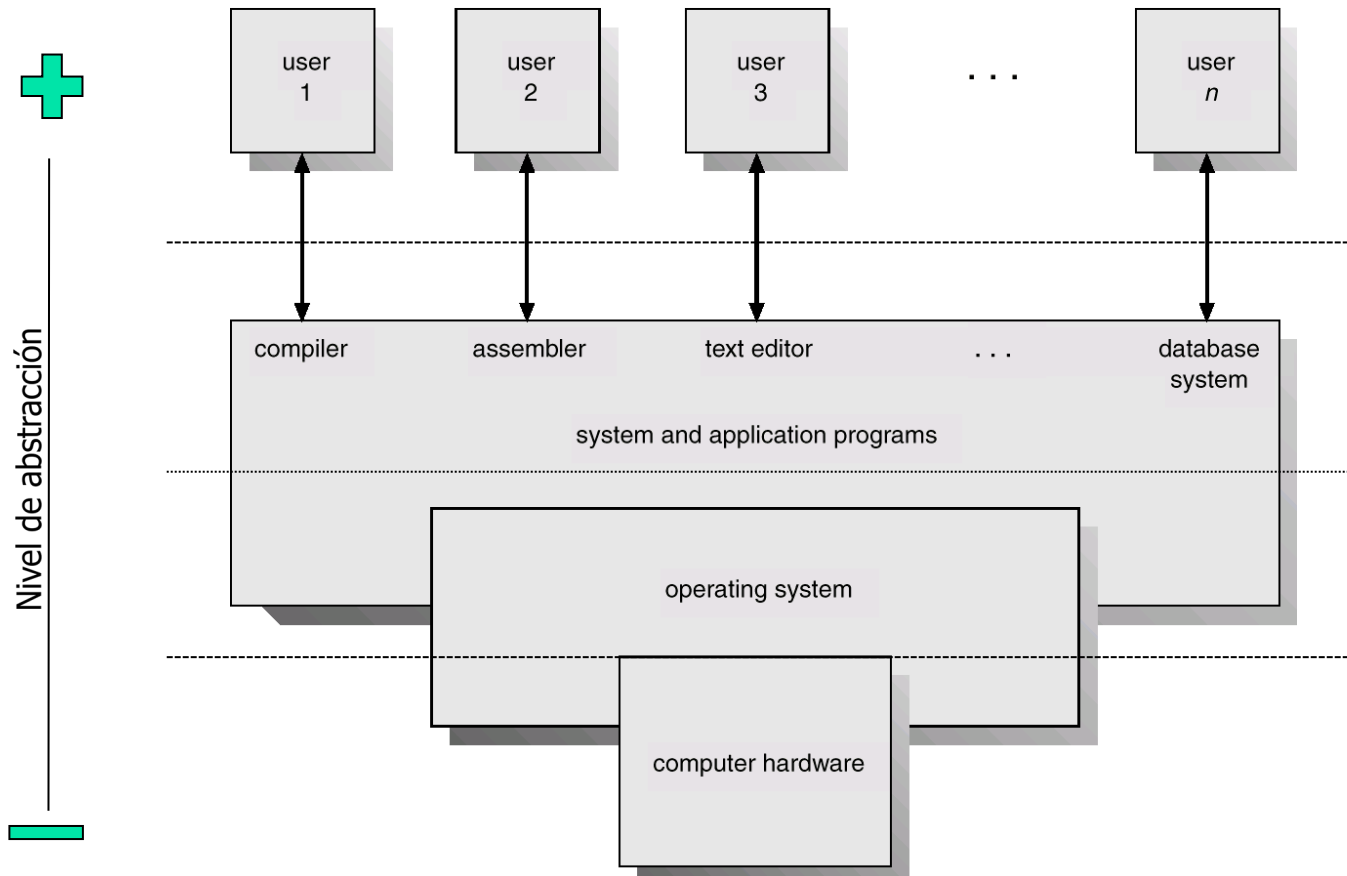
Definición, funciones y objetivos de un SO

Sistema informático o sistema de computación

- Un sistema informático consta de 3 componentes:
 1. Hardware (componentes físicos de la máquina).
 2. Software:
 - Programas de aplicación.
 - Software de sistemas.
 - Programas de sistemas.
 - **Sistema operativo.**
 3. Usuarios.
 - Personas que se identifican individualmente ante el SO.
 - Ven al sistema informático en términos de las aplicaciones que usan.
 - Sesión: conjunto de actividades entre la identificación y el fin de realización de actividades.

Definición, funciones y objetivos de un SO

Sistema informático o sistema de computación





Definición, funciones y objetivos de un SO

Sistema informático o sistema de computación

- La informática tiende a una MAYOR abstracción
- Entre más abstracto más sencillo de usar:
 - *Aplicaciones de usuario*
- Entre menos abstracto más complicado de usar:
 - *Dispositivos, código máquina*



Definición, funciones y objetivos de un SO

Definición de sistema operativo

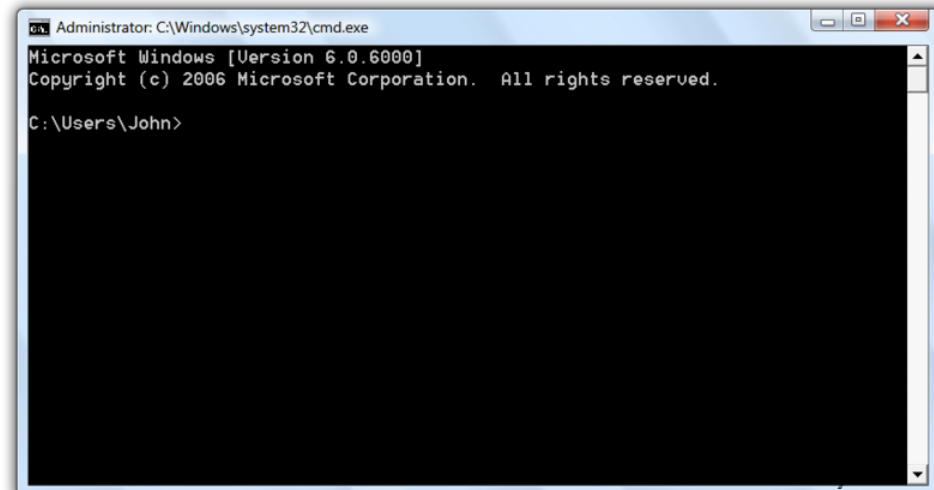
- **Programa** que **controla** la ejecución de los programas de aplicación y que actúa como **interfaz** entre el usuario del computador y el hardware del mismo. [Stallings]
- El único **programa** que se está ejecutando en todo momento en el computador (denominado **núcleo** o kernel), siendo el resto programas de aplicación. [Silberschatz]

Definición, funciones y objetivos de un SO

Tipos de usuarios de un SO

1. Usuario de nivel comandos y/o aplicaciones

- Utiliza programas de aplicación y software de sistemas.
- **Intérprete de comandos:** programa de sistemas usado como interfaz entre el usuario y el SO.
 - Gráfico – GUI (Graphical User Interface)
 - Textual – CLI (Command Line Interface)





Definición, funciones y objetivos de un SO

Tipos de usuarios de un SO

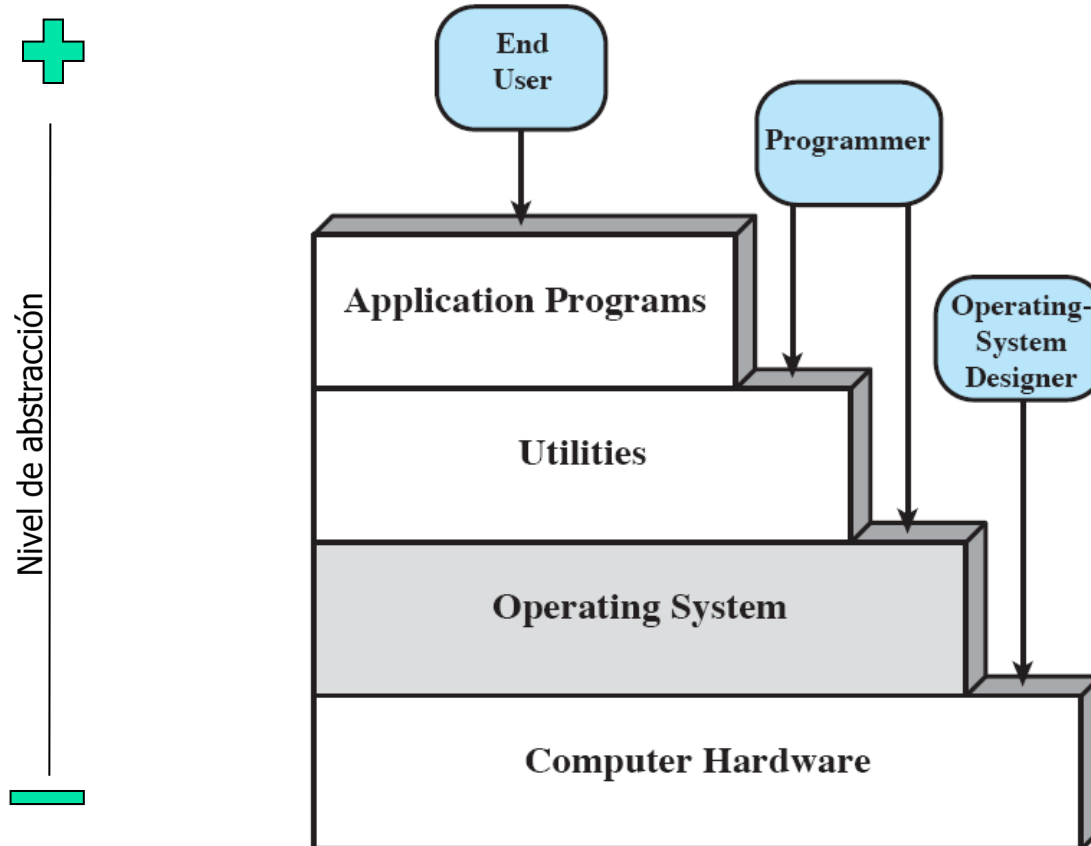
2. Usuario programador

- Utilidades (compiladores, depuradores, etc.).
- Lenguajes de programación (y sus librerías)
- **Librería de Llamadas al sistema**
 - Interfaz entre los procesos y el SO.
 - Sirven para solicitar servicios del SO.
 - Se ejecutan con máximos privilegios.

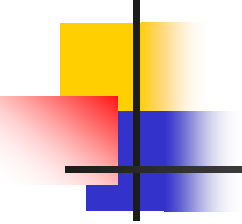
3. Diseñador/implementador del sistema operativo.

Definición, funciones y objetivos de un SO

Tipos de usuarios de un SO (resumen)



- 1.2.1. Sin sistema operativo.
- 1.2.2. Procesamiento por lotes (monoprogramación).
- 1.2.3. Sistemas multiprogramados.
- 1.2.4. Sistemas de tiempo compartido.
- 1.2.5. Sistemas operativos modernos.



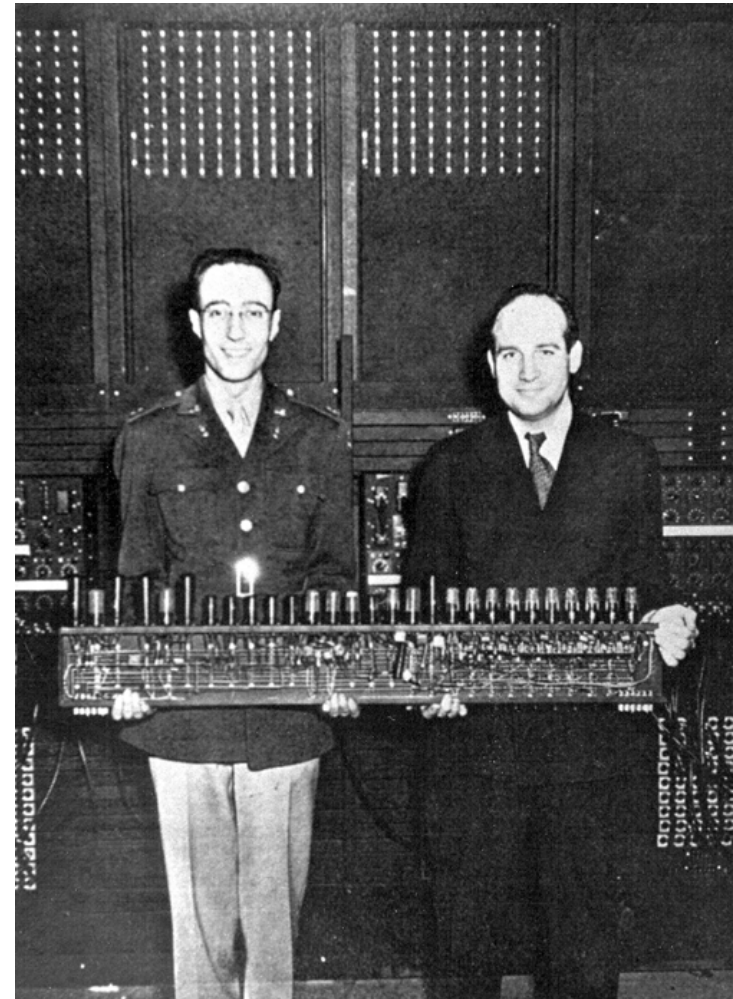
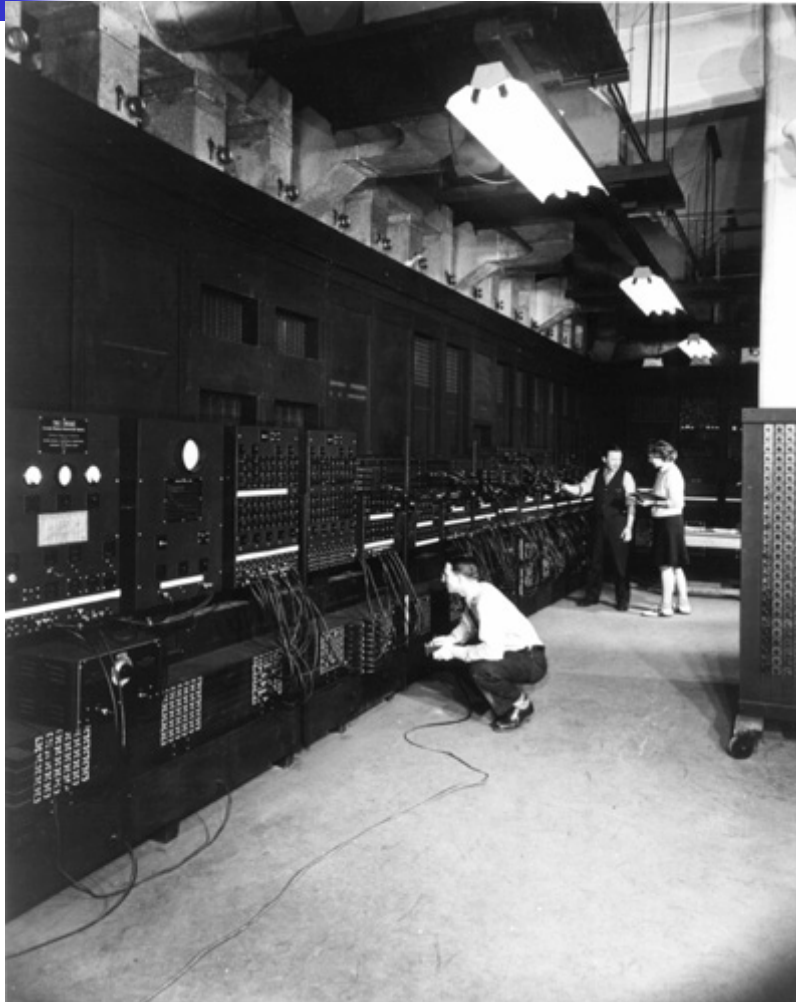
Evolución histórica

Sin sistema operativo

- **No hay sistema operativo**
 - **No hay usuarios**, solo un operador sabe manejar el ordenador.
 - **Hay un operador** que interactúa con el ordenador desde una consola:
 - Formada por unos conmutadores, indicadores luminosos, dispositivo de entrada y una impresora.
 - Programas escritos en código máquina.

Evolución histórica

Sin sistema operativo





Evolución histórica

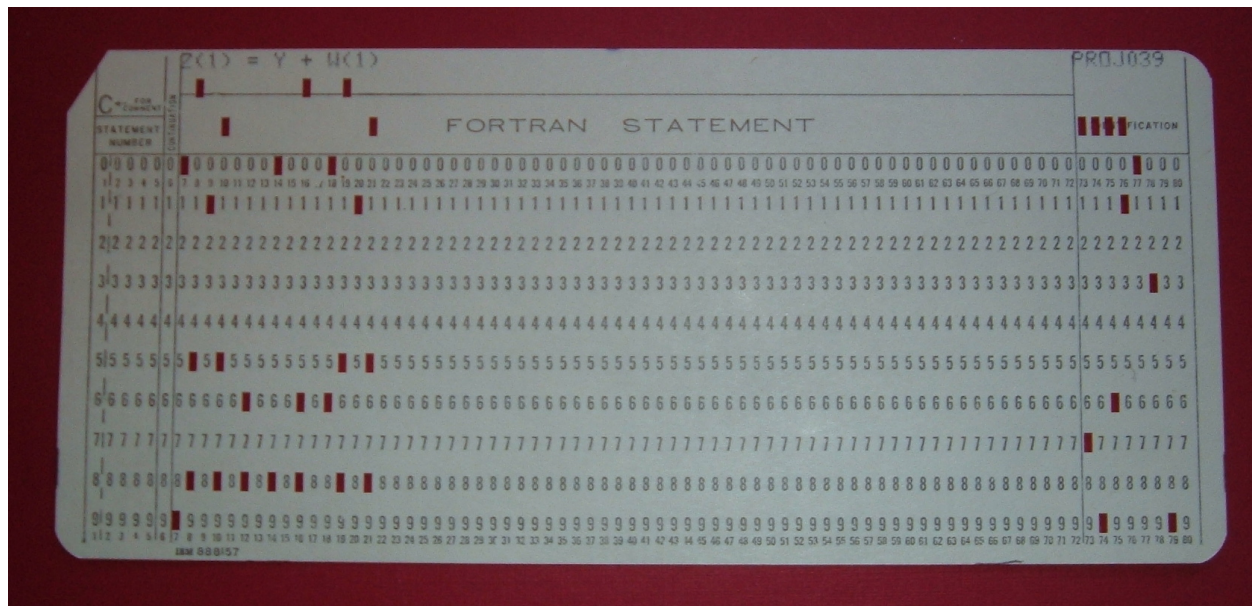
Procesamiento por lotes (*monoprogramación*)

- Secuenciación automática de programas: **monitor**
 - **Antepasado de los sistemas operativos.**
 - Gran parte de él siempre en memoria (monitor residente).
- Ahora hay un operador **y varios usuarios**
 - Los usuarios no saben manejar la máquina, solo saben programar.
 - El usuario entrega los trabajos al operador → tarjetas perforadas.
 - El operador agrupa los trabajos con requisitos semejantes en **lotes** y los coloca en un dispositivo de entrada (lector de **tarjetas perforadas**).

Evolución histórica

Procesamiento por lotes (*monoprogramación*)

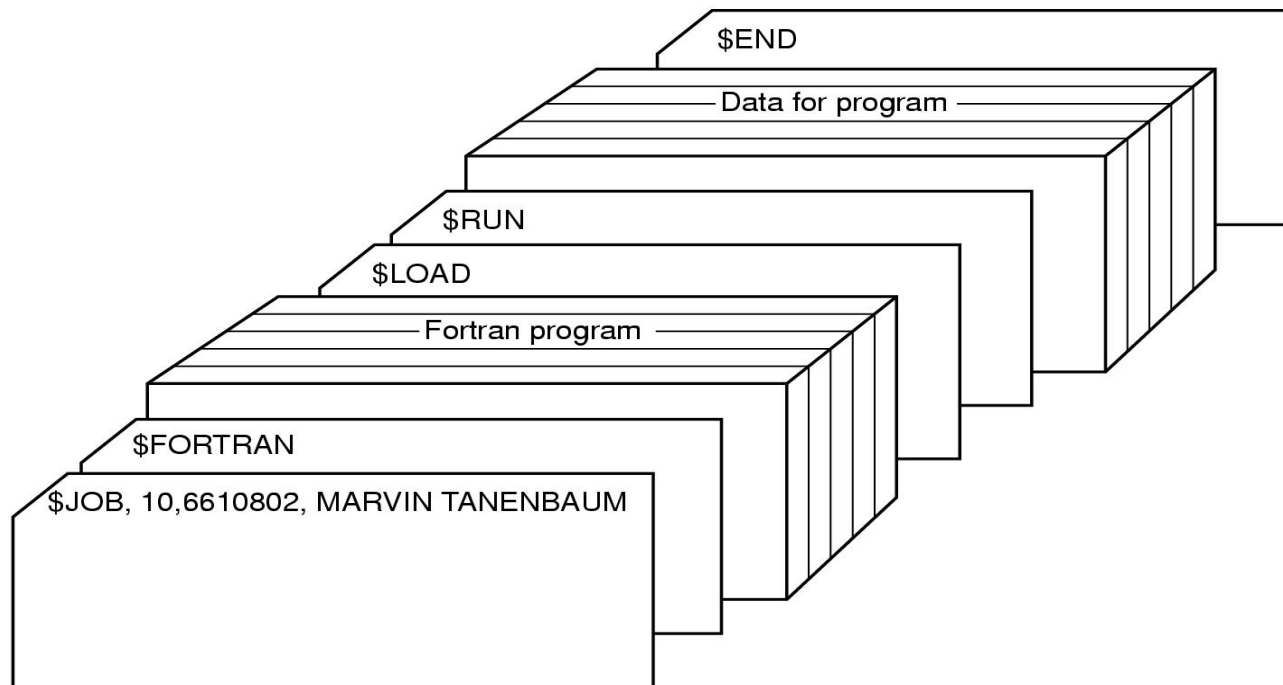
- Tarjeta perforada con un programa en fortran
 - **Ya no se escribe en código máquina.**



Evolución histórica

Procesamiento por lotes (*monoprogramación*)

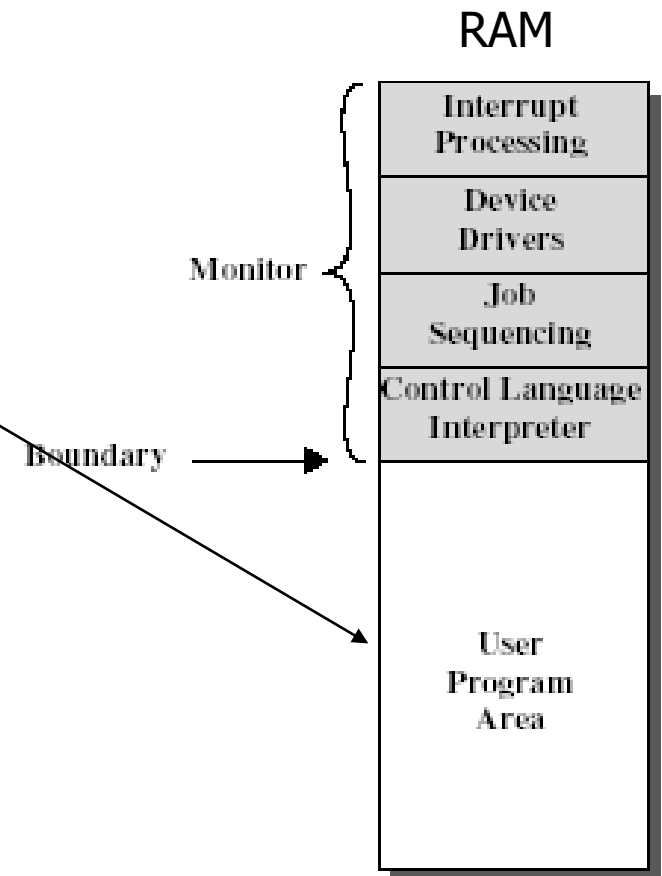
- Las tarjetas perforadas se agrupan en el lector de tarjetas



Evolución histórica

Procesamiento por lotes (*monoprogramación*)

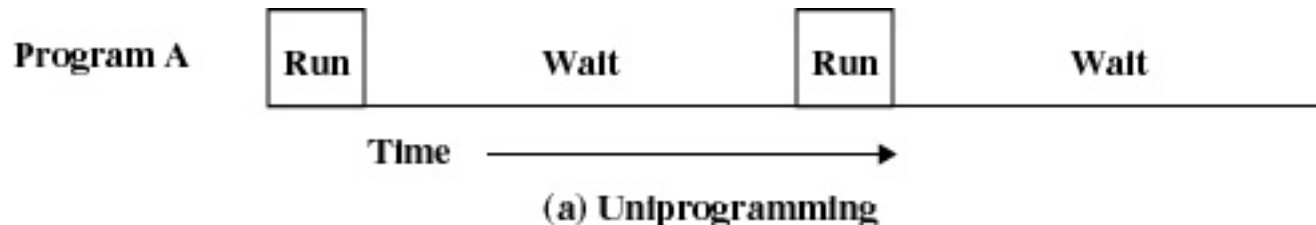
- El monitor lee los trabajos **de uno en uno (sistema monoprogramado)** del dispositivo de entrada.
 - El trabajo se coloca en la zona del programa de usuario.
 - El monitor le **cede el control** para su ejecución.
 - El trabajo devuelve el control al monitor cuando termina.
 - El monitor lee el siguiente trabajo.



Evolución histórica

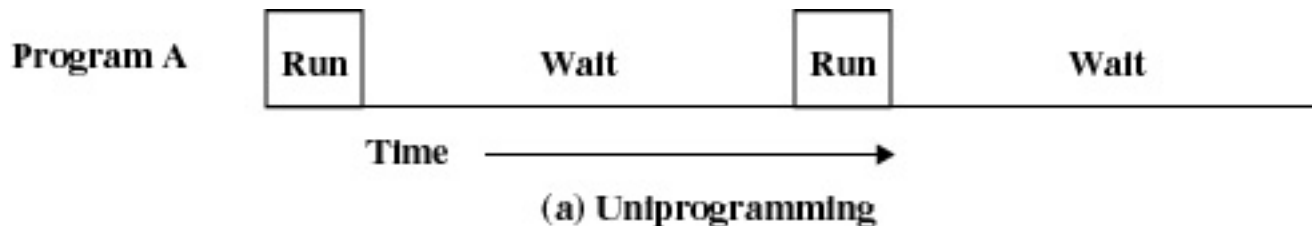
Defectos de la monoprogramación

- El lector de tarjetas perforadas (dispositivo de entrada) es MUY LENTO
- La cintas magnética (dispositivo de salida) es MUY LENTO
- **Los dispositivos son MUY LENTOS comparados con la CPU**
- Los programas no pueden continuar hasta que el dispositivo no finalice → **El programa debe esperar sin usar la CPU**



Evolución histórica Sistemas multiprogramados

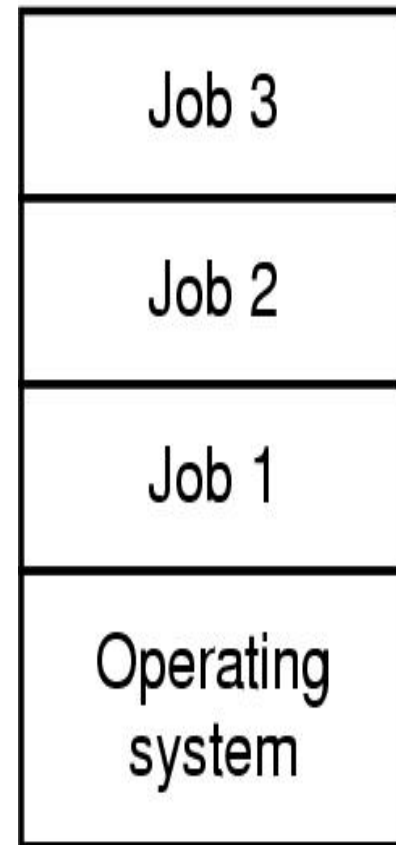
- **Aparece la técnica de multiprogramación** (todavía usada hoy en día) **y los primeros sistemas operativos.**
- En un momento dado un programa está usando la CPU o bien realizando una E/S (esperando por un dispositivo).
- Los tiempos de E/S son mucho mayores que los tiempos de CPU \Rightarrow **CPU ociosa la mayor parte del tiempo**



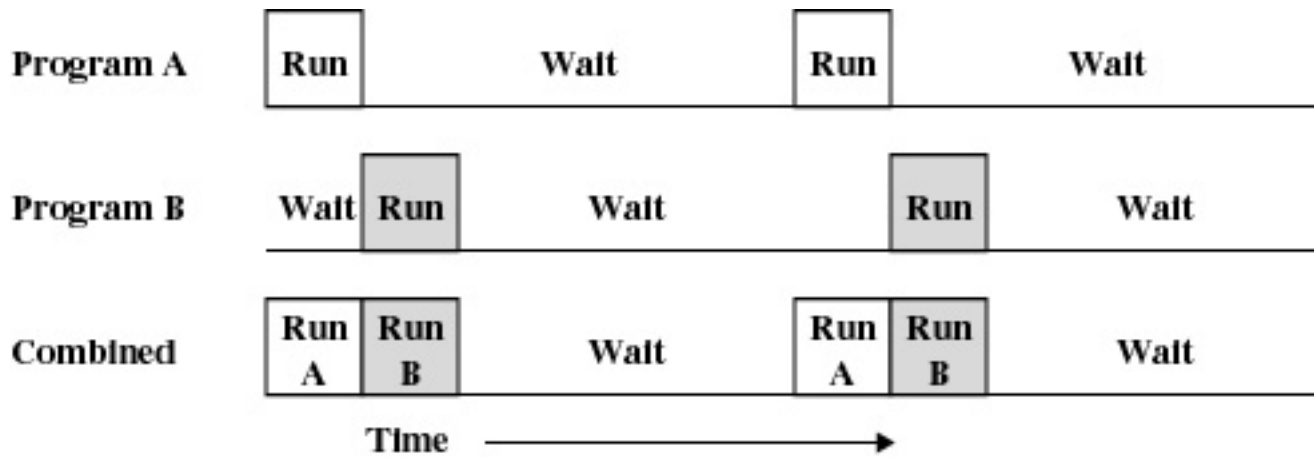


Evolución histórica Sistemas multiprogramados

- Esta ineficiencia es superable.
 - La memoria podría albergar, además del SO, varios programas de usuario.
 - Cuando un programa necesite esperar un evento, el procesador puede ejecutar otro programa que no esté esperando por un dispositivo.
 - A esta técnica se le denomina **multiprogramación** → **Reduce el tiempo de CPU inactiva**

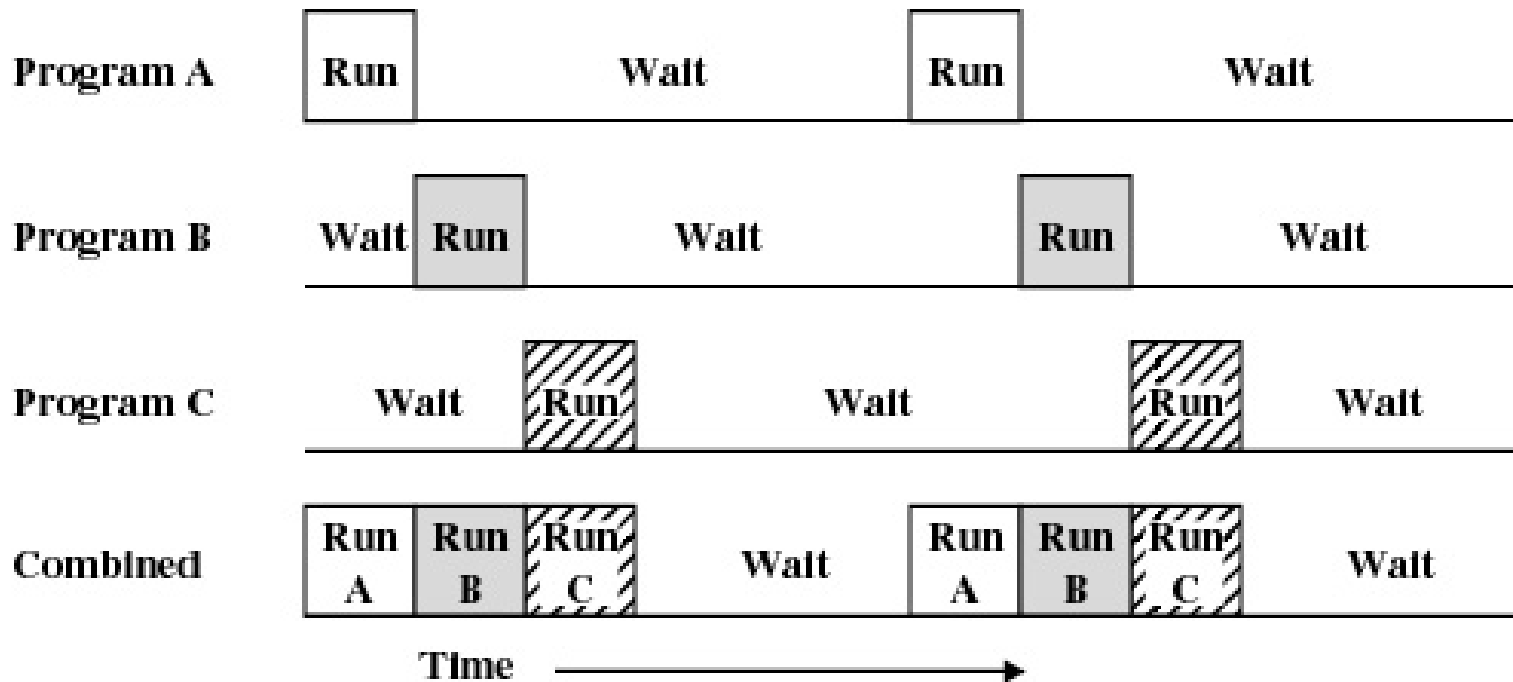


Evolución histórica Sistemas multiprogramados



(b) Multiprogramming with two programs

Evolución histórica Sistemas multiprogramados



(c) Multiprogramming with three programs



Evolución histórica Defectos de la multiprogramación

- Los procesos solo ceden el control al SO cuando acceden a un dispositivo de E/S → Un proceso puede acaparar el uso de CPU si no hace E/S
- Puede haber procesos que tarden en usar la CPU por primera vez → Tiempo de respuesta alto
- **Usuarios/as somos impacientes → Demandamos tiempo de respuesta bajo**

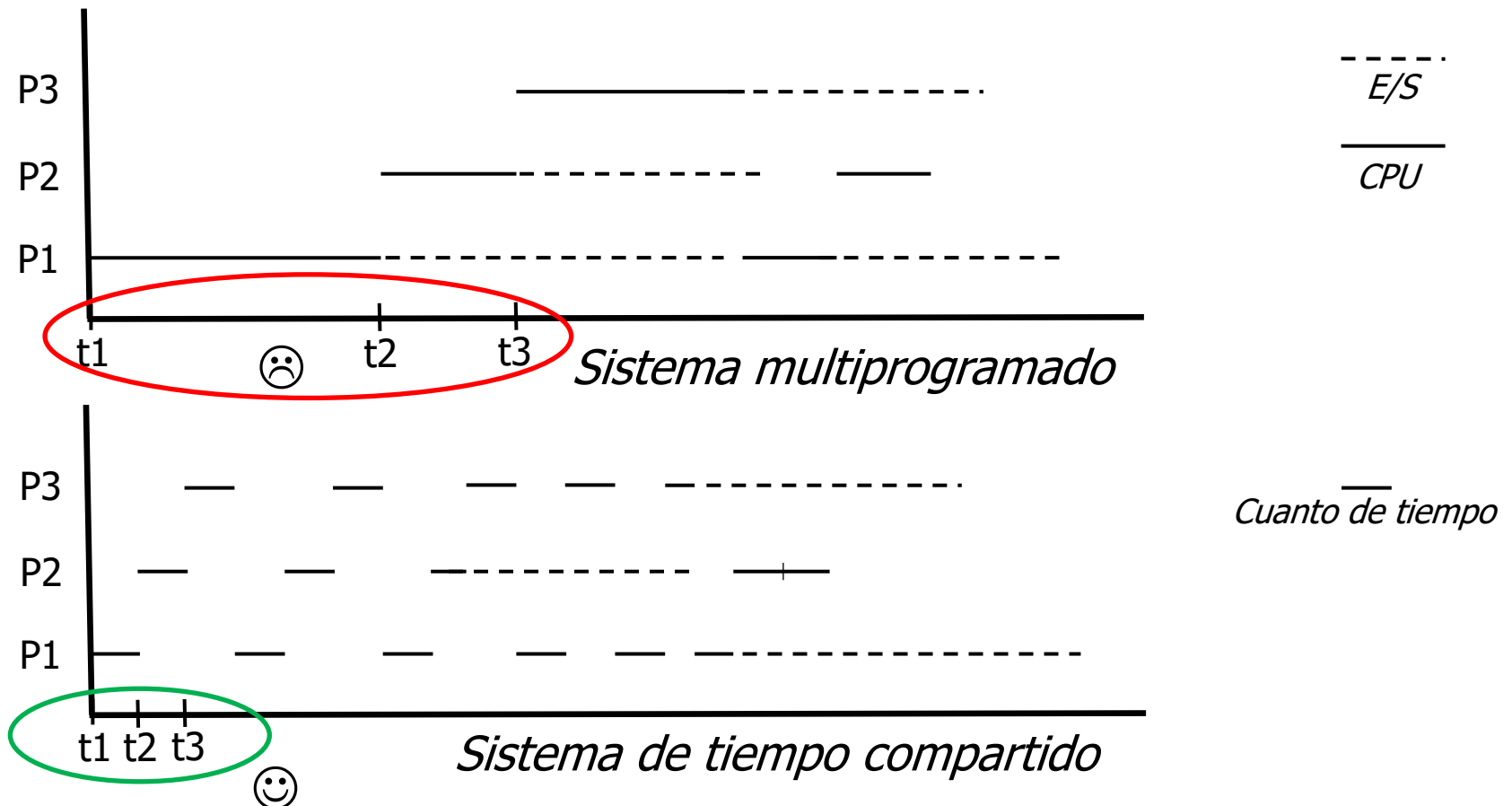


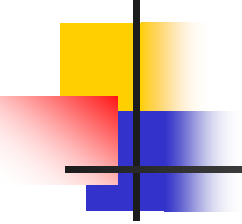
Evolución histórica Sistemas de tiempo compartido

- Cuando se tienen muchos programas corriendo en el ordenador ninguno debe monopolizar el uso de la CPU → **Tiempo de respuesta debe ser bajo**
- Se desarrolla el **tiempo compartido**.
 - El SO limita el uso continuado de CPU de cada programa a un **cuanto** (*quantum*) y asigna la CPU a otro programa → **Multiplexación en el tiempo de la CPU**
 - **Se sigue utilizando multiprogramación.**

Evolución histórica

Multiprogramación frente a Tiempo compartido





Evolución histórica Sistemas operativos modernos

- Incorporan nuevas técnicas y elementos de diseño:
 - Sistemas operativos en tiempo real → Ciertos procesos pueden tener tiempo respuesta cercano a cero.
 - Multiprocesamiento simétrico → Gestionar varias CPUs.
 - Multihilo (no son varias CPUs).
 - Sistemas operativos distribuidos → Programas se ejecutan en distintos equipos.
 - Diseño orientado a objetos → Facilita la implementación de los SOs.
- Casi omnipresentes en todo tipo de dispositivos:
 - Móviles, Tablet, Consolas de videojuegos
 - Sistemas empuetrados (automóviles, Internet de las cosas, etc).



Evolución histórica

Resumen

1. Primeros ordenadores **sin sistema operativo ni usuarios.**
2. **Antecesor de los SO: Monitor.** Ya hay usuarios programadores.
3. **Técnica de multiprogramación** → Maximiza uso de CPU
4. **Técnica de tiempo compartido** → Minimiza tiempo de respuesta de los procesos.
5. **Tiempo actual:**
 - Los SO se diseñan con metodologías avanzadas (POO, etc)
 - Soportan varias CPUs, varios hilos y varios usuarios simultáneamente
 - Integrados con LA RED
 - Aparición de dispositivos inteligentes → Necesitan un SO



Bloques funcionales de un sistema operativo

Un SO se divide, desde un **punto de vista lógico**, en un **conjunto de módulos** que cooperan entre sí y con funciones, entradas y salidas bien definidas:

■ **Administración de procesos**

- Creación, eliminación, suspensión y reanudación de procesos.
- Mecanismos para sincronización y comunicación de procesos.

■ **Administración de memoria principal**

- Registrar qué partes de la memoria están en uso y por quién.
- Asignar y liberar espacio de memoria cuando sea necesario.



Bloques funcionales de un sistema operativo

- **Administración de E/S** (no lo vemos)
 - Facilita el uso de los dispositivos periféricos.
- **Administración del almacenamiento secundario**
 - Administra el espacio libre.
 - Planifica las operaciones sobre disco.
 - Creación y borrado de ficheros.
 - Primitivas para manipulación de ficheros.
 - Ubicación de ficheros en memoria secundaria.



Bloques funcionales de un sistema operativo

- **Seguridad y protección** (en otra asignatura)
 - Garantiza la identidad de los usuarios y define las operaciones posibles para cada uno.
- **Redes** (en otra asignatura)
 - Posibilita la comunicación entre diferentes computadores.
- **Intérprete de comandos**
 - En algunos SO forma parte de su núcleo.
 - En otros es un programa especial.



Arranque y parada del sistema

Índice

- 1.4.1. Arranque hardware. El *firmware*
- 1.4.2. Cargador de arranque en BIOS
- 1.4.3. Ejecución del cargador de arranque en BIOS
- 1.4.4. Cargadores de arranque en UEFI
- 1.4.5. Ejecución con varios dispositivos
- 1.4.6. Tipos de cargadores de arranque.
- 1.4.7. Arranque del sistema operativo
- 1.4.8. Parada del sistema



Arranque y parada del sistema

Arranque hardware

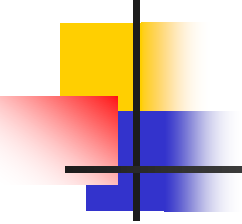
- **Paso 0:** Botón de encendido.
 - El ordenador o dispositivo inteligente sólo puede realizar actividades útiles si tiene un programa cargado en memoria principal.
 - Problema → Memoria principal (volátil) no contiene información válida.
- **Paso 1:** Ejecución del *firmware*
 - Se usa, entonces, un **software** de arranque (*firmware*) grabado en memoria no volátil.
 - Se carga en el registro PC de la CPU la dirección de comienzo del *firmware*.
 - **Está escrito y grabado por el fabricante.**



Arranque y parada del sistema

Arranque hardware

- **Paso 1:** Ejecución del *firmware* (continuación)
 - El *firmware* NO reside en memoria secundaria → No se puede modificar fácilmente.
 - El *firmware* está protegido por el fabricante → En dispositivos móviles suele estar además cifrado y controla el arranque de un SO concreto.
 - En los ordenadores personales el *firmware* (llamado comúnmente BIOS) no está protegido → Se puede instalar y arrancar cualquier SO.
 - El *firmware* se puede actualizar → Con un proceso proporcionado por el fabricante.



Arranque y parada del sistema

Arranque hardware: Tipos de *firmware*

- El firmware tipo BIOS está siendo reemplazo por los de tipo UEFI (*Unified Extensible Firmware Interface*).
- Proporciona un interfaz gráfico (frente al de texto de la BIOS).
- Necesita dispositivos con tabla de particiones tipo GPT (se verá más adelante lo que es).
- Ejecuta directamente ejecutables tipo EFI (se verá más adelante lo que es).
- Pueden iniciar el SO de forma segura (*Secure Boot*).
- Puede emular el arranque tradicional BIOS



Arranque y parada del sistema

Chequeo del hardware

- **Paso 2:** Chequeo básico del hardware del equipo
 - Este paso lo realiza el *firmware* → **No se puede impedir que se haga este paso.**



Arranque y parada del sistema

Cargador de arranque (Boot loader)

- **Paso 3:** Lectura y ejecución del “cargador de arranque” (*boot loader*)
 - Este paso lo realiza el *firmware* → **No se puede impedir que se haga este paso.**
 - Es un código especial que **normalmente** permite cargar un SO → PERO NO NECESARIAMENTE.
 - Puede ser cualquier código (por ejemplo un virus)
 - Reside SIEMPRE en memoria secundaria (USB, tarjeta SD, CD, DVD, Disco duro, tarjeta interna del móvil, etc) → Se puede reescribir fácilmente



Arranque y parada del sistema

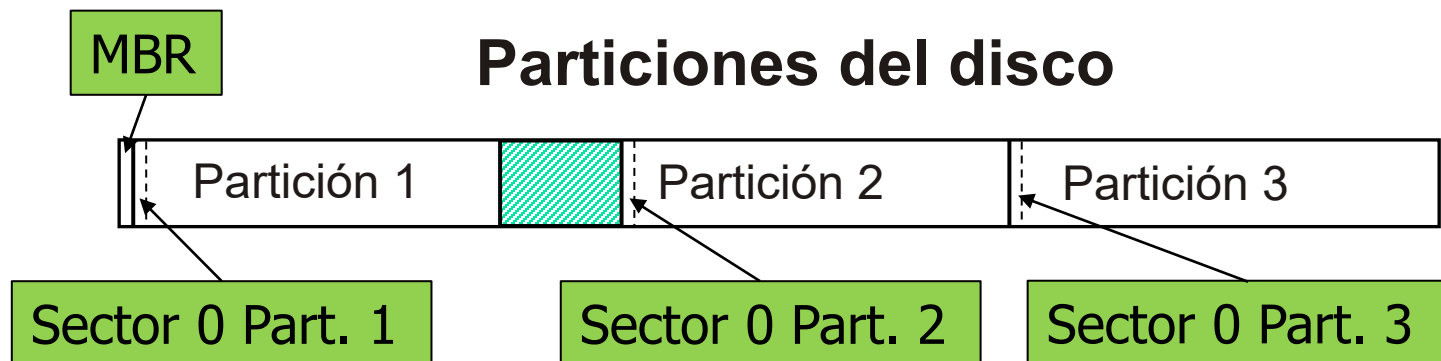
Cargador de arranque (Boot loader)

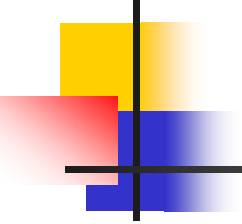
- **Paso 3A:** Lectura del “cargador de arranque” (*boot loader*)
 - **No suele estar proporcionado por el fabricante** (salvo en el supuesto del punto siguiente).
 - En el caso de dispositivos con *firmware* cerrado y protegido (móviles, consolas de juegos, etc) es un código cifrado y firmado → Si se modifica, el *firmware* detiene el arranque.
 - **El proceso de lectura de este paso es diferente dependiendo del tipo de firmware** (BIOS o UEFI)
 - Comenzaremos viendo el proceso en BIOS y luego lo veremos para UEFI

Arranque y parada del sistema

Cargador de arranque: Ubicación con BIOS

- En zonas predefinidas del disco:
 - En el PRIMER sector de TODO EL DISCO (MBR → *Master Boot Record*).
 - En el PRIMER sector de cada partición.

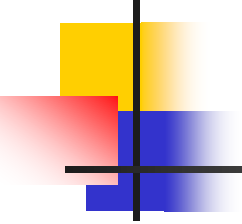




Arranque y parada del sistema

Cargador de arranque: Ubicación con BIOS

- El sistema tradicional (BIOS) usa una tabla de particiones tipo BIOS/MSDOS con las siguientes limitaciones:
 - Máximo de 4 particiones primarias ó 3 particiones primarias + 1 extendida
 - La partición extendida no se puede usar directamente y debe dividirse a su vez en unidades lógicas
 - Solo las particiones primarias pueden tener flag de arranque



Arranque y parada del sistema

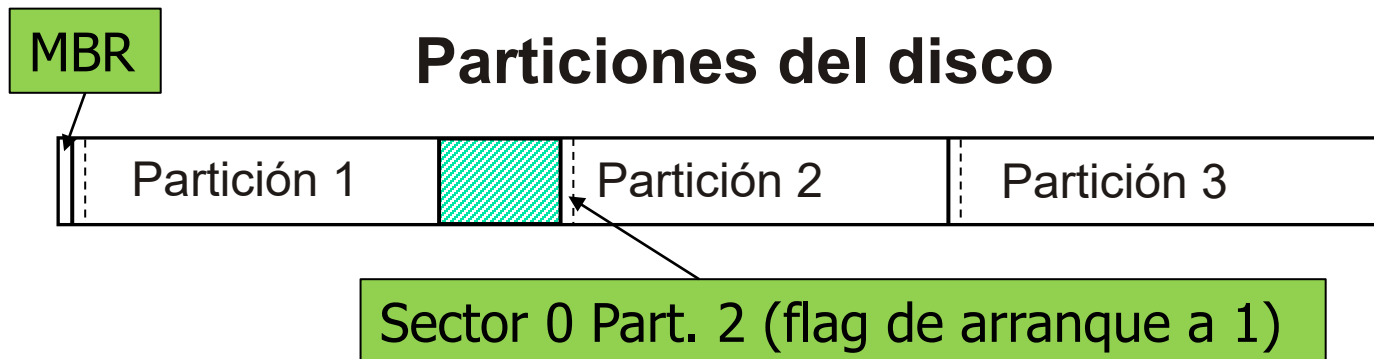
Cargador de arranque (Boot loader)

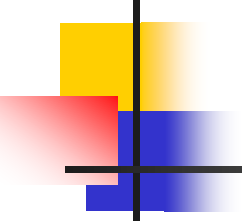
- **Paso 3A (en BIOS):** Lectura del “cargador de arranque” (*boot loader*) (continuación)
 - Intenta ejecutar todos los posibles programas cargadores que pueda haber en un dispositivo en este orden:
 1. **Siempre en primer lugar el que haya en su MBR.**
 2. **Si el MBR está vacío**, el que haya en el sector 0 de la partición primaria con el flag de arranque activo.
 - El paso 2 no se realiza si no hay particiones primarias o si no hay ningún flag de arranque activo.
 - Si no encuentra ningún cargador de arranque el ordenador da un error y se detiene.

Arranque y parada del sistema

Cargador de arranque (Boot loader)

- **Paso 3A (en BIOS) :** (continuación)
 - Primero lo intenta buscar en el MBR...
 - Luego, **solo si lo anterior falla**, en la partición primaria con el flag de arranque activo
 - **COMO MUCHO HAY DOS INTENTOS**





Arranque y parada del sistema

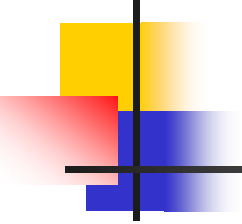
Cargador de arranque: Ubicación con UEFI

- Los programas cargadores son ficheros ejecutables tipo EFI
- Todos ellos residen en una partición especial (Partición EFI) formateada con SF tipo FAT.

Disco GPT



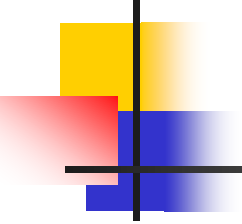
Programas cargadores



Arranque y parada del sistema

Cargador de arranque: Ubicación con UEFI

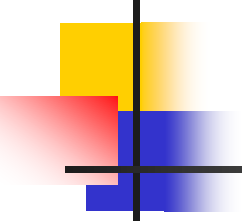
- El sistema moderno (UEFI) usa una tabla de particiones tipo GPT que NO TIENE limitaciones en:
 - Máximo número de particiones que puede tener.
 - El tamaño de cada partición puede ser mucho mayor que con el sistema tradicional
 - Todas las particiones son del mismo tipo (no hay particiones extendidas ni unidades lógicas)



Arranque y parada del sistema

Cargador de arranque (Boot loader)

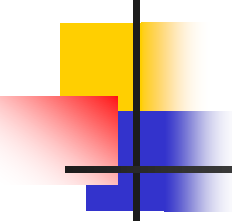
- **Paso 3A (en UEFI):** Lectura del “cargador de arranque” (*boot loader*)
 - El dispositivo de almacenamiento debe tener tabla de particiones tipo GPT
 - Debe haber una partición “especial” de tipo EFI y formateada con sistema de ficheros FAT
 - En dicha partición FAT deben almacenarse ficheros compilados de tipo EFI con el cargador de arranque
 - Lee UNO de los ficheros (UNO de los cargadores de arranque) según un orden establecido dentro del *firmware*



Arranque y parada del sistema

Cargador de arranque (Boot loader)

- **Paso 3B (en ambos):** Ejecución del “cargador de arranque” (*boot loader*)
 - Una vez cargado el “cargador de arranque” en memoria principal el *firmware* lo ejecuta y le cede el control absoluto.
 - Al igual que el código del *firmware* el “cargador de arranque” se ejecuta con máximos privilegios.



Arranque y parada del sistema

Ejecución del *boot loader* con varios dispositivos

- ¿Qué ocurre si hay más de un dispositivo **de almacenamiento**?
 - La BIOS / UEFI crea una lista ordenada con los dispositivos y realiza el proceso descrito en la transparencias anteriores con cada uno de ellos.
 - En la lista no tienen por qué estar todos los dispositivos del ordenador (incluso puede estar vacía).
 - La lista es modificable por el usuario entrando en la configuración de la BIOS / UEFI.



Arranque y parada del sistema

Tipos de cargadores de arranque

- El “cargador de arranque” suele ser una de estas dos cosas:
 1. Un programa cargador (iniciador) de un SO **concreto**.
 2. Un gestor de arranque.
- En realidad podría llegar a ser cualquier otro software (incluso un virus).



Arranque y parada del sistema

Tipos de cargadores de arranque: Gestor de arranque

- Como se dijo anteriormente el cargador de arranque puede ser un “gestor de arranque”.
- Permite iniciar cualquier SO **desde cualquier partición** (sin necesidad de que esté activa o de que sea primaria).
- Normalmente presentan un menú en pantalla para seleccionar qué SO iniciar. **En realidad de qué partición arrancar.**



Arranque y parada del sistema

Tipos de cargadores de arranque: Gestor de arranque

- Una vez seleccionado el SO del menú, el gestor de arranque **carga y ejecuta el cargador de arranque correspondiente a la partición seleccionada. Dicho segundo código (*cargador del SO*) carga y ejecuta el kernel del SO**
- En lugar de lo anterior también puede cargar y ejecutar directamente el kernel del SO elegido (nada habitual)
- **El gestor de arranque suele estar en el MBR (en el sistema tradicional)**



Arranque y parada del sistema

Arranque del sistema operativo

- Misión del *cargador del SO*
 - Traer a memoria **el kernel** del SO.
 - Una vez cargado, se realiza la **fase de iniciación**.
- Fase de iniciación:
 - Comprobación del sistema.
 - Completar las pruebas hardware.
 - Comprobar que el sistema de ficheros tiene un estado coherente.
 - Establecer estructuras de datos propias del SO.
 - Tabla de procesos, de memoria, de E/S, etc.



Arranque y parada del sistema

Arranque del sistema operativo

- Fase de iniciación (*continuación*):
 - Cargar en memoria el resto de componentes del sistema operativo que han de estar siempre en memoria.
 - Al conjunto de componentes del SO que están permanentemente en memoria se le denomina **sistema operativo residente**.
 - Crear un proceso de inicio (*login*) por cada terminal definida en el sistema.
 - Presentan un mensaje de bienvenida y esperan a que el usuario inicie la sesión.
 - Crear un conjunto de procesos auxiliares y “demonios”
 - Para impresión, comunicaciones, etc.



Arranque y parada del sistema

Parada del sistema

- El SO mantiene en memoria principal gran cantidad de información crítica.
 - Para maximizar su eficiencia.
- Es necesario un apagado ordenado para que dicha información no se pierda o corrompa.
 - Se finalizan todos los trabajos/procesos.
 - Se escribe toda la información crítica en disco.
- En el siguiente arranque tras un apagado brusco el SO comprobará si se ha corrompido información crítica.
 - La reparará, en su caso, si es posible.



Clasificación de los sistemas operativos

Índice

- 1.5.1. Según la utilización de recursos.
- 1.5.2. Según la interactividad.
- 1.5.3. Según el número de usuarios.



Clasificación de los sistemas operativos

- **Según la utilización de recursos** (sobre todo, CPU):
 - Sistemas monoprogramados.
 - Sistemas multiprogramados.
 - Sistemas de multiprocesamiento.
- **Según la interactividad** (tiempo de respuesta):
 - Sistemas de procesamiento por lotes (*batch*).
 - Sistemas de tiempo compartido.
 - Sistemas de tiempo real.
- **Según el número de usuarios** (definir “usuario”):
 - Sistemas monousuario.
 - Sistemas multiusuario.



Lecturas recomendadas

- Stallings, “Sistemas Operativos”, 5ª edición
 - Capítulo 1, “Introducción a los computadores”
 - Capítulo 2, “Introducción a los sistemas operativos”
- Tanenbaum, “Modern Operating Systems”, 2ª ed.
 - Capítulo 1, “Introduction”
- Silberschatz, “Fundamentos de Sistemas Operativos”, 7ª edición
 - Capítulo 1, “Introducción”
 - Capítulo 2, “Estructuras de sistemas operativos”
- Nutt, “Sistemas Operativos”, 3ª edición
 - Capítulo 1, “Introducción”