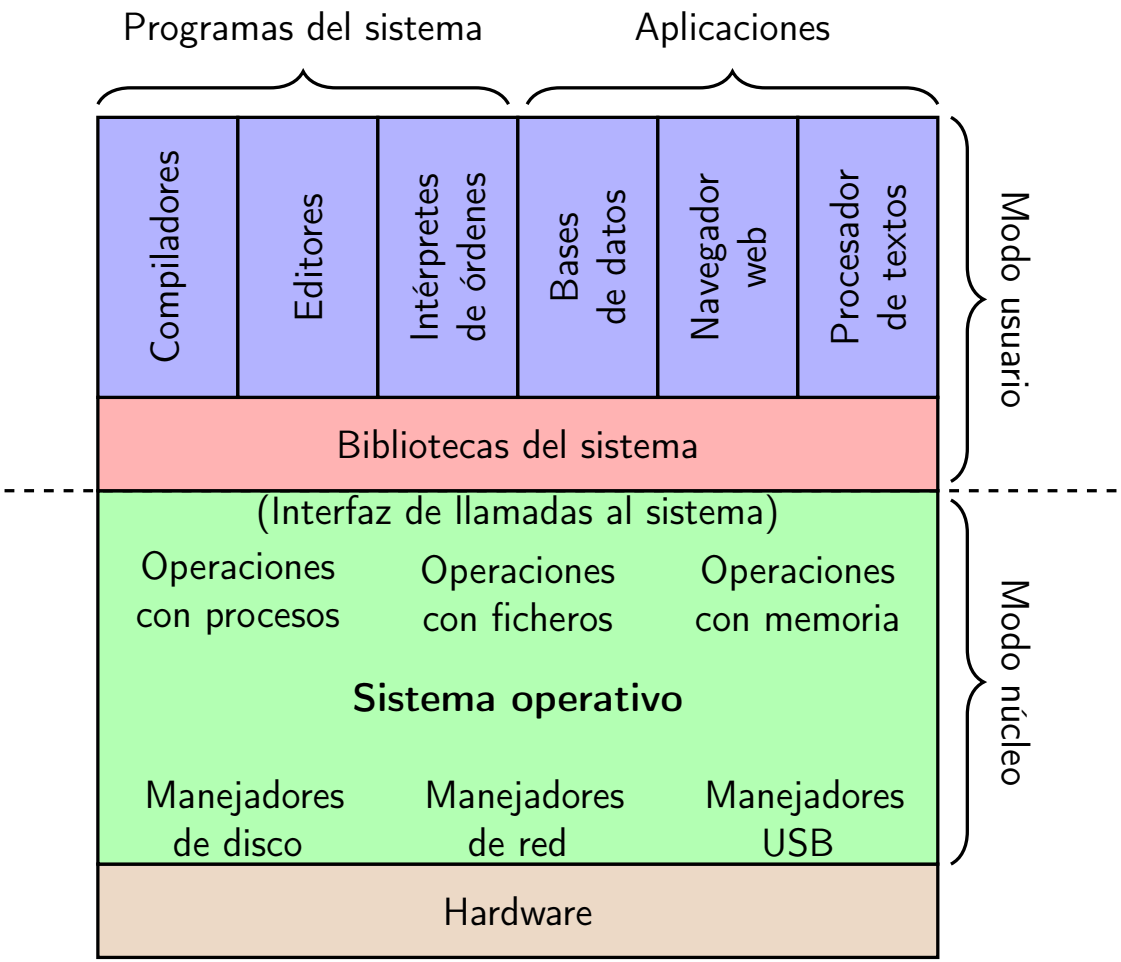
Tema 1: Introducción

Septiembre de 2023

Concepto de SO

- Intermediario entre usuarios y hardware
- Funciones complejas y diversas → No existe una única definición
- Dos puntos de vista principales:
  - Máquina extendida o virtual
    - Oculta funcionamiento del HW: interrupciones, E/S, ...
    - Presenta una interfaz de la máquina sencilla y amigable
    - Añade capacidades que el HW no posee: SSFF, procesos, ...
    - Servicios ofrecidos mediante *llamadas al sistema*
  - Controlador de recursos
    - Asigna ordenada y controladamente los recursos (CPU, memoria, impresoras, etc.) → Lleva registro del uso, atiende solicitudes, media en conflictos, etc.
- Nada mágico; programa complejo que se apoya en HW para hacer su trabajo (atender solicitudes y realizar tareas periódicas)
- Ejemplos: Linux, Microsoft Windows, Apple macOS, etc.

Posición del SO



Protección del SO

- El funcionamiento de un sistema de computación depende del correcto funcionamiento de su SO → Necesario protegerlo
- Se consigue con la ayuda de tres mecanismos hardware:
  - Modos de ejecución del procesador. Dos al menos:
    - *Modo núcleo*: el procesador ejecuta cualquier instrucción disponible. Usado por el SO
    - *Modo usuario*: la ejecución de ciertas instrucciones (E/S, etc.) produce excepciones. Usado por programas de usuario. Solicitudes al SO mediante llamadas al sistema
  - Protección de memoria: evita que un programa acceda a zonas de RAM que no le pertenecen → Se protegen SO y programas
  - Interrupciones periódicas: para que el SO tome periódicamente control y se ejecute. Evita que un proceso se apropie de CPU
- Importante: necesario usar los tres mecanismos a la vez para una protección completa

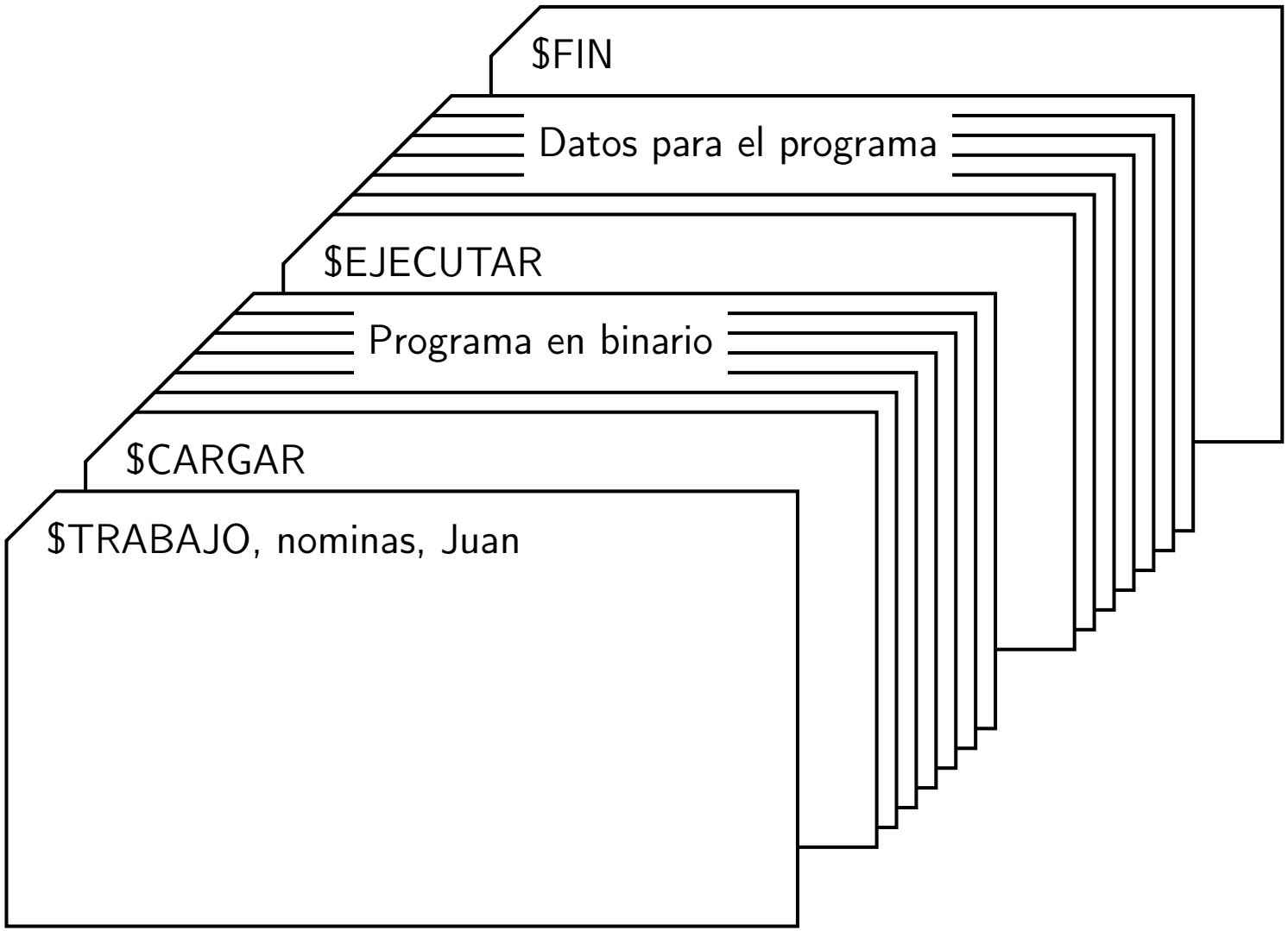
1ª generación (1945–1955)

- Máquinas enormes y pesadas, con miles de válvulas de vacío y cables conectados a mano → Gran consumo de energía y fiabilidad muy pobre
- Un solo grupo de personas diseñaba, construía, programaba, operaba y mantenía cada máquina
- Se desconocían SSOO y lenguajes de programación:
  - Programas en lenguaje máquina. Introducidos instrucción a instrucción mediante conmutadores/tarjetas perforadas
  - El programador tenía que incluir código de E/S
- Interacción directa programador-máquina: el programador reservaba máquina ciertas horas para uso exclusivo. Problemas:
  - Si terminaba pronto, la máquina estaba ociosa
  - Si no terminaba → Reservar de nuevo → Desarrollo lento

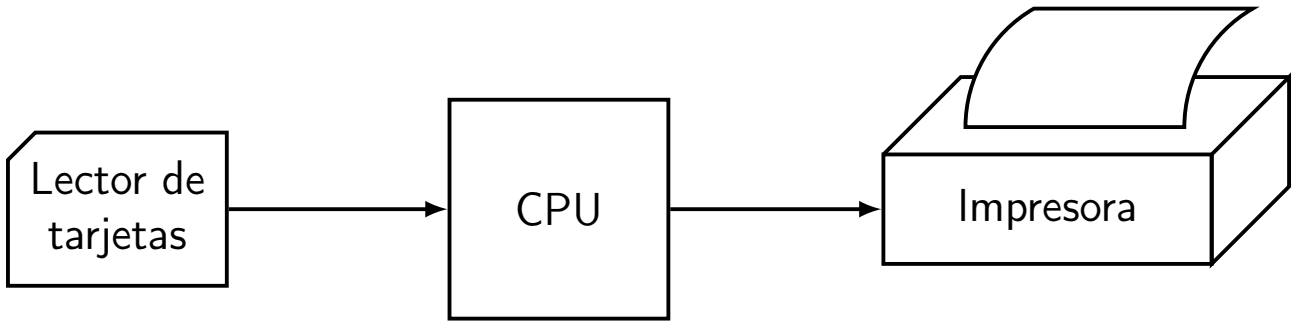
2ª generación (1955–1965)

- Máquinas construidas con transistores → Fiables → Se podían vender → Separación entre diseñadores, constructores, operadores, programadores y personal de mantenimiento
- Diversos avances hardware (lectores de tarjetas, impresoras de líneas, cintas magnéticas) y software (cargadores, enlazadores, ensambladores) que facilitan uso y programación de las máquinas
- Aparecen bibliotecas de E/S → No código de E/S en programas
- Cambia forma de usar máquina: se pierde interacción directa → Programador entrega trabajos y luego recoge resultados
- Máquina gestionada independientemente:
  - Primero, por operador profesional (SO humano) → Ineficiente
  - Después, por un *monitor residente*: SO rudimentario para el procesamiento por lotes; interpretaba tarjetas de control para procesar trabajos, cargaba programas, controlaba E/S, etc.

Estructura de un trabajo por lotes



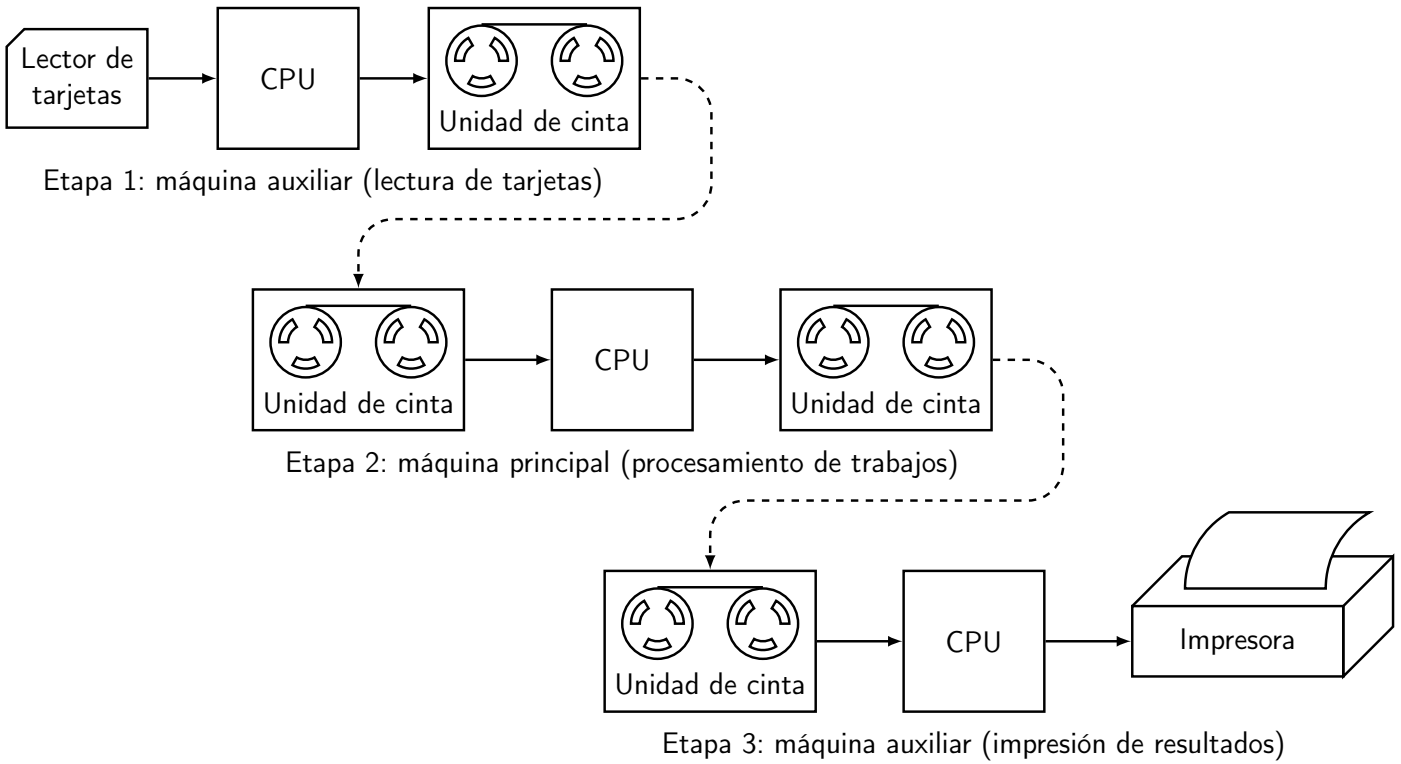
Operación en línea



- Problema: CPU ociosa muchas veces esperando terminación de E/S en dispositivos lentos (lector de tarjetas, impresora, . . .)
- Solución: operación fuera de línea

Operación fuera de línea

- CPU interactúa con dispositivos rápidos (cintas)
- Entrada: tarjetas → cintas. Salida: cintas → impresora
- Cara: necesarios varios dispositivos para entrada y salida
- Lenta para el programador: las cintas se tienen que llenar



Independencia de dispositivo

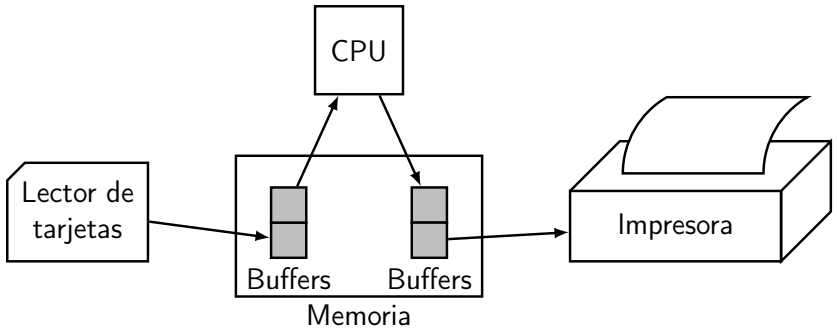
- En la operación fuera de línea, un programa pasa de leer de tarjetas a leer de cintas; también de escribir en una impresora a escribir en cintas $\rightarrow$ El programa debe poder funcionar independientemente de dónde lea o escriba
- Solución: creación de dispositivos lógicos de E/S por parte del SO
- Ejemplo: almacén de tarjetas. El SO hace corresponde este dispositivo lógico con uno real (lector de tarjetas, cinta, disco, etc.)

Buffers y spoolers

- Con la operación fuera de línea la E/S y la CPU funcionan en paralelo, pero a un alto precio (necesarias varias máquinas). ¿Cómo conseguir algo parecido en una sola máquina?
- Dos posibles soluciones:
  - Buffers
  - Spoolers

Buffers

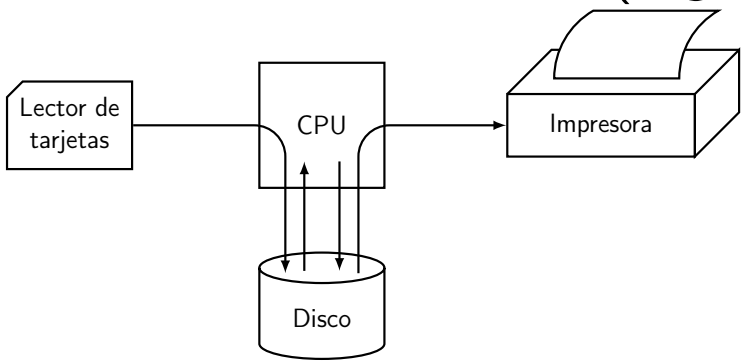
- Los datos no pasan directamente del dispositivo de E/S a la CPU o viceversa, sino que se dejan en o se leen de una memoria temporal o *buffer*:



- Ejemplo para la entrada: el dispositivo deja los datos en un buffer de memoria principal y la CPU los lee de ahí. Mientras la CPU los usa, el dispositivo inicia la siguiente lectura, dejando los nuevos datos en otro buffer
- *Se solapa la E/S de un programa con su propio cómputo* → CPU y dispositivos ocupados a la vez; solape perfecto si dispositivos y CPU igual de rápidos en cuanto a velocidad de procesamiento

Spoolers

- Necesario dispositivo de acceso aleatorio (e.g., disco)



- Ejemplo de funcionamiento para la entrada (lector de tarjetas):
  1. CPU (SO) solicita a dispositivo siguiente tarjeta
  2. CPU continúa ejecución del programa en memoria
  3. Dispositivo termina e interrumpe CPU
  4. CPU (SO) pasa datos de tarjeta a disco y vuelve al paso 1
- *Se solapa la E/S de unos programas con el cómputo de otro*
- La E/S de los programas se hace con el disco
- Especialmente útil en multiprogramación y tiempo compartido

3ª generación (1965–1980)

- Circuitos integrados → Ordenadores más pequeños y fiables
- Dos conceptos importantes de esta generación:
  - *Multiprogramación*:
    - Se tienen varios trabajos en memoria; cuando el que usa la CPU no puede continuar su ejecución porque tiene que esperar (p.e., por E/S), se pasa la CPU a otro que esté listo
    - Se superpone E/S de unos programas con el cómputo de otros
    - Necesarias protección y planificación
  - *Tiempo compartido* o *multitarea*:
    - Variante de la multiprogramación
    - Se realiza un rápido cambio entre tareas → Se crea la sensación de que varias tareas avanzan a la vez → Cada usuario pueda interactuar directamente con la máquina
    - Los usuarios recuperan la interactividad a un coste razonable
- Surgen importantes SSOO: OS/360 para el IBM 360 y MULTICS (que dio paso a Unix)

4ª generación (1980–1995)

- Circuitos LSI y VLSI → Millones de transistores en un solo chip → Ordenadores pequeños y baratos → Se populariza la informática
- Aparecen SSOO para ordenadores personales (MS-DOS) y estaciones de trabajo (Unix). Posteriormente aparecen las distintas versiones de Windows, Linux, etc., que popularizan conceptos como la multitarea, la memoria virtual, etc.
- También aparecen nuevos tipos de SSOO: de red, distribuidos, de tiempo real, etc.

5ª generación (1995–actualidad)

- Caracterizada por la explosión de Internet y de los dispositivos móviles, especialmente los teléfonos inteligentes
- Aparecen Windows 2000, Windows XP y los sucesores de ambos (en la actualidad, Windows Server 2022 y Windows 11, respectivamente)
- También se popularizan Linux, Mac OS X (macOS actualmente), Android e iOS

Tipos de sistemas operativos

- Gran variedad de sistemas de cómputo, desde supercomputadores hasta ordenadores más pequeños que una moneda
- En cada sistema se pueden usar uno o más sistemas operativos
- También, un mismo sistema operativo en diferentes tipos de computadores

Sistemas operativos de propósito general

- Diseñados para realizar gran variedad de tareas en entornos diversos
- Flexibles, capaces de adaptarse tanto al entorno hardware como a los trabajos a procesar
- Ejemplos: Windows, Linux, macOS
- Dependiendo del sistema, encontraremos unos u otros
  - Supercomputadores y mainframes: Linux
  - Servidores: cualquiera (predominantes Windows y Linux)
  - Ordenadores personales: cualquiera (predominante Windows)
  - Teléfonos inteligentes, tabletas y otros sistemas integrados: cualquiera (predominantes Android (Linux) e iOS (Mac))

Sistemas operativos de red

- Todos los de propósito general
- Usuarios conscientes de la existencia de varios ordenadores conectados mediante una red
- Cada máquina es fundamentalmente independiente de las demás
- Cuando lo necesita, cada máquina interactúa con el resto para compartir información, recursos, etc.

Sistemas operativos distribuidos

- Al igual que los anteriores, conjunto de computadores conectados entre sí mediante una red
- Sin embargo, vistos por los usuarios como un sistema tradicional, con un único computador y un único sistema operativo (imagen única del sistema)
- El usuario no es consciente del lugar donde se ejecutan sus programas ni en dónde se encuentran sus ficheros (*transparencia de localización*)
- Razones para la existencia de estos sistemas:
  - Compartición de recursos (hardware o software)
  - Aceleración de cálculos
  - Confiabilidad o tolerancia a fallos
- Ejemplos: Amoeba, Plan 9 e Inferno

Sistemas operativos de tiempo real

- Parámetro clave: restricciones temporales
- Dos tipos:
  - Rigurosos
    - Una acción se debe realizar necesariamente en cierto momento o intervalo
    - Si no se cumple la restricción $\rightarrow$ Situación crítica
  - No rigurosos
    - Es aceptable no cumplir de vez en cuando un plazo, siempre y cuando se ajuste a unos parámetros como un determinado porcentaje de fallos
- Ejemplos: VxWorks y QNX

Sistemas operativos para tarjetas inteligentes

- Dispositivos muy pequeños, con grandes limitaciones de potencia y memoria
- Sistemas operativos solo capaces de realizar unas pocas funciones
- Habitual disponer de una JVM que ejecuta los *applets* que se cargan en la tarjeta
- En algunos casos, es posible cargar y ejecutar varios *applets* a la vez $\rightarrow$ Necesarios mecanismos de multiprogramación y planificación

Componentes y servicios de los sistemas operativos

- Un sistema operativo proporciona un entorno dentro del cual se ejecutan los programas
- Construimos este entorno dividiendo lógicamente el sistema operativo en pequeños módulos o componentes
- Los componentes ofrecen servicios a los programas a través de llamadas al sistema
- Los usuarios usan esos programas para realizar diversas tareas en el sistema
- Por lo tanto, vamos a hablar de diversos aspectos (procesos, ficheros, etc.) a través de tres puntos de vista:
  - Funcionamiento interno del sistema operativo (componentes)
  - Servicios ofrecidos por el sistema operativo (llamadas al sistema)
  - Usuario y programas (programas del sistema)

Componentes del sistema

- Solo podemos crear un sistema tan grande y complejo como un sistema operativo dividiéndolo en fragmentos más pequeños
- Cada fragmento debe estar bien definido en cuanto a su función, entradas y salidas
- Componentes principales en un sistema operativo moderno:
  - Administración de procesos
  - Administración de la memoria principal
  - Administración de la E/S
  - Administración de ficheros
  - Sistema de protección
- Componentes interrelacionados: difícil hacerse una idea completa de cada uno de ellos sin conocer los demás
- Veremos cada uno en su correspondiente tema

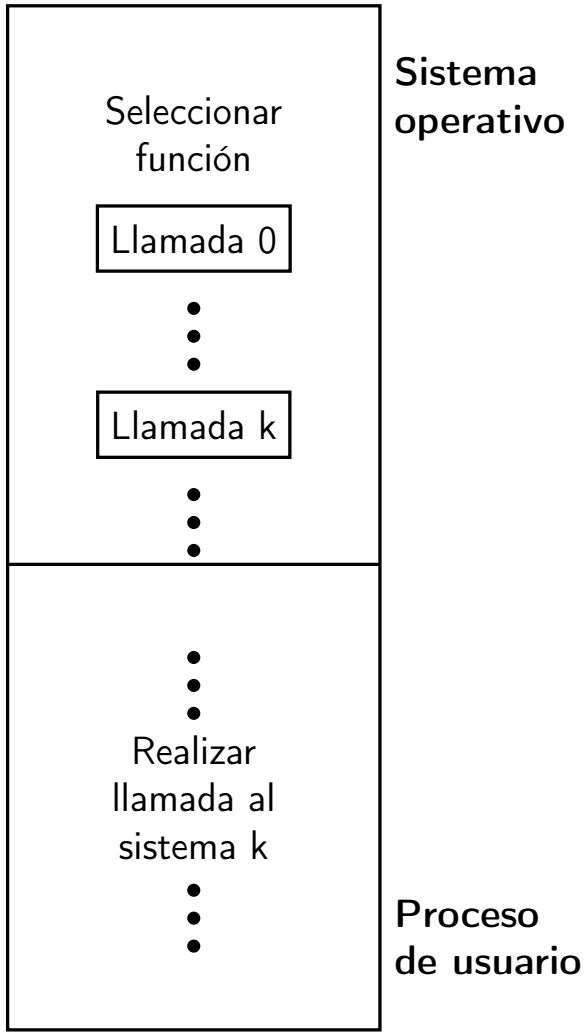
Servicios del sistema operativo

- Proporcionan un entorno amigable para los programas y usuarios
- Los más comunes:
  - Ejecución de programas
  - Realización de operaciones de E/S sobre ciertos dispositivos
  - Manipulación de sistemas de ficheros
  - Comunicación entre procesos
  - Detección de errores que se puedan producir al utilizar los diferentes servicios
- Otros para asegurar un funcionamiento eficiente:
  - Control de la asignación de recursos
  - Contabilidad del uso de los recursos
  - Protección para que cada usuario controle la información que le pertenece

Llamadas al sistema

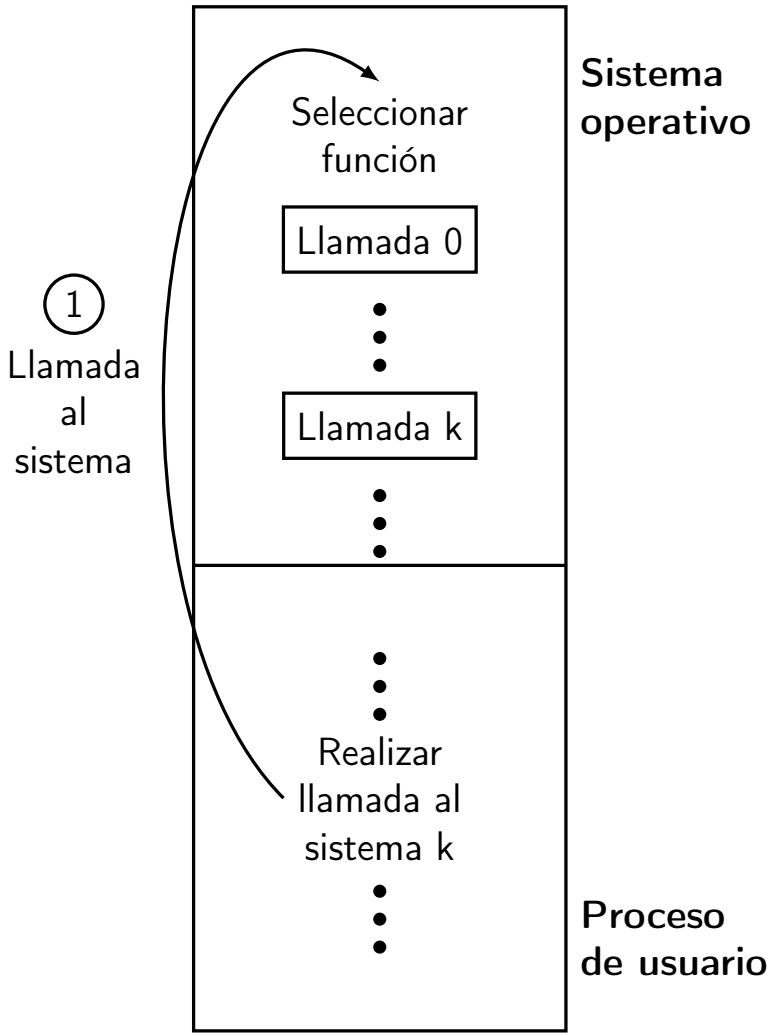
- Definen la interfaz entre el sistema operativo y un proceso
- A través de ellas, los procesos obtienen los servicios ofrecidos por el sistema operativo
- Generalmente, instrucciones en ensamblador, aunque se suelen ocultar en los procedimientos de las bibliotecas de los lenguajes de programación de alto nivel
  - Estos procedimientos son una interfaz más sencilla que la proporcionada por las llamadas al sistema al ocultar la mayor parte de los detalles de la interfaz con el SO
  - Ejemplo: `printf` $\rightarrow$ Da formato a la cadena, realiza llamadas al sistema para mostrar la cadena, comprueba errores, etc.

Funcionamiento de una llamada al sistema



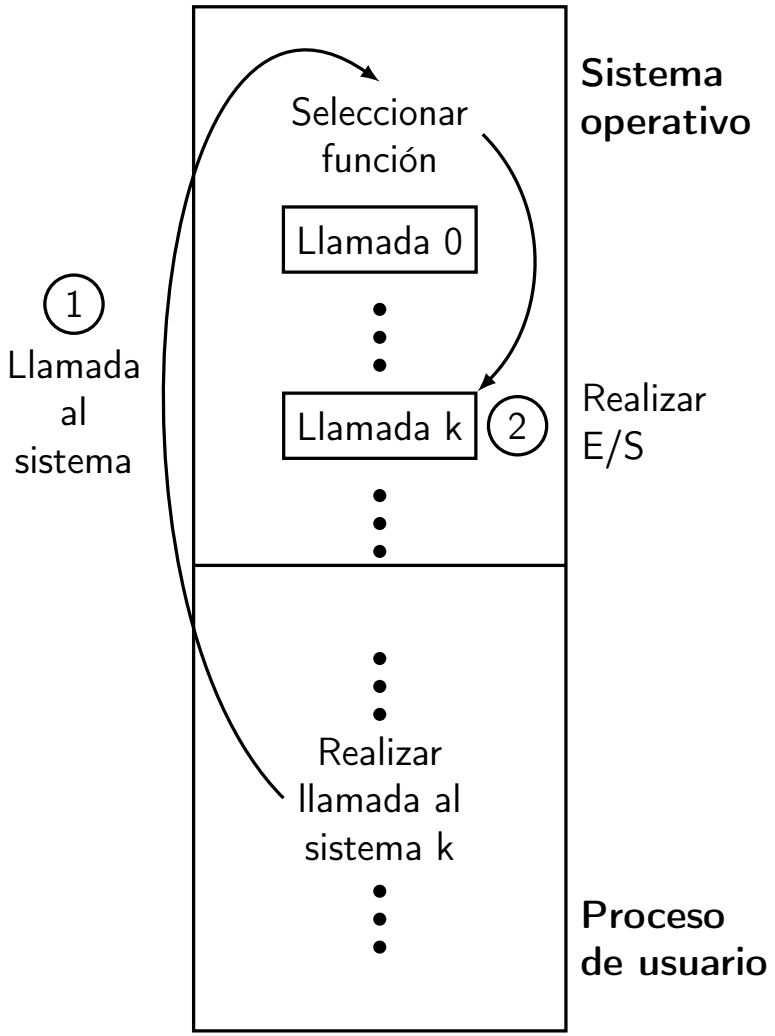
Funcionamiento de una llamada al sistema

1. Se inicia la llamada al sistema. La ejecución pasa de modo usuario a modo núcleo. El SO toma el control y examina los parámetros de la llamada para determinar cuál es



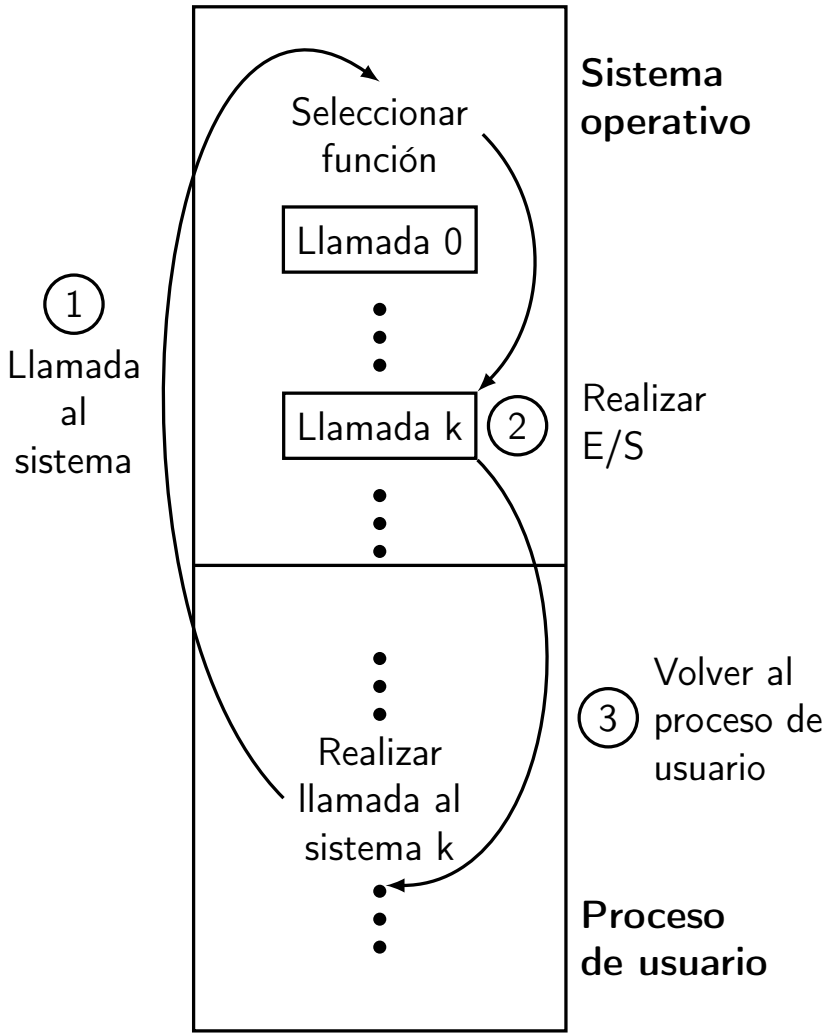
Funcionamiento de una llamada al sistema

1. Se inicia la llamada al sistema. La ejecución pasa de modo usuario a modo núcleo. El SO toma el control y examina los parámetros de la llamada para determinar cuál es
2. El SO toma de la entrada k de una tabla la dirección del procedimiento que realiza la k-ésima llamada al sistema y lo llama



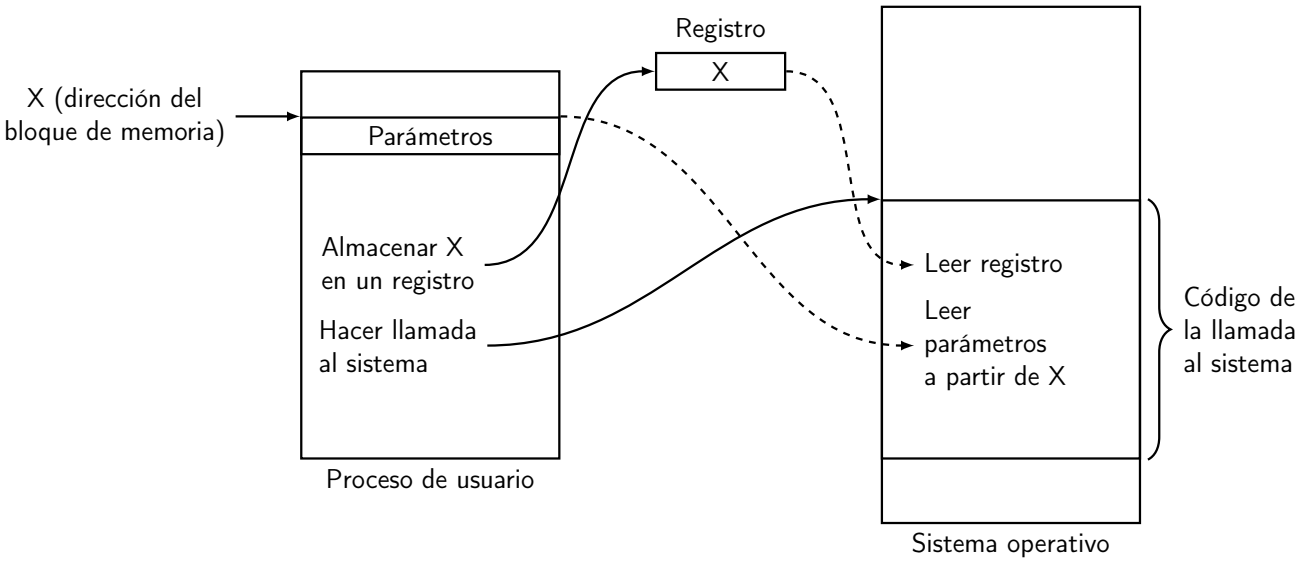
Funcionamiento de una llamada al sistema

1. Se inicia la llamada al sistema. La ejecución pasa de modo usuario a modo núcleo. El SO toma el control y examina los parámetros de la llamada para determinar cuál es
2. El SO toma de la entrada k de una tabla la dirección del procedimiento que realiza la k-ésima llamada al sistema y lo llama
3. La llamada al sistema termina y el control regresa al programa de usuario en modo usuario



Parámetros de una llamada al sistema

- No existe una llamada al sistema (instrucción en ensamblador) para cada tipo de solicitud: el mecanismo es único y, mediante parámetros, se indica servicio a realizar y datos necesarios
- Tres métodos para pasar parámetros al sistema operativo:
  - Registros
  - Pila (se apilan antes de la llamada y el SO los extrae de ahí)
  - Bloque de memoria (dirección en registro; ver figura)

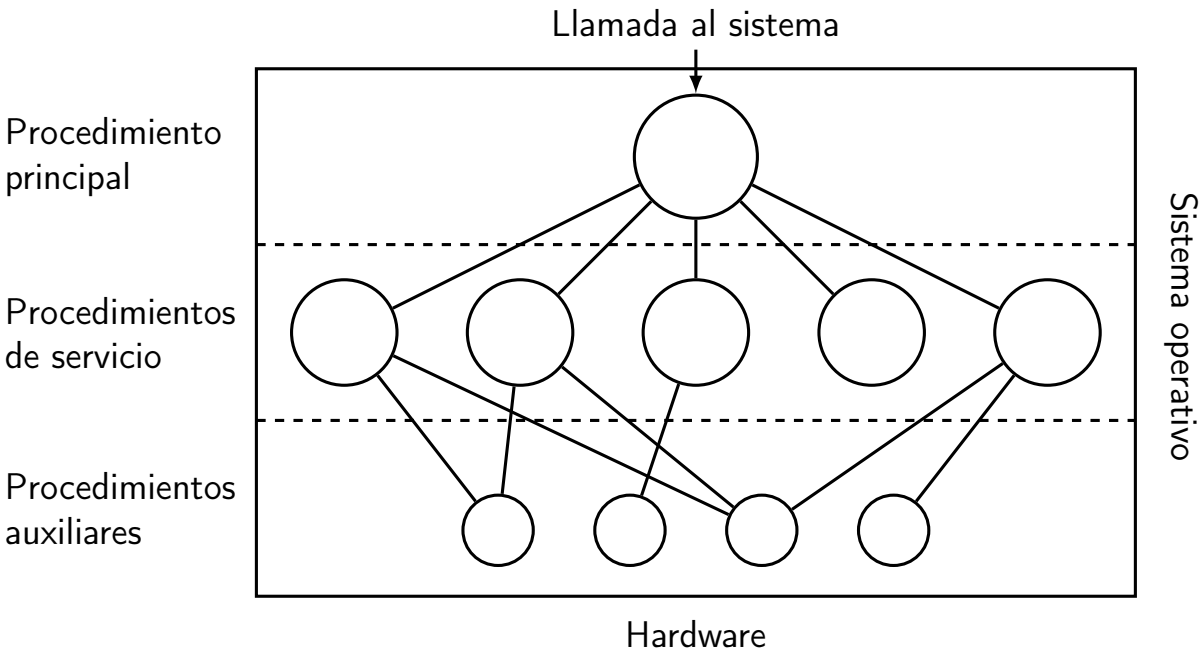


Programas del sistema

- Programas que acompañan al SO para ofrecer un entorno más cómodo para el desarrollo y ejecución de programas
- Son programas normales: para el SO no existe diferencia entre estos programas y los programas de usuario
- Definen la perspectiva que tienen los usuarios del SO
- Varias categorías:
  - Manipulación de ficheros
  - Información de estado: fecha, hora, memoria libre, etc.
  - Desarrollo de programas: compiladores, depuradores, etc.
  - Comunicaciones: correo electrónico, transferencia de ficheros, acceso remoto, etc.
  - Programas de aplicación: editores de texto, bases de datos, hojas de cálculo, juegos, etc.
- El intérprete de órdenes es uno de los más importantes

Sistemas monolíticos

- Se caracterizan por:
  - Colección de procedimientos que se llaman unos a otros
  - Interfaz de procedimientos muy clara
  - No hay ocultación: cada procedimiento es visible a los demás
  - Aunque hay una estructura básica, ésta no está bien definida

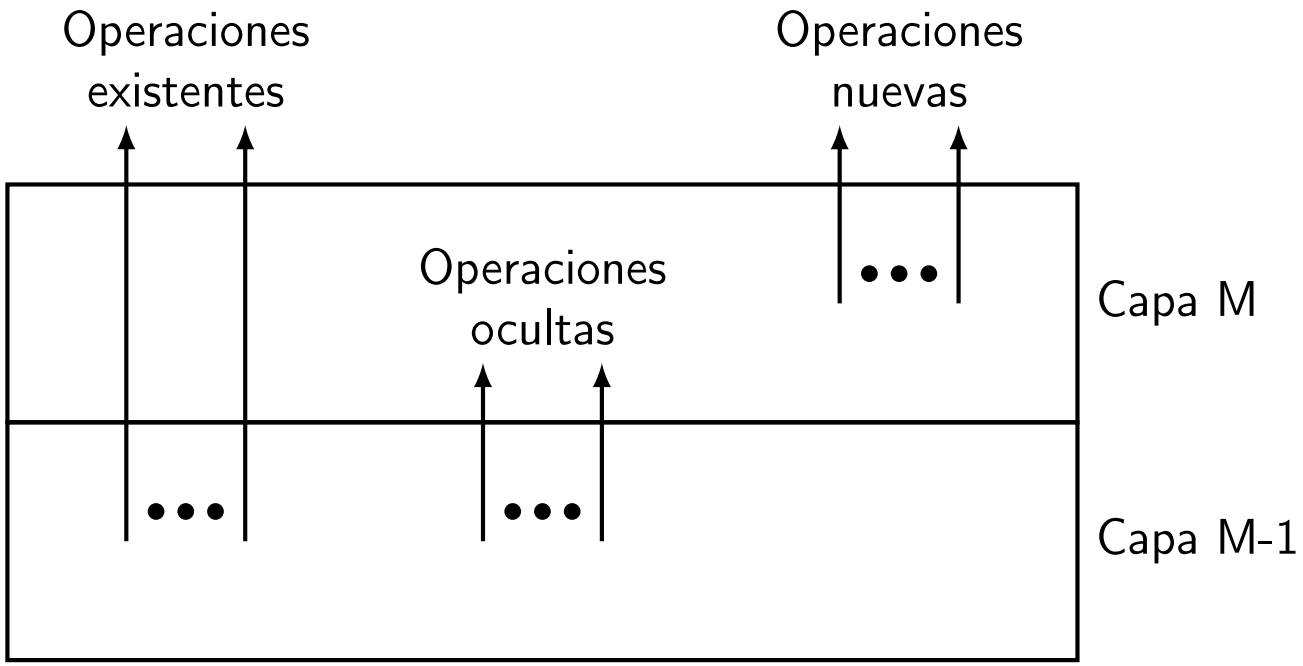


- Ejemplos: Linux, Windows, etc.

Sistemas con capas

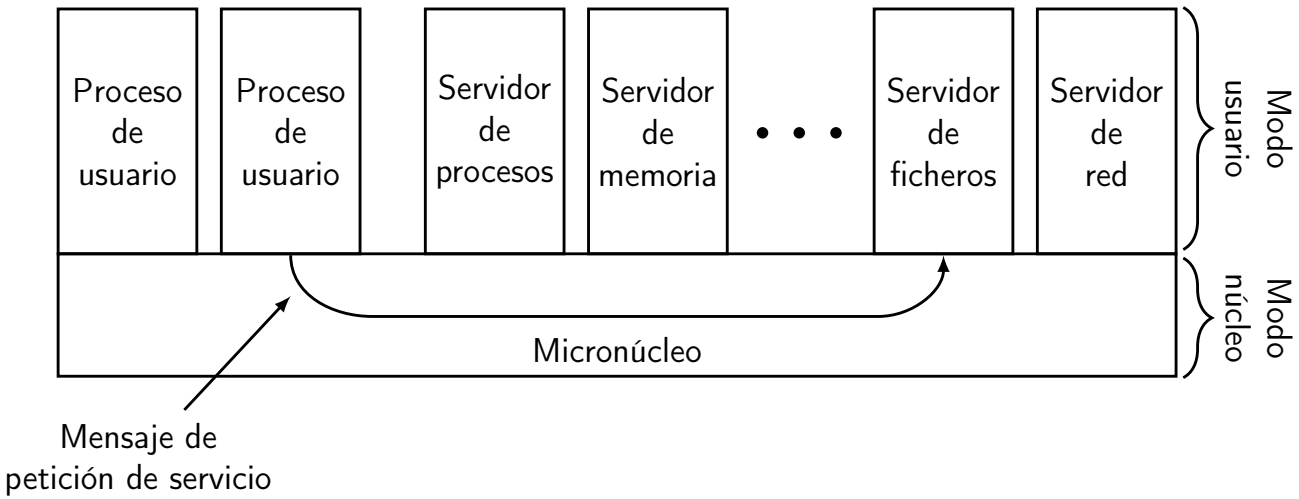
- Diseño modular mediante una jerarquía de capas
- El SO se divide en capas (o niveles), cada una construida sobre la anterior. Capa 0: hardware. Capa N: interfaz con el usuario
- Ventajas:
  - Depuración y verificación capa a capa
  - Ocultación de la información: una capa se puede modificar si no cambia su interfaz
- Inconveniente: dependencias entre capas $\rightarrow$ ¿En qué capa se pone cada función? ¿Cuál es el orden de las capas?

Sistemas con capas



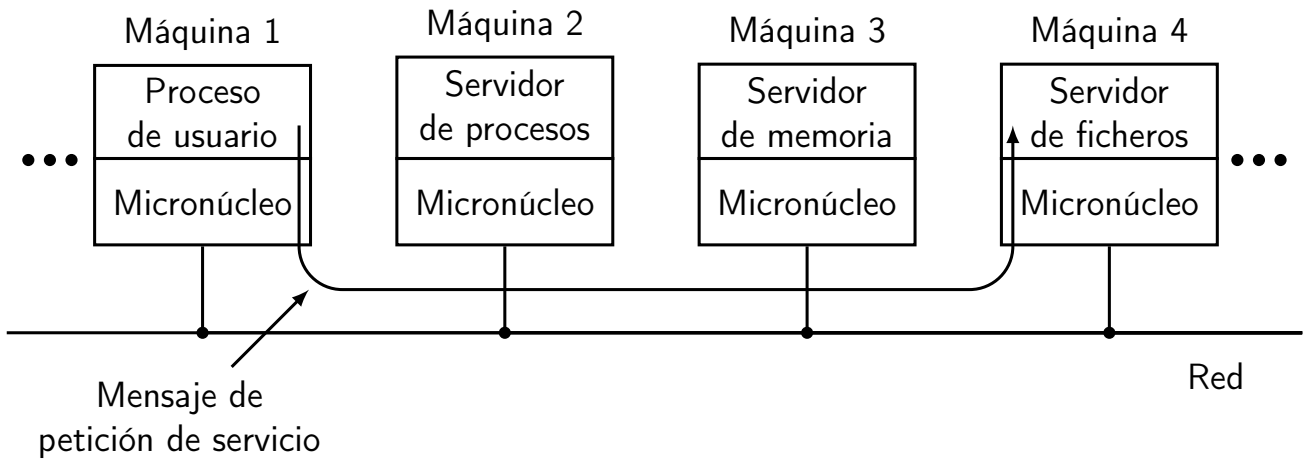
Modelo cliente-servidor

- Implantar la mayoría de las funciones del sistema operativo en los procesos de usuario. Luego 2 tipos de procesos:
  - Procesos clientes: solicitan servicios
  - Procesos servidores: realizan los servicios solicitados por los procesos clientes. También pueden actuar como procesos clientes
- Queda un núcleo mínimo (llamado micronúcleo o *microkernel*) que controla las comunicaciones cliente-servidor



Modelo cliente-servidor

- Ventajas:
  - Los servidores no tienen acceso directo al hardware. Por ejemplo, un fallo en el sistema de ficheros no afecta a todo el sistema
  - Fácil adaptación a los sistemas distribuidos



- ¿Qué pasa con la E/S, etc.? Dos soluciones:
  - Servidor en el núcleo → Desparecen ventajas en ese caso
  - Mensajes especiales enviados por los servidores que captura el núcleo para procesarlos él mismo