



Universidad
Europea
del Atlántico

www.uneatlantico.es

GRADO EN INGENIERÍA INFORMÁTICA

Sistemas Operativos – Tema I

Introducción a los Sistemas Operativos

Juan Luis Vidal Mazón

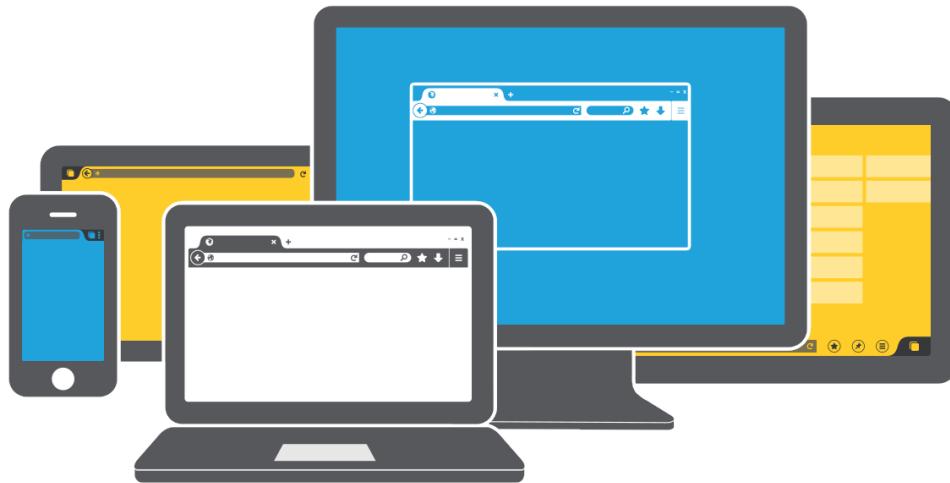
juanluis.vidal@uneatlantico.es

Introducción

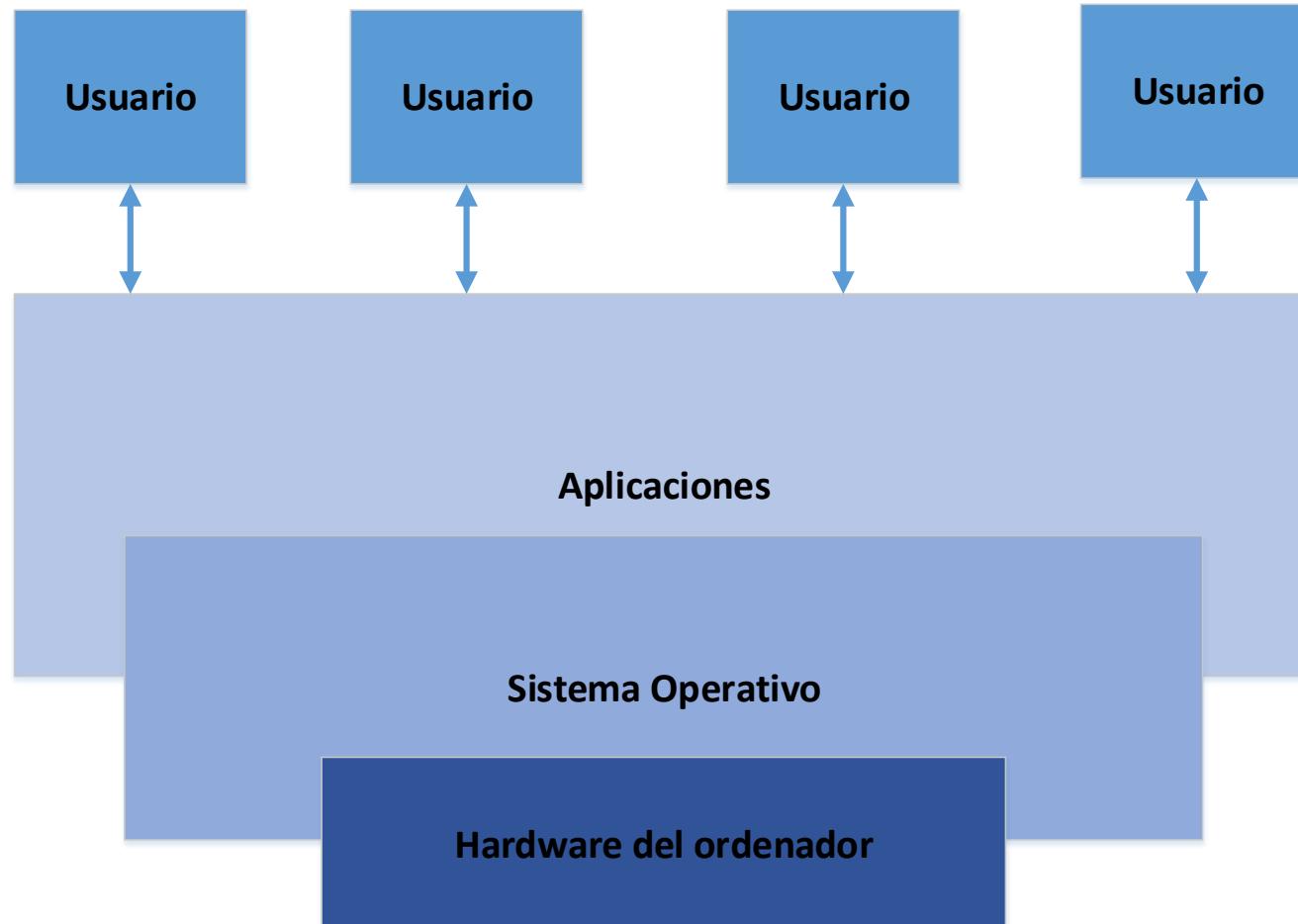
Un sistema operativo es un programa que gestiona el hardware. Proporciona un entorno para que las aplicaciones puedan ser ejecutadas y realiza labores de intermediario entre el usuario del sistema informático y el hardware.

Objetivos de todo sistema operativo

- Ejecutar programas de usuario y resolver cualquier problema
- Hacer sencillo de usar el sistema informático
- Usar el hardware de la manera más eficiente posible



¿Qué hacen los sistemas operativos?



Vista general de los componentes de un sistema informático

¿Qué hacen los sistemas operativos? – Punto de vista del usuario

www.euroatlantico.es

En muchos casos se trata de sistemas de un único usuario. El objetivo es priorizar la facilidad de uso y el rendimiento.

En estos casos no se prioriza el uso compartido de recursos.

En otros escenarios un usuario se sienta ante una terminal que está conectada a otras terminales y servidores.

En estos casos los recursos son compartidos entre los usuarios de la red por lo que el SO está diseñado para alcanzar un equilibrio entre usabilidad de cada usuario individual y la optimización del uso compartido de recursos

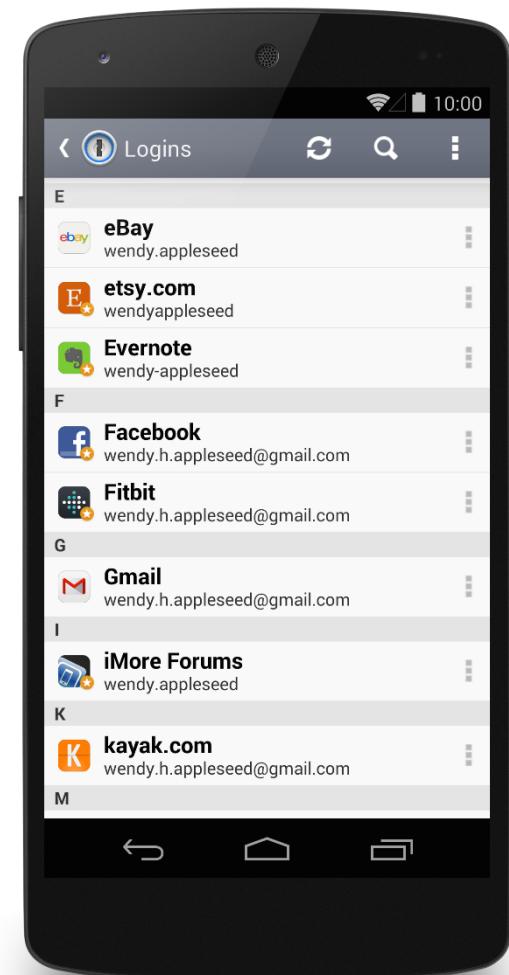


¿Qué hacen los sistemas operativos? – Punto de vista del usuario

Recientemente el diseño de SO móviles está cobrando gran interés.

La mayoría son unidades de un único usuario, conectadas a redes de datos y con unas capacidades hardware que les comienza a perfilar como sustitutos de los tradicionales portátiles o PCs

Otros sistemas no cuentan con vista de usuario, o la tienen muy limitada, como las computadoras integradas en automóviles o electrodomésticos.



¿Qué hacen los sistemas operativos? – Punto de vista del sistema



El SO es programa más íntimamente ligado al hardware.

Actúa como un asignador de recursos.

Otra visión es aquella que se centra en sus funciones de control de los dispositivos de E/S y en la ejecución de software de usuario

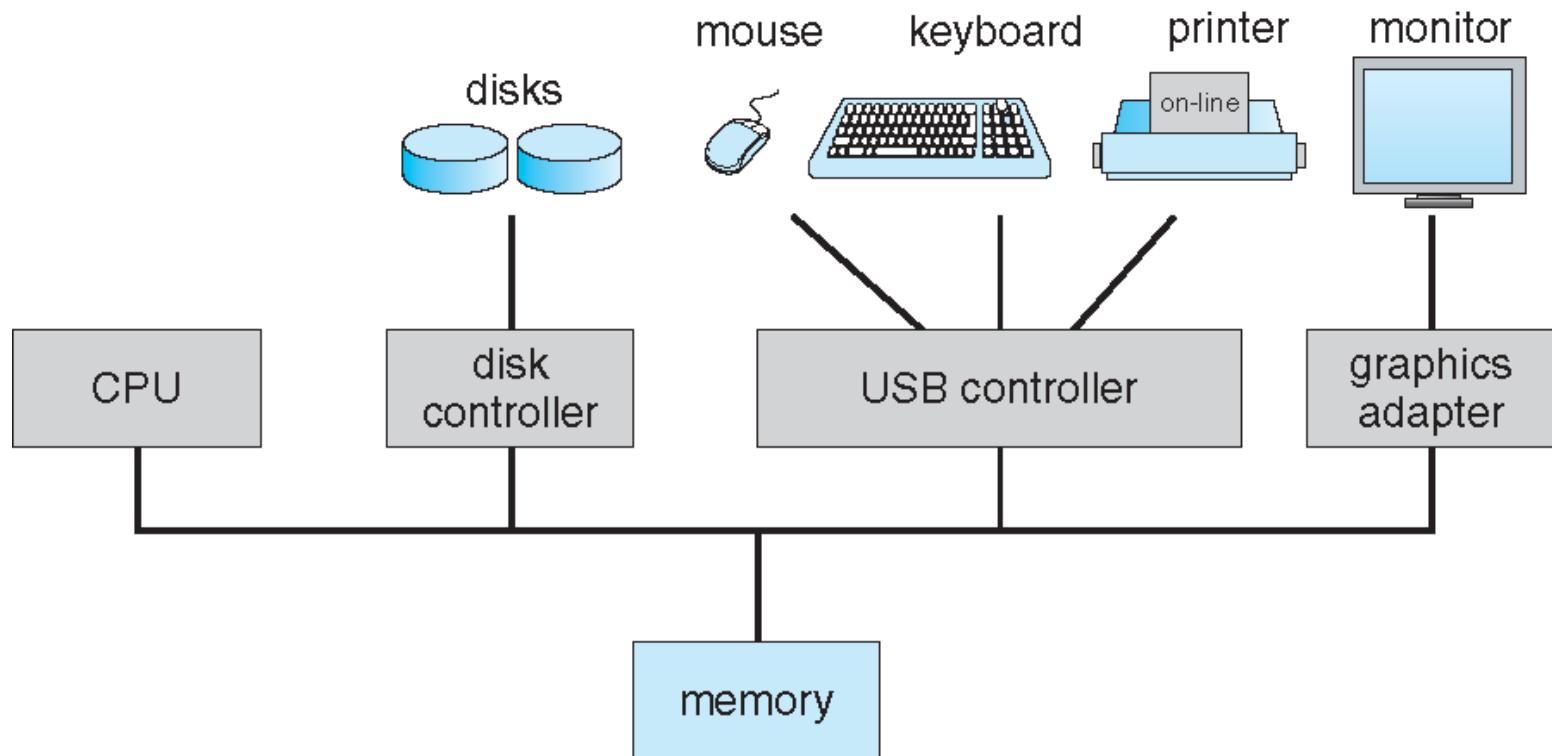
Actúa también como programa de control, que trata de evitar errores y supervisa la ejecución del software.

Definición de Sistema Operativo

- No existe una definición universalmente aceptada
- “Todo lo que un fabricante incluye cuando eliges un determinado sistema operativo” es una primera aproximación.
 - Pero varía mucho!
- “Es el programa que siempre se está ejecutando en un computador”, el **Kernel**
- Todo lo demás son:
 - Programas del sistema (que vienen con el SO)
 - Programas de aplicación



Organización de una computadora



Esquema general de una computadora moderna

Organización de una computadora – Arranque del sistema

www.uneatlantico.es

Cuando se presiona el botón de inicio ocurre lo siguiente:

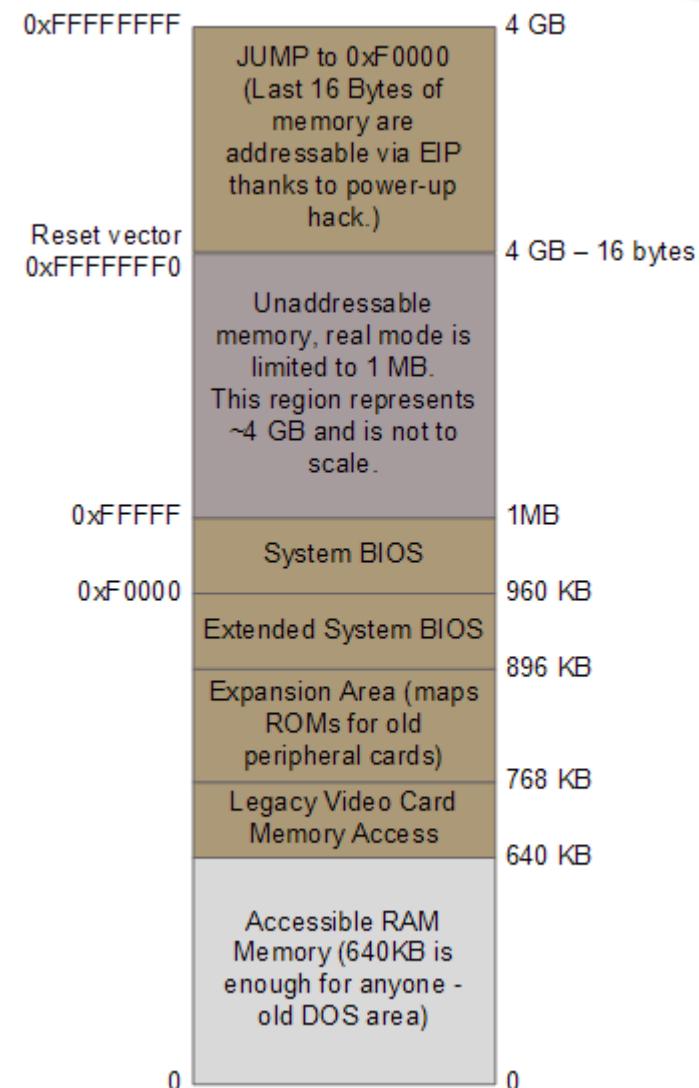
- Si el sistema es multiprocesador se elige uno de los procesadores. El resto los activará el kernel.
- Los pins de la CPU se resetean y los registros se colocan a valores específicos.
- La CPU comienza a ejecutarse en modo real. Sólo puede acceder a un mega de memoria.
- La CPU accede a la dirección de memoria **BIOS**. Para ello las CPU Intel usan un “hack” que hace que la primera dirección de memoria que se lee es la 0xFFFFFFFF0, 16 bytes menos que 4 GB.



Organización de una computadora – Arranque del sistema

www.uneatlantico.es

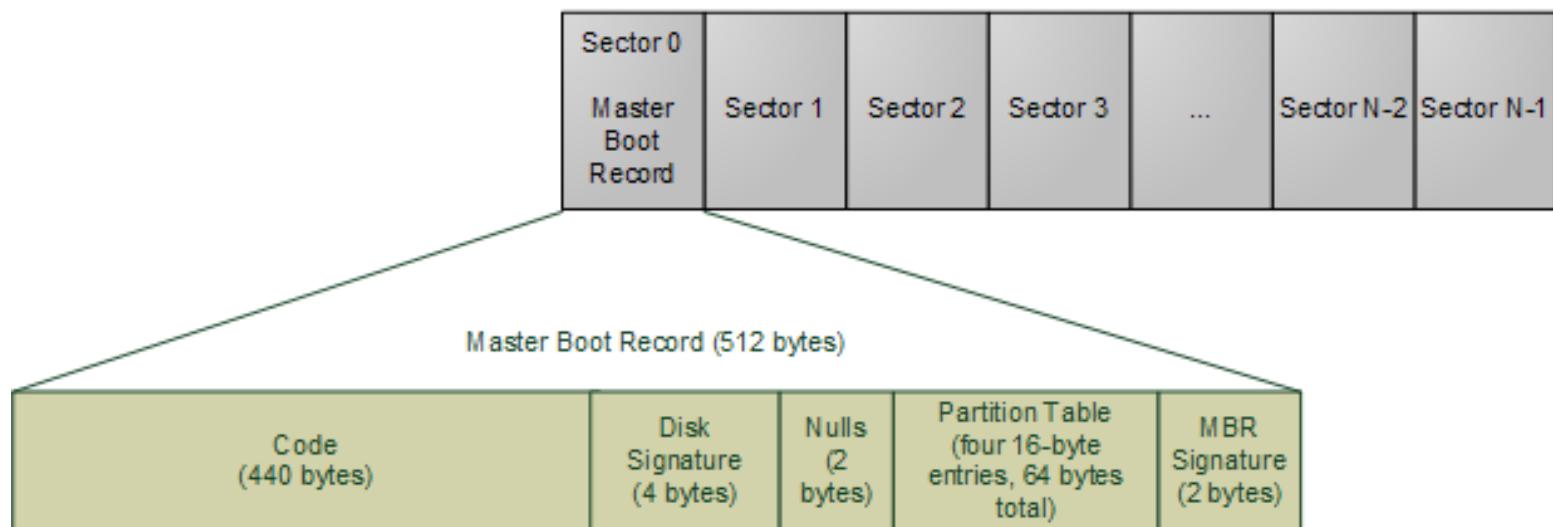
- Esta dirección contiene la dirección de memoria base de la BIOS.
- La BIOS ejecuta **POST** (Power-On Self Test) y cualquier otra función de chequeo
 - Inicialización del hardware, que ejecuta sus propias BIOS
- La BIOS salta al **MBR** (master boot record) y copia sus contenidos a la memoria, cediéndole el control.
 - El MBR es el primer sector de una partición (sector 1 del cilindro 0, cabeza 0)



Organización de una computadora – Arranque del sistema

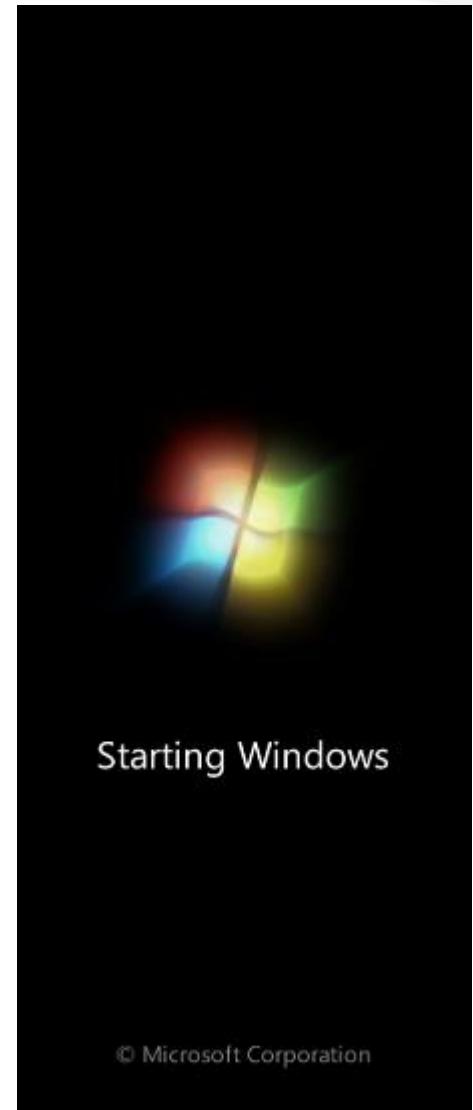
- De los 512 bytes que ocupa el MBR, los primeros 446 son el cargador primario, o PBL (*primary boot loader*)
- Los siguientes 64 contienen los registros de cada partición de disco.
- Los siguientes dos deben contener el número 0xAA55, que actúa a modo de validación.

N-sector disk drive. Each sector has 512 bytes.



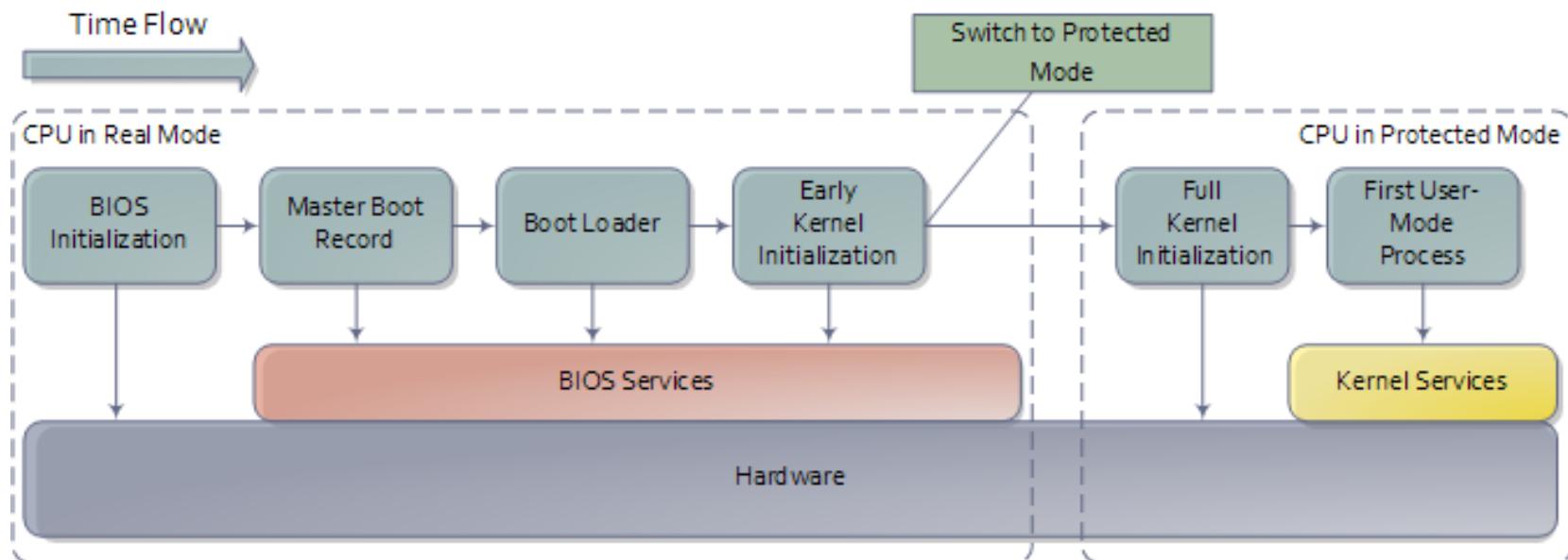
Organización de una computadora – Arranque del sistema

- La BIOS lee estos contenidos y los coloca en la posición de memoria 0x7c00, comenzando su ejecución.
- Debido a su pequeño tamaño el MBR carga otros sectores de disco que contienen información adicional de arranque.
- El MBR carga la segunda fase del inicio (c:\ntldr en Windows) y lee la configuración de inicio (boot.ini en Windows).
- En este punto el cargador necesita ya iniciar el kernel del SO (ntoskrnl.exe en Windows) y le cede la ejecución.



Organización de una computadora – Arranque del sistema

www.uneatlantico.es



Esquema del arranque de una computadora

Organización de una computadora

Cuando finaliza la carga del kernel, este puede ya comenzar a dar servicio al sistema y a sus usuarios.

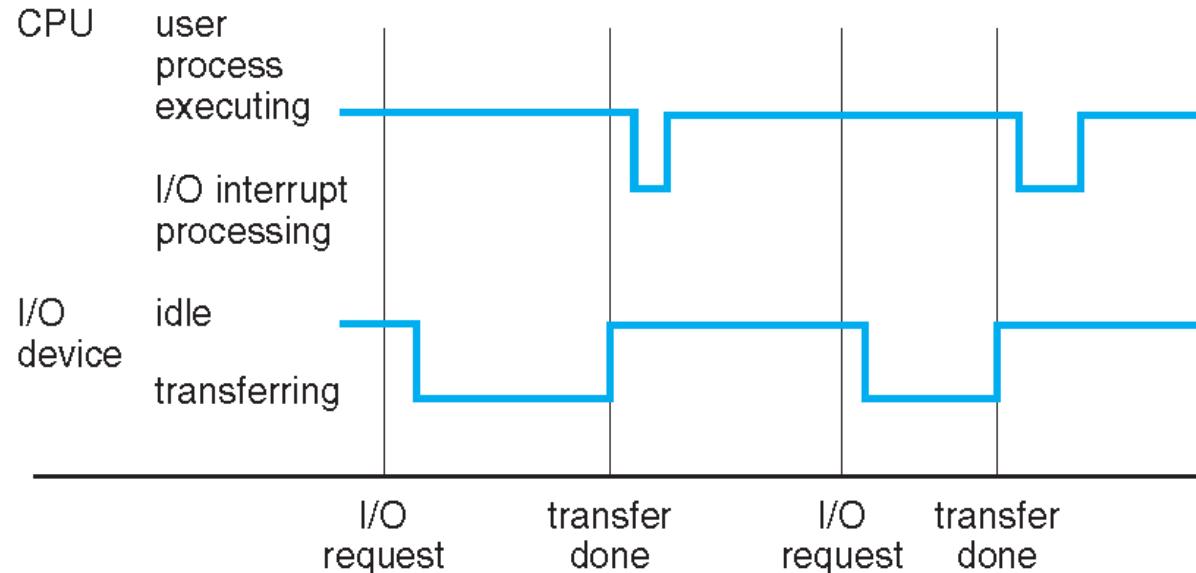
Algunos de estos servicios se proporcionan de manera externa al kernel, a través de los programas de sistema que se cargan al inicio.

Cuando esta fase se completa el sistema está completamente iniciado y la CPU queda a la espera de eventos, señalado a través de las **interrupciones**.

Las interrupciones pueden ser:

- De hardware, que las lanza a través del bus del sistema
- De software, a través de operaciones especiales denominadas '**llamadas al sistema**'

Organización de una computadora



Cuando la CPU es interrumpida abandona lo que hace, y ejecuta una posición de memoria que contiene la rutina encargada de dar servicio a la interrupción.

Una vez que ésta termina, la CPU reanuda la operación que estuviese haciendo.

Organización de una computadora – Estructura de almacenamiento

www.atlantico.es

En una maquina con arquitectura Von Neumann, el ciclo de ejecución de instrucciones consiste en:

- Extraer una instrucción de la memoria y colocarla dentro del registro de instrucciones de la CPU.
- La instrucción se decodifica.
- Adicionalmente, pueden requerirse mas operandos de la memoria, que se almacenarán en registros internos.
- La instrucción se ejecuta junto con los operandos que sean necesarios.
- El resultado de la operación se almacena en memoria

Por tanto los programa deben estar en la memoria principal para ser ejecutados



John Von Neumann

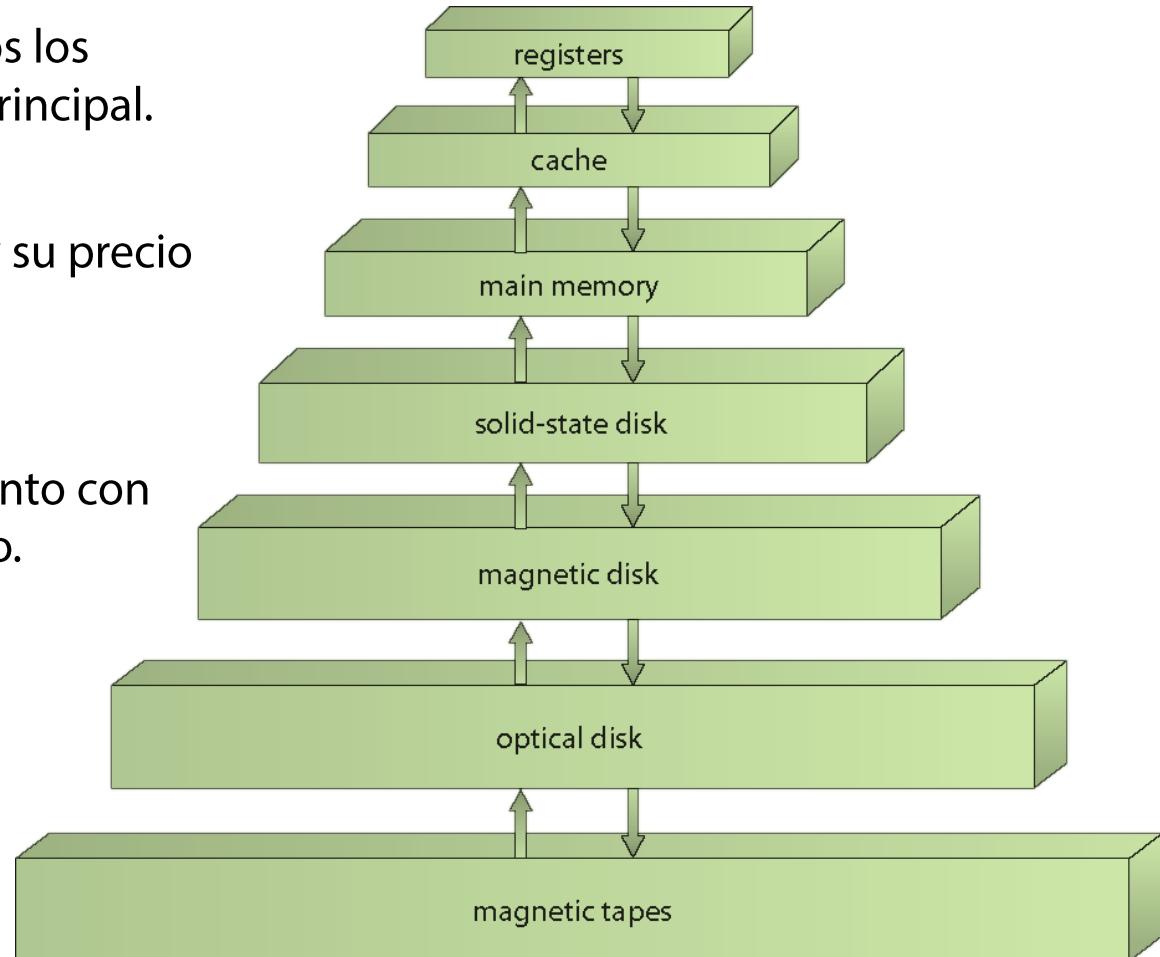
Organización de una computadora – Estructura de almacenamiento

www.atlantico.es

Sería ideal poder tener todos los programas en la memoria principal.
Pero:

- Es demasiado pequeña y su precio por bit muy elevado
- Es volátil

Los sistemas cuentan por tanto con almacenamiento secundario.



Jerarquía de dispositivos de almacenamiento

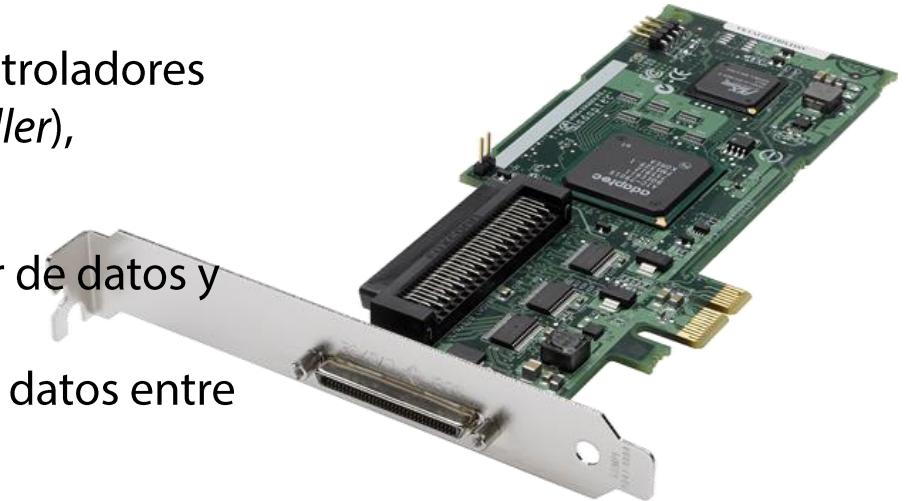
Estructura de E/S

Gran parte del código de un SO se dedica a gestionar los dispositivos de E/S.

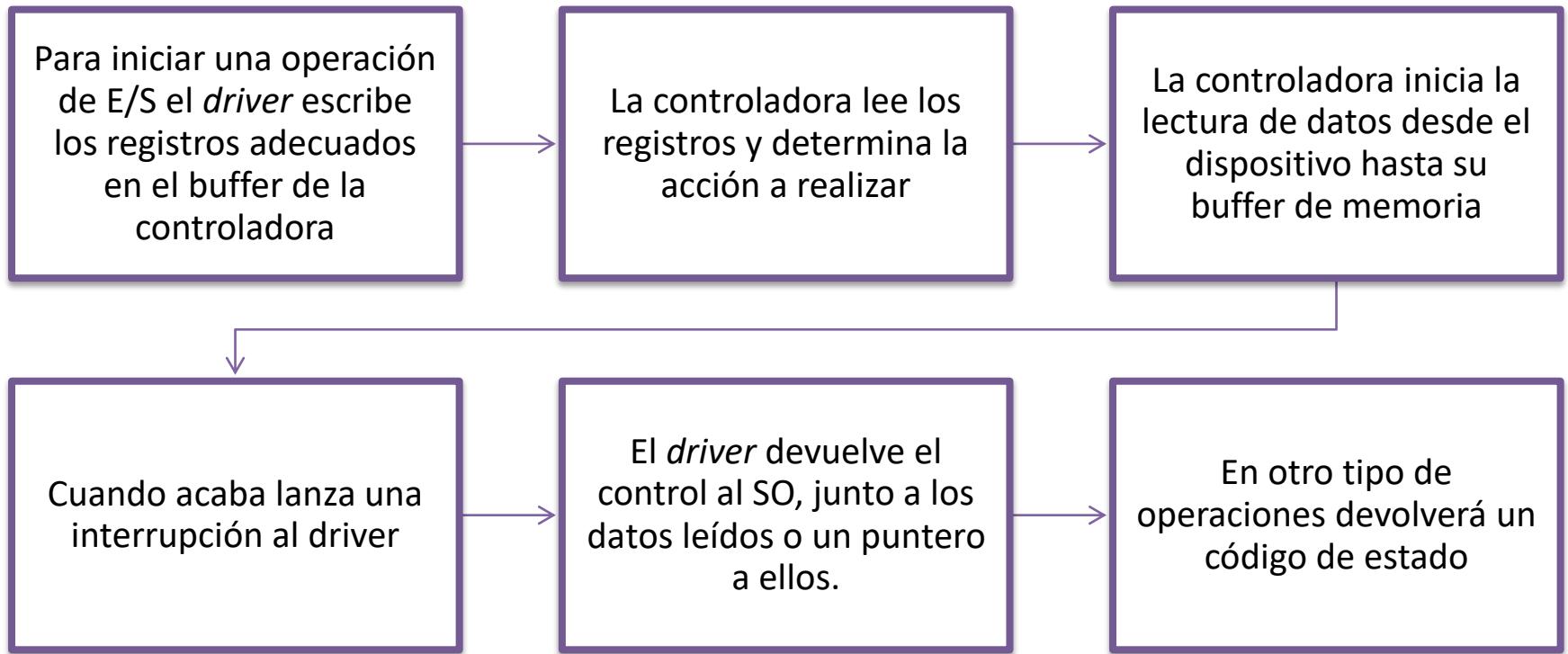
Un sistema se compone de CPUs y controladores hardware de dispositivo (*device controller*), conectados a un bus de datos común.

La controladora mantiene algún buffer de datos y un conjunto de registros especiales. La controladora es responsable de mover datos entre el dispositivo y el buffer.

Cada controlador maneja uno o más dispositivos, y el SO gestiona al controlador hardware a través del controlador de dispositivo (*device driver*).

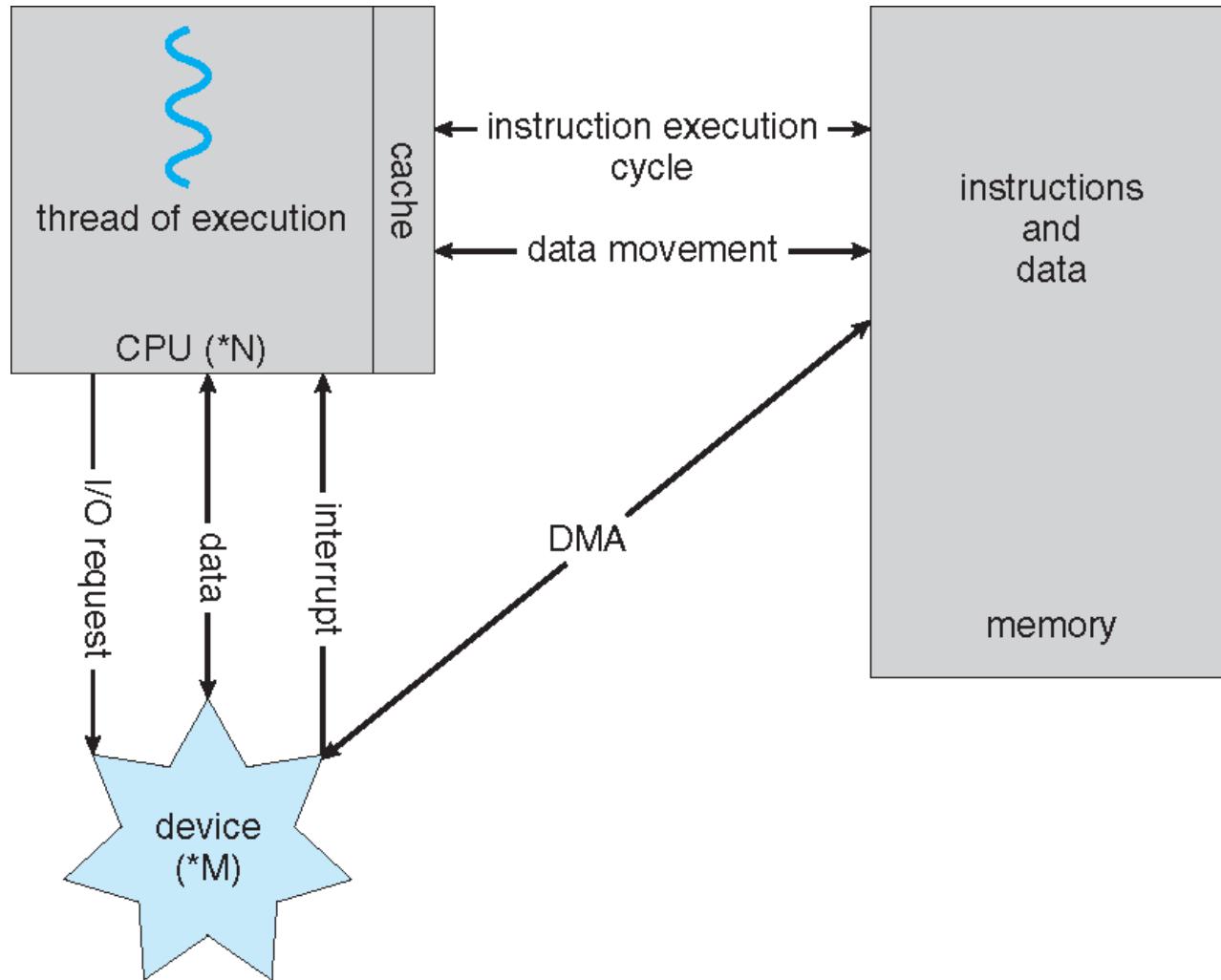


Estructura de E/S



Este sistema es valido para pequeñas cantidades de datos, pero para grandes operaciones es lento. Para ello se utiliza el Acceso Directo a Memoria (DMA), que mueve datos del buffer a la memoria principal sin intervención de la CPU.

Estructura de E/S



Arquitecturas de computadores

- Sistemas mono-procesador
- Sistemas multiprocesador

Ventajas

- Mayor capacidad de trabajo
- Economía de escala
- Mejora de la fiabilidad
 - Degradación suave
 - Sistemas tolerantes al fallo



Arquitecturas de computadores

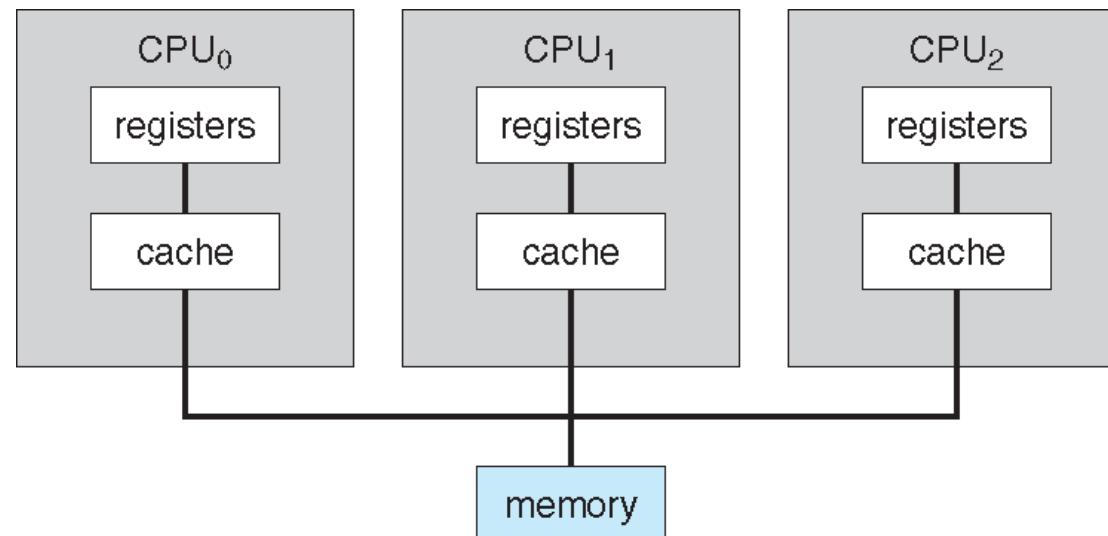
Los equipos multiprocesador son de dos tipos.

Multiprocesamiento asimétrico

- Un procesador 'jefe' coordina a los demás. Cada uno de los otros se encarga de una tarea específica

Multiprocesamiento simétrico

- Cada procesador realiza todas las tareas del SO. No hay un 'jefe'.

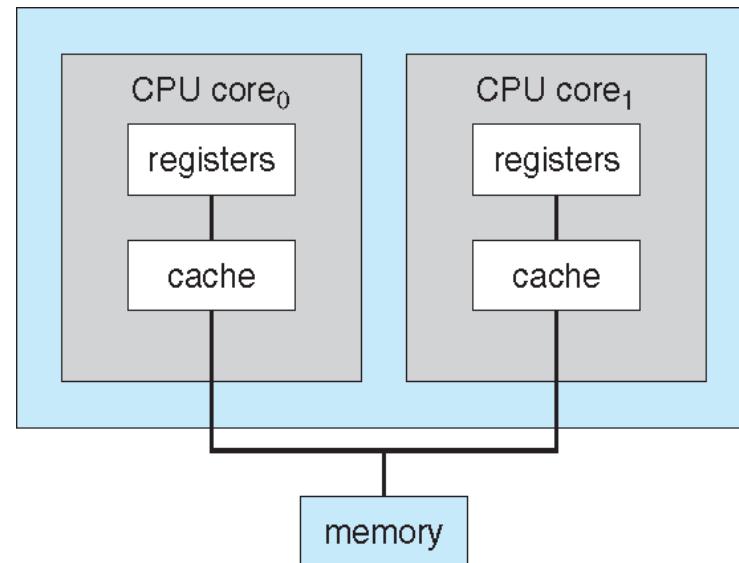


Arquitecturas de computadores

Una tendencia actual es la de incluir varios núcleos dentro de un mismo chip. Estos sistemas son los llamados multi-núcleo o *multicore*.

Pueden llegar a ser más eficientes que los sistemas multi-procesador de núcleo único porque la comunicación es más rápida. Además son más eficientes energéticamente.

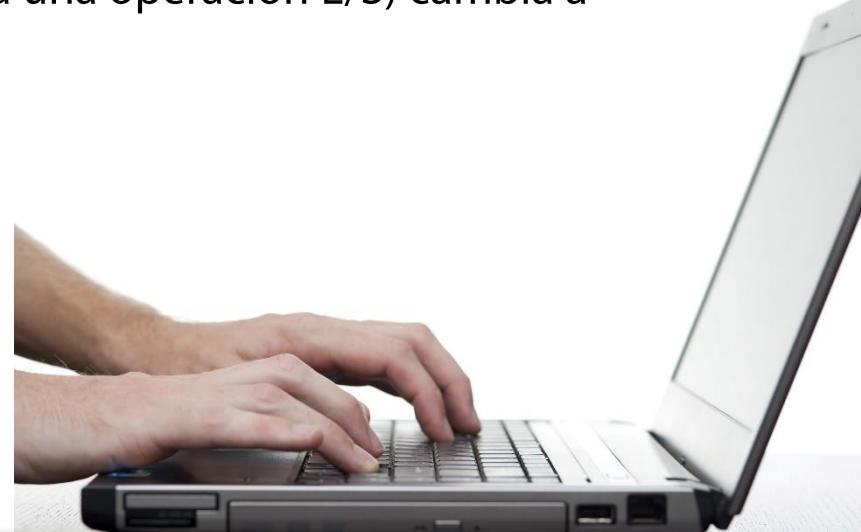
Todos los sistemas multi-núcleo son multiprocesador pero no todos los sistemas multi-procesador son multi-núcleo.



Estructura del sistema operativo

Multiprogramación:

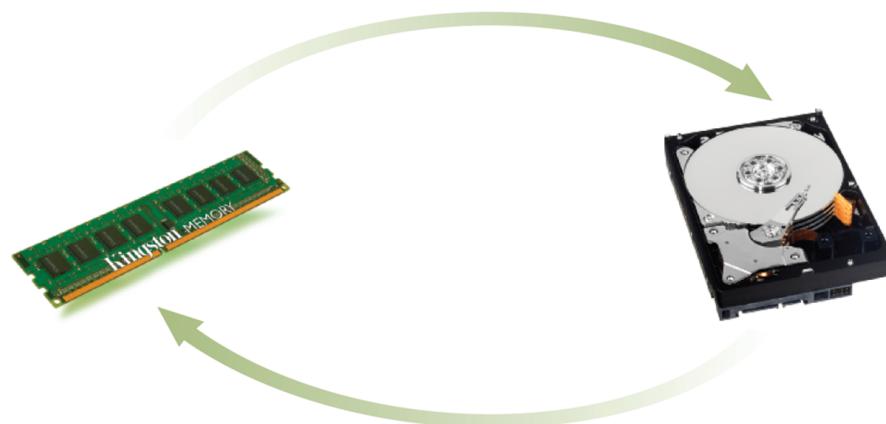
- Un único usuario / proceso es incapaz de mantener ocupada la CPU y los sistemas E/S de manera constante.
- La multiprogramación organiza los tareas (código y datos) de manera que la CPU tiene siempre uno que ejecutar.
- Las tareas se mantienen en memoria y el SO elige uno para ejecutar siguiendo un algoritmo.
- Cuando tiene que esperar (por ejemplo a una operación E/S) cambia a otro.



Multitarea (multitasking)

La **multitarea** es una extensión natural de la multiprogramación en la que la CPU cambia de tarea de manera tan rápida que los usuarios pueden interactuar con todos los procesos a la vez.

- El tiempo de respuesta debe ser de menos de un segundo.
- Cada usuario tiene al menos un programa ejecutándose en memoria.
- Si varias tareas están listas para ejecutarse → Gestión de la CPU
- Si los procesos no caben en memoria → Paginación / Swapping
- La memoria virtual permite ejecutar procesos que no están cargados completamente en memoria.



Operaciones del sistema operativo

Los SO están controlados mediante interrupciones

- Interrupciones de hardware lanzadas por los dispositivos
- Interrupciones software lanzadas por los programas en ejecución
 - Errores en el software (divisiones por cero...)
 - Petición de servicios al SO
 - Otros errores (procesos que se sobreescreiben entre ellos o al SO)

```
public class MyException extends Exception{}  
//try, throw y catch  
try {  
    Integer i;  
    //Código con probabilidad de fallar  
    if ( i < 5 ) throw new MyException();  
} catch ( MyException e ) {  
    //Gestión del error  
}
```

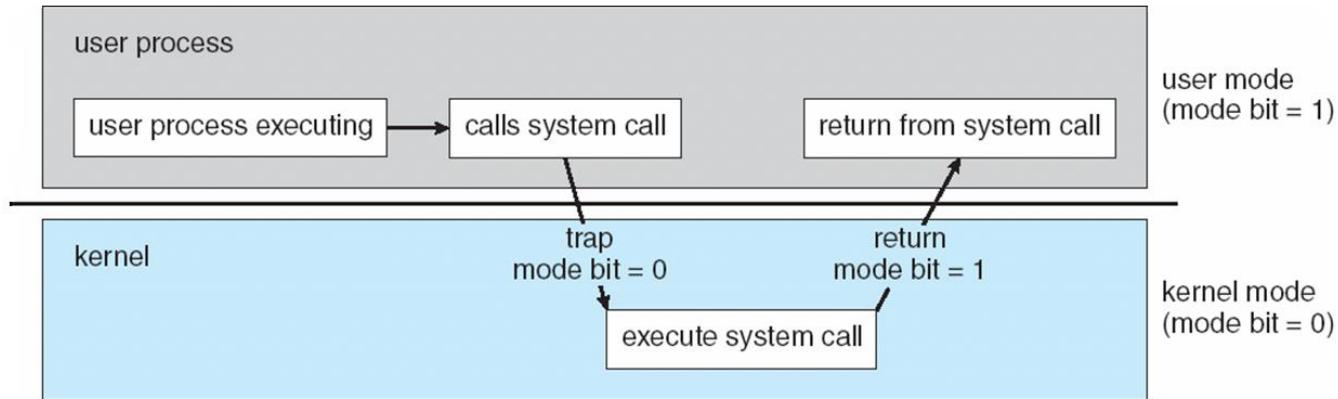
Operaciones del sistema operativo – Modo dual

Las **operaciones en modo dual** permiten al SO protegerse a sí mismo y a otros componentes del sistema

- Se distingue entre **modo usuario** y **modo kernel**.
- Existe un bit de modo gestionado por hardware.
 - Permite distinguir cuando el sistema ejecuta código kernel o código de usuario.
 - Se definen ciertas instrucciones como **privilegiadas**. Sólo pueden ejecutarse en modo kernel (instrucciones de E/S...).
 - Las llamadas al sistema activan el modo kernel, al acabar devuelven el sistema a modo usuario



Operaciones del sistema operativo – Modo dual



Cuando un sistema se inicia lo hace en modo kernel y después cede el control a las aplicaciones de usuario (modo usuario).

El SO vuelve a tener el control por una llamada al sistema, una interrupción o instrucción trap. Para ello se hace uso del vector de interrupciones.

Servicios del sistema operativo



Interfaz de usuario



Ejecución de programas



Operaciones E/S



Manejo sistema ficheros



Comunicaciones



Detección de errores



Asignación de recursos

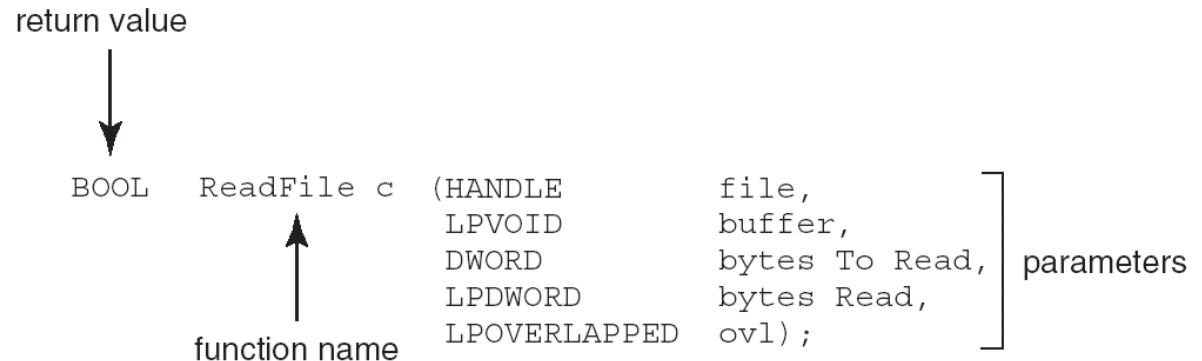


Protección y seguridad

Llamadas al sistema

Interfaz del SO con las que poder llamar a servicios que este proporciona.

Programadas en C, C++ o ensamblador.



Generalmente los programadores no descienden a este nivel sino que usan las APIs del sistema

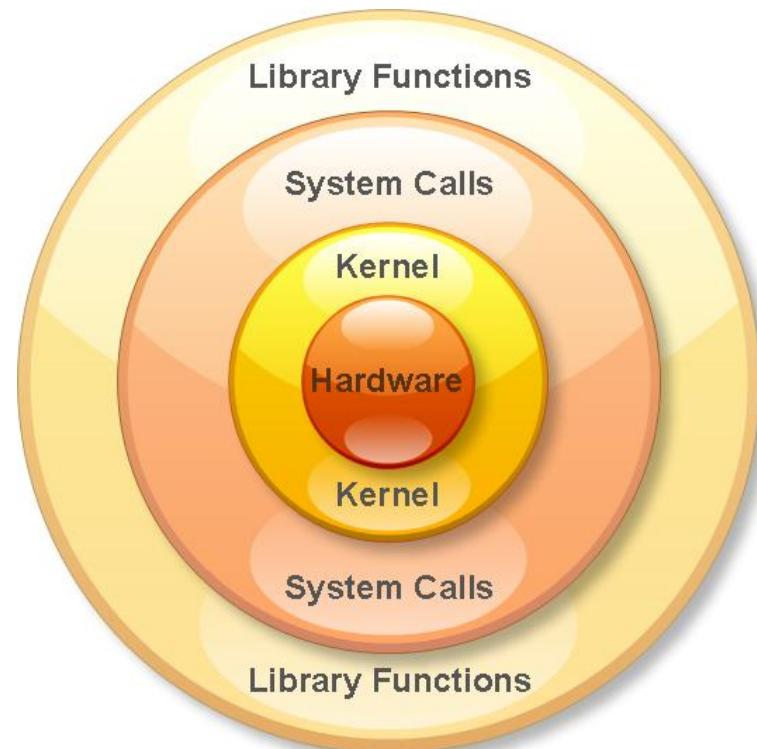
Llamadas al sistema - Motivación

Se supone que un proceso no debe acceder al kernel. No debe acceder ni a la memoria del kernel o a sus funciones de manera directa.

La razones para ello es que:

- Puede alterar a otros procesos que se estén ejecutando
- Puede causar daño físico a los dispositivos
- Pueden alterar el comportamiento de todo el sistema

La llamada al sistema proporciona un mecanismo seguro para solicitar determinadas funciones del kernel.

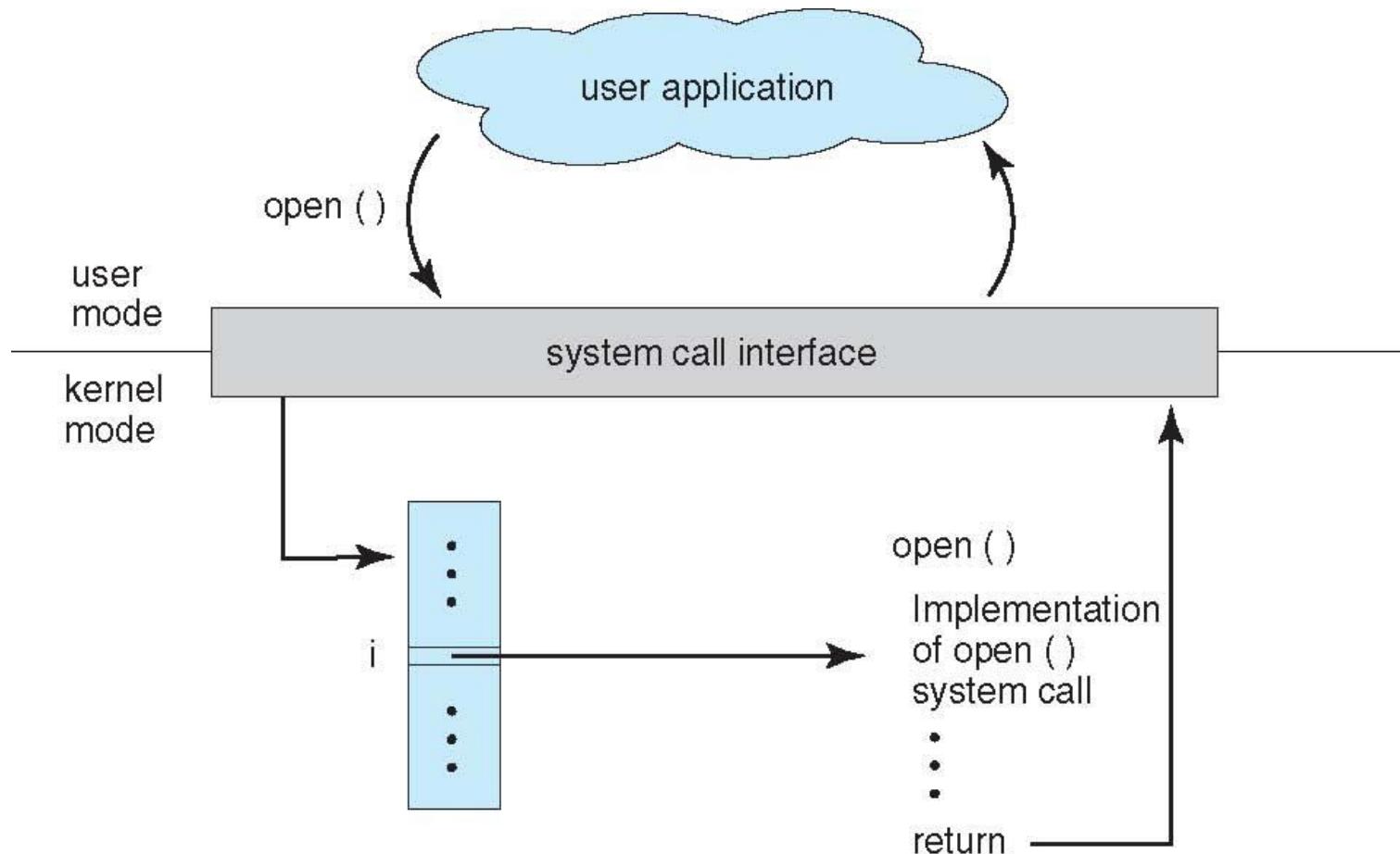


Llamadas al sistema

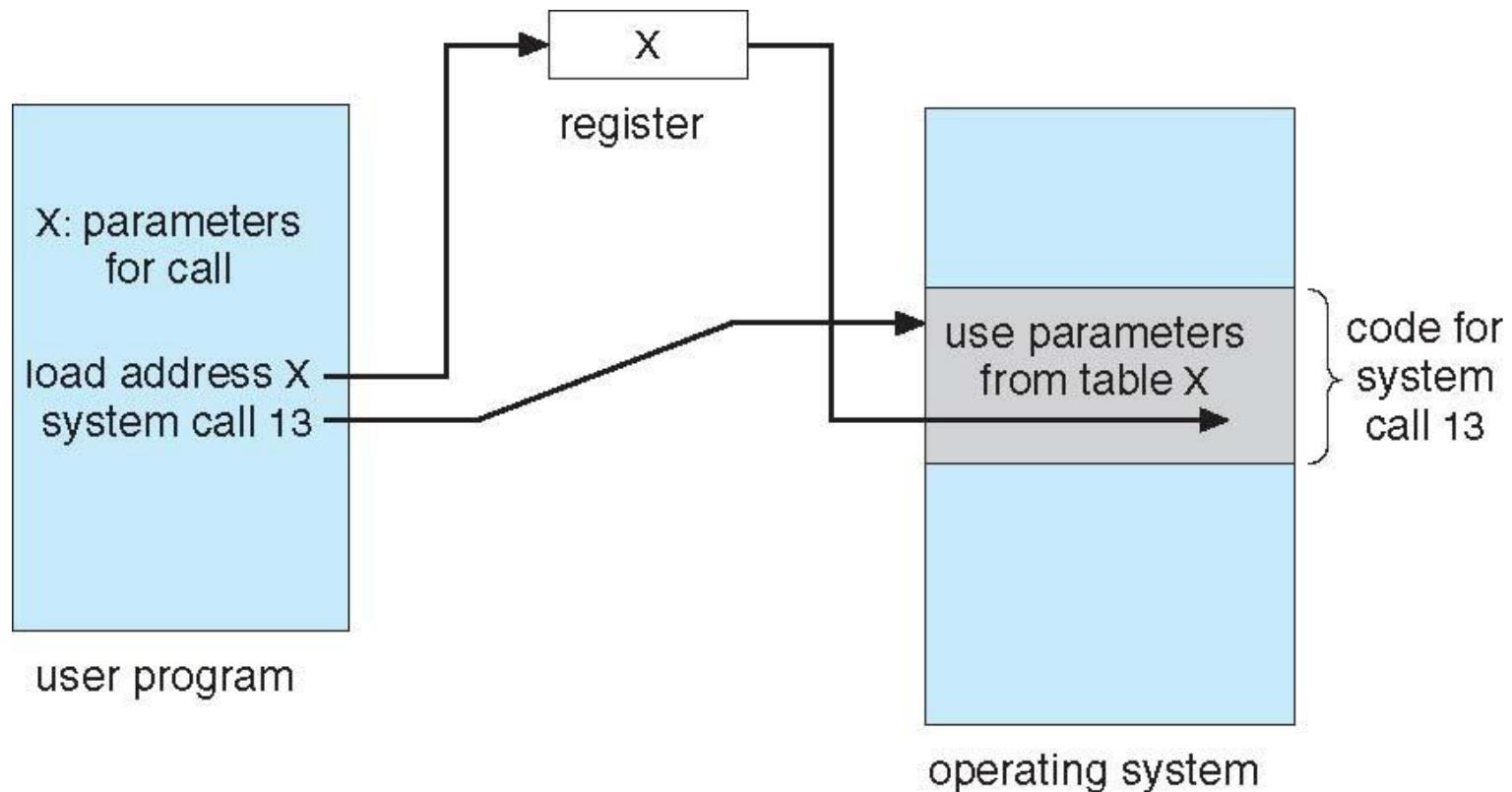
```
#include <stdio.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <asm/unistd.h>

void function(void)
{
    int pid;
    pid = __syscall(__NR_getpid);
}
```

Llamadas al sistema



Llamadas al sistema – Paso de parametros



Llamadas al sistema

www.uneatlantico.es

Control de procesos

- Crear procesos, terminarlos
- Fin de procesos, abortar procesos
- cargar, ejecutar
- establecer y leer atributos de los procesos
- esperar un tiempo
- reservar y liberar memoria
- Volcar memoria si hay error
- **Locks** para gestionar el acceso a datos compartidos

Llamadas al sistema

File management

- Creación y borrado de fichero
- Abrir, cerrar fichero
- Leer, escribir, reposicionar
- Leer y establecer atributos de fichero

Device management

- Solicitar y liberar dispositivos
- Leer, escribir, reposicionarse
- Leer o establecer atributos de un dispositivo

Llamadas al sistema

Information maintenance

- Establecer o leer la fecha y la hora
- Escribir o leer datos del sistema
- Establecer o leer atributos de dispositivos, ficheros o procesos

Communications

- Crear o borrar comunicaciones con otros equipos
- Enviar y recibir mensajes entre procesos
 - O de cliente a servidor
- **Shared-memory model** crear y ganar acceso a regiones de memoria compartida
- Enviar información de estado
- Enchufar o desenchufar dispositivos remotos

Llamadas al sistema

www.uneatlantico.es

Protection

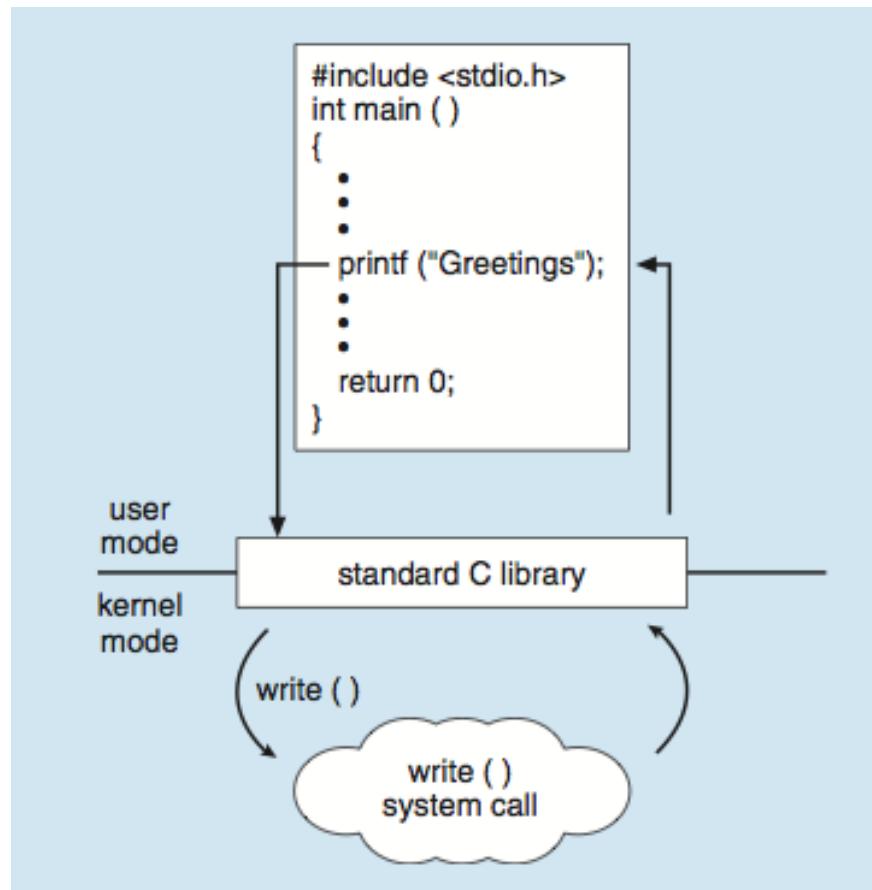
- Garantizar acceso a recursos
- Establecer y leer permisos
- Permitir y denegar acceso a usuarios

Llamadas al sistema

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Llamadas al sistema

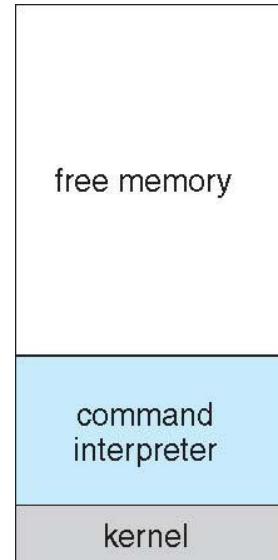
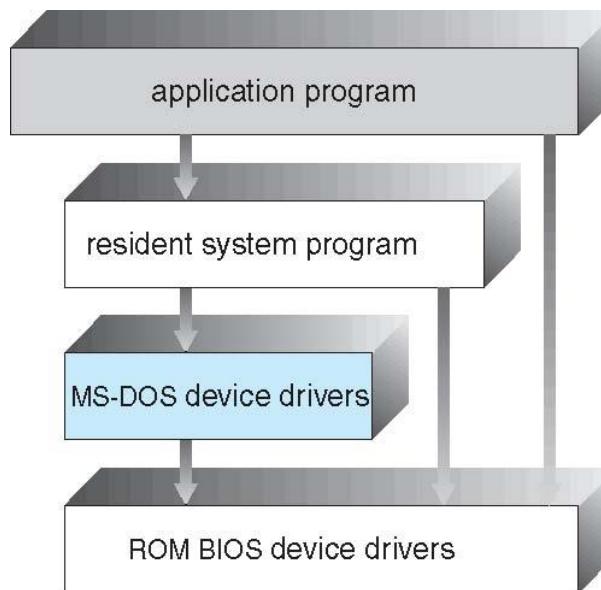
Programa en C que llama a la función printf(), la cual invoca a la llamada al Sistema write()



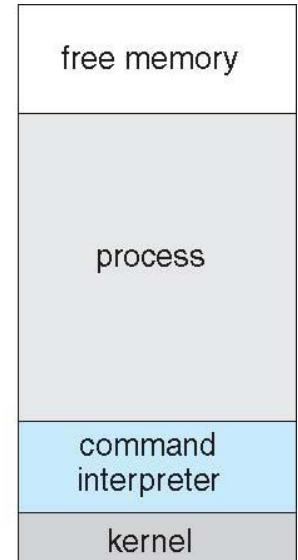
Organización de un SO

Estructura simple. No tienen una estructura definida, como MS-DOS o las primeras versiones de UNIX.

No está dividido en módulos. No están bien separados



(a)



(b)

Al inicio del Sistema

Ejecutando una tarea

Organización de un SO

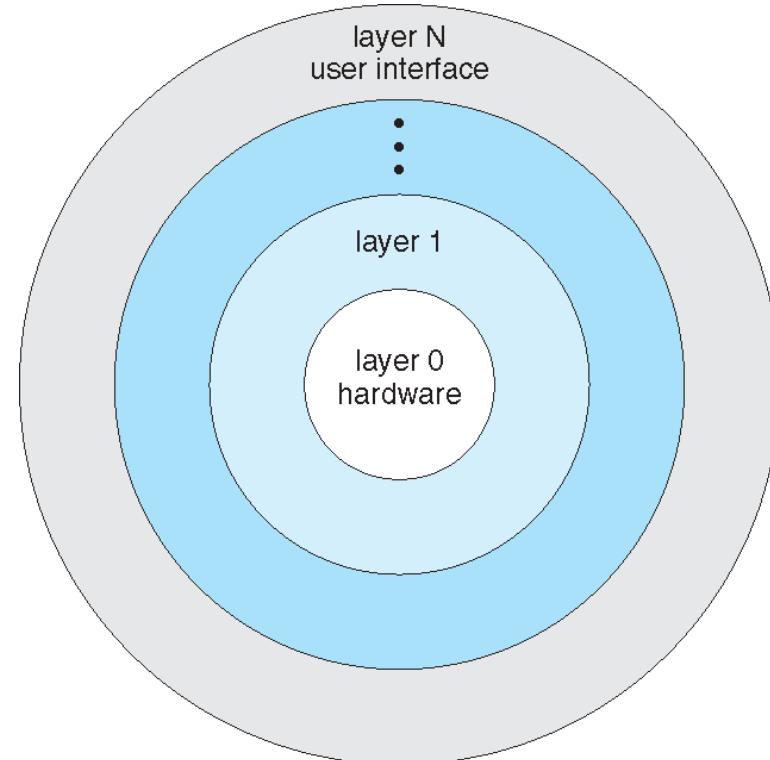
Estructura en capas

El SO está organizado en niveles independientes entre sí. Estos niveles son divisiones lógicas de las tareas desempeñadas por el SO.

Las funciones de una capa pueden ser invocadas por las capas superiores y ésta puede invocar las funciones de las capas inferiores.

Facilidad de ensamblaje y de comprobación de errores.

La dificultad estriba en realizar correctamente las divisiones



Organización de un SO

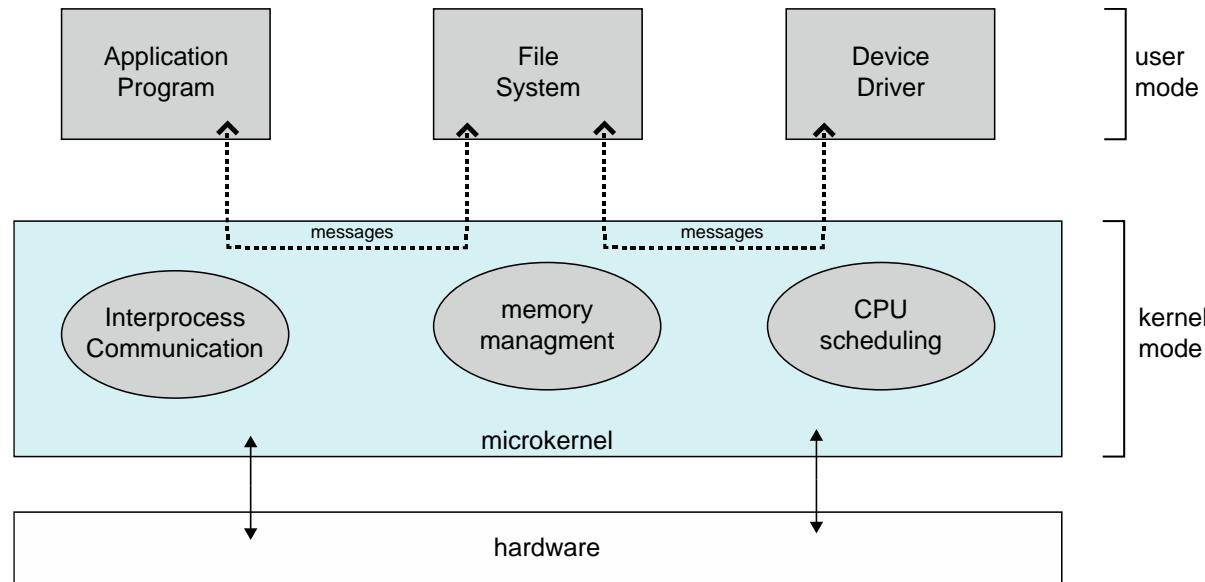
Microkernel

Consistente en reducir el kernel hasta dejar en él únicamente las funciones mínimas esenciales.

El resto de funciones se llevan al espacio de usuario.

Hace muy sencilla la modificación del SO

Puede presentar problemas de rendimiento.



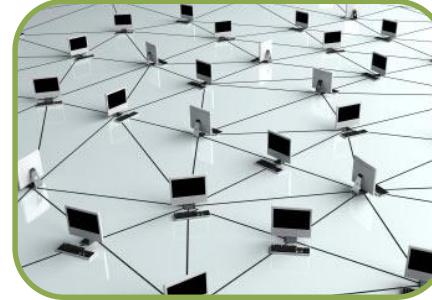
Entornos de computación



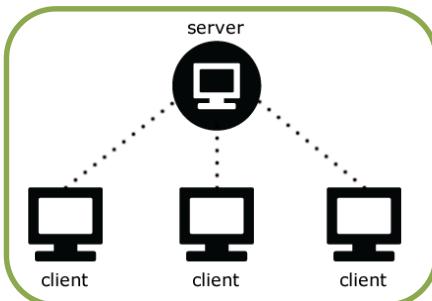
Computación
tradicional



Computación
móvil



Sistemas
distribuidos



Sistemas cliente
servidor



Sistemas P2P



Virtualización



Universidad
Europea
del Atlántico

www.uneatlantico.es