# Sistemas Operativos y Redes

# Introducción

Service Science, Management and Engineering

Grado en Ciencia, Gestión e Ingeniería de Servicios

**Profesor:**

**David Granada**

david.granada@urjc.es

# Tipos de S.O.

- S.O. para mainframe
- S.O. para servidores
- S.O. multiprocesador
- S.O. para ordenadores personales
- S.O. de tiempo real
- S.O. integrados
- S.O. para tarjetas inteligentes

# Clasificación de los S.O.:

- Por el número de **usuarios**: monousuario vs. multiusuario
- Por el número de **tareas**: monotarea vs. multitarea
- Por el numero de **procesadores** que se pueden usar simultáneamente: monoprocesador vs. multiprocesador
- Por el tipo de **interfaz**: modo texto vs. modo gráfico
- Por la **estructura interna**: monolítico, máquinas virtuales, cliente-servidor, exokernels.

# Otros conceptos de los S.O.

- La mayoría de los S.O. proporcionan ciertos conceptos básicos y abstracciones tales como **procesos**, espacios de **direcciones** y **archivos**, los cuales son la base para entender su funcionamiento.

- La **multiprogramación** aumenta la **eficiencia** de un sistema informático:
  - Un solo usuario **no mantiene ocupado** el procesador y los dispositivos de E/S continuamente
  - La multiprogramación organiza los trabajos de tal forma que el procesador se **mantiene trabajando** el máximo tiempo posible
  - Cuando un trabajo tiene que esperar (por una operación de I/O por ejemplo), el S.O. **pone a ejecutar otro** trabajo en el procesador.

- El **tiempo compartido** o **multitarea** es una extensión de la multiprogramación.
  - El S.O. cambia tan rápidamente de trabajo que el usuario puede **interactuar** con cada trabajo mientras se ejecuta
    - **Temporizador**. Planificación de la CPU
  - El tiempo compartido permite también que haya **más de un usuario** utilizando el sistema

## Procesos

- Un proceso es en esencia un **programa en ejecución**.
- Cada proceso tiene asociado un **espacio de direcciones**, es decir, una lista de ubicaciones en memoria donde el proceso puede leer y escribir información.
- Cada proceso tiene asociado un conjunto de recursos que generalmente incluye **registros**, una lista de **archivos** abiertos, **alarmas** pendientes, **listas** de procesos relacionados y demás información necesaria para que se ejecute.

## Procesos

- Procesos en sistemas de **multiprogramación**. Por ejemplo: editar un vídeo, en la espera de la conversión navegar en la Web, y mientras tanto, otro programa comprueba si hay nuevos emails.
- Cada cierto tiempo el S.O. decide **detener** la ejecución de un proceso y **empezar** a ejecutar otro según determinados criterios.

## Procesos

- Cuando un proceso se **suspende** temporalmente, debe **reiniciarse** después exactamente en el mismo estado que se detuvo.
- Toda la información sobre el proceso debe **guardarse** de forma explícita. En muchos S.O. esto se guarda en una tabla llamada **tabla de procesos**.
- De este modo, un **proceso** (**suspendido**) consiste en su espacio de direcciones, conocido como imagen de núcleo.

# Procesos

- Lo habitual es que un sistema tenga **varios** procesos, algunos de usuario y otros del S.O., ejecutándose de forma **concurrente**.
- El S.O. es responsable de:
  - **Crear** y **eliminar** los procesos
  - **Suspender** y **reanudar** los procesos
  - Proporcionar mecanismos para la **comunicación** y **sincronización** de procesos
  - Proporcionar mecanismos para el tratamiento de los **interbloqueos**

## Gestión de memoria

- Cada computadora cuenta con una **memoria principal** que se utiliza para mantener los programas en ejecución.
- Los S.O. actuales permiten colocar **varios** programas en memoria al **mismo tiempo**.
- Para evitar que **interfieran** unos con otros y con el S.O., en el hardware hay mecanismos de protección controlados por el mismo S.O.

# Gestión de memoria

- El S.O. se encarga de:
  - Controlar **qué** partes de la memoria están en uso y por parte de **quién**
  - Decidir qué datos y procesos **añadir** o **quitar** de la memoria
  - **Asignar** y **liberar** espacio de memoria según sea necesario

## Espacio de direcciones

- En muchas computadoras las direcciones son de 32 o 64 bits, con lo cual se obtiene un espacio de direcciones de $\mathbf{2^{32}}$ (4.294.967.296)o $\mathbf{2^{64}}$ bytes, respectivamente.
- Las direcciones de memoria se expresan a menudo en **hexadecimal**.
- ¿Qué pasa si un proceso tiene más espacio de direcciones que la memoria principal de la computadora? -> **Memoria virtual**: una parte en la memoria principal y otra parte en el disco.

# Sistemas de numeración – Espacio de direcciones

- Ejercicio. Las direcciones de memoria se expresan a menudo en hexadecimal. Realizar la conversión a hexadecimal de los siguientes números binarios:
  a) 10011010100010100101
  b) 10100111111000011111100
  c) 11101001010110110001101 00

Soluciones:

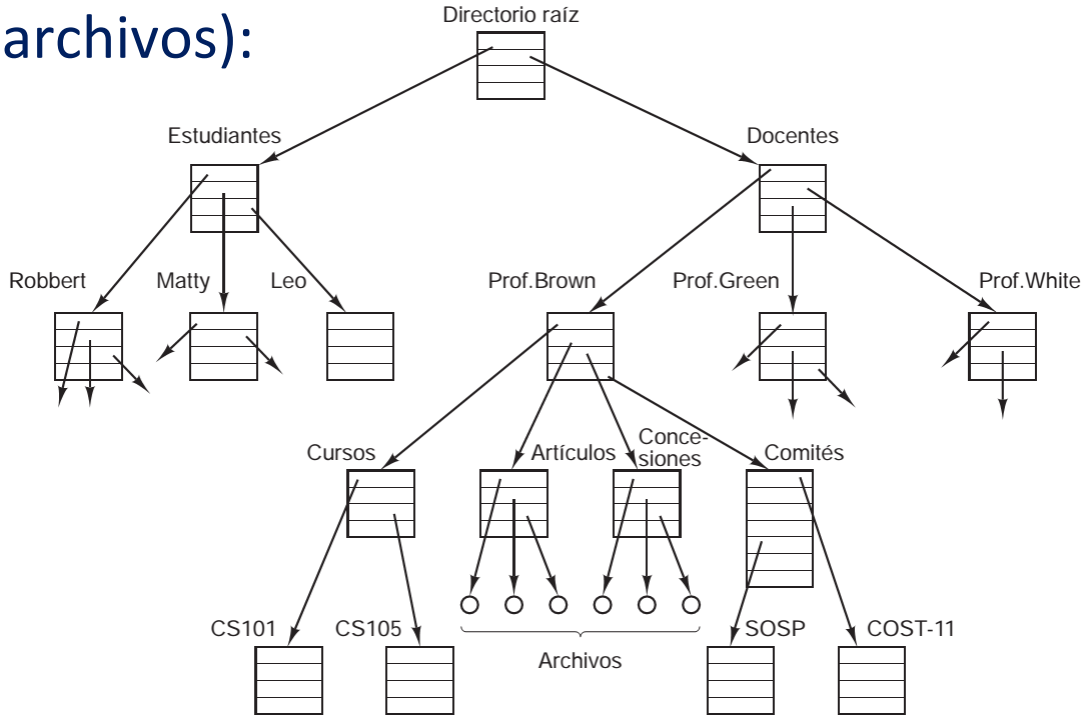
a) 9A8A5
b) A7E1FC
c) 1D2B634

# Archivos

- Otro concepto **clave** en los S.O.
- Se requieren **llamadas al sistema** para crear archivos, eliminarlos, leerlos y escribir en ellos.
- Por ejemplo, antes de leer un archivo es necesario localizarlo en el disco, y una vez leído, debe cerrarse, lo que implica diferentes llamadas al sistema.

# Archivos

- La mayoría de los S.O. cuentan con el concepto de un **directorio** como una forma de agrupar archivos, lo que conlleva a una **jerarquía** (el sistema de archivos):



# Archivos

- Para especificar cada archivo dentro de la jerarquía de directorio, se proporciona su **nombre de ruta**. Dichos nombres indican los directorios que se deben recorrer desde el **directorio raíz**:
  - /Docentes/Prof.Brown/Cursos/CS101
  - En **Windows** se utiliza la barra diagonal inversa (\), mientras que en **UNIX** se utiliza la barra diagonal (/)

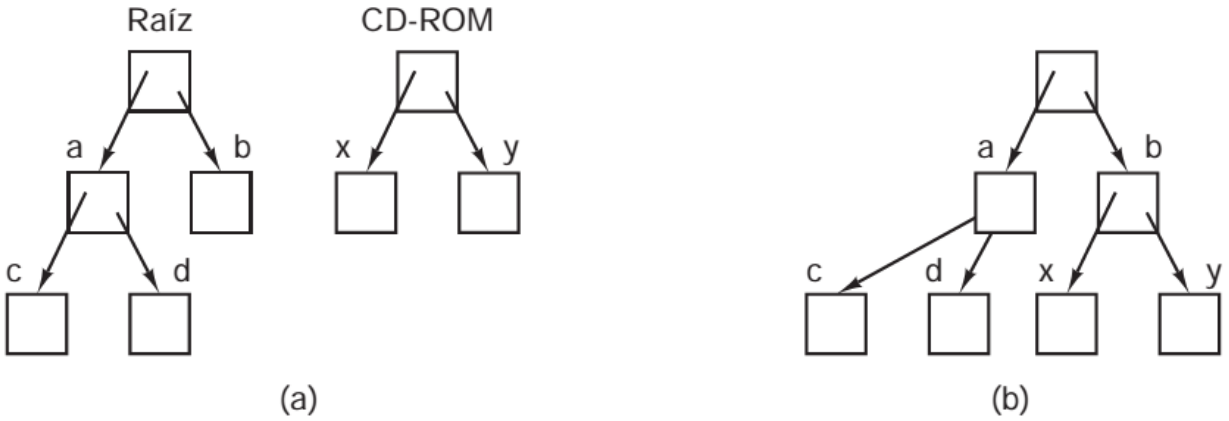
## Archivos

- Antes de poder leer o escribir en un archivo, se deben comprobar los **permisos** con el fin de verificar si está permitido el acceso.
- Si está permitido, el sistema devuelve un número entero conocido como **descriptor de archivo**. Si el acceso está prohibido, se devuelve un **código de error**.

# Archivos

- Otro concepto importante en UNIX es el sistema de **archivos montado**. CD-ROMs, DVDs, memorias USB, discos duros externos

- (a) Antes de montarse, los archivos del CD-ROM no están accesibles
- (b) Después de montarse, forman parte de la jerarquía de archivos



## Entrada/Salida (I/O)

- Todas las computadoras cuentan con dispositivos físicos para **adquirir entrada** y **producir salida**.
- Esto permite la **interacción** con los usuarios.
- Teclados, monitores, impresoras, etc.
- Es **responsabilidad** del S.O. administrar estos dispositivos.
- Cada S.O. tiene un **sub-sistema** de I/O para administrarlos.

## Protección

- Mucha información presente en las computadoras se suele **proteger** y mantener de manera **confidencial**.
- Es responsabilidad del S.O. administrar la **seguridad** de manera que los archivos, por ejemplo, sean accesibles solo para los usuarios **autorizados**.
- Existen otras cuestiones de seguridad como proteger el sistema de los **intrusos** no deseados, tanto humanos como no humanos (**virus**).

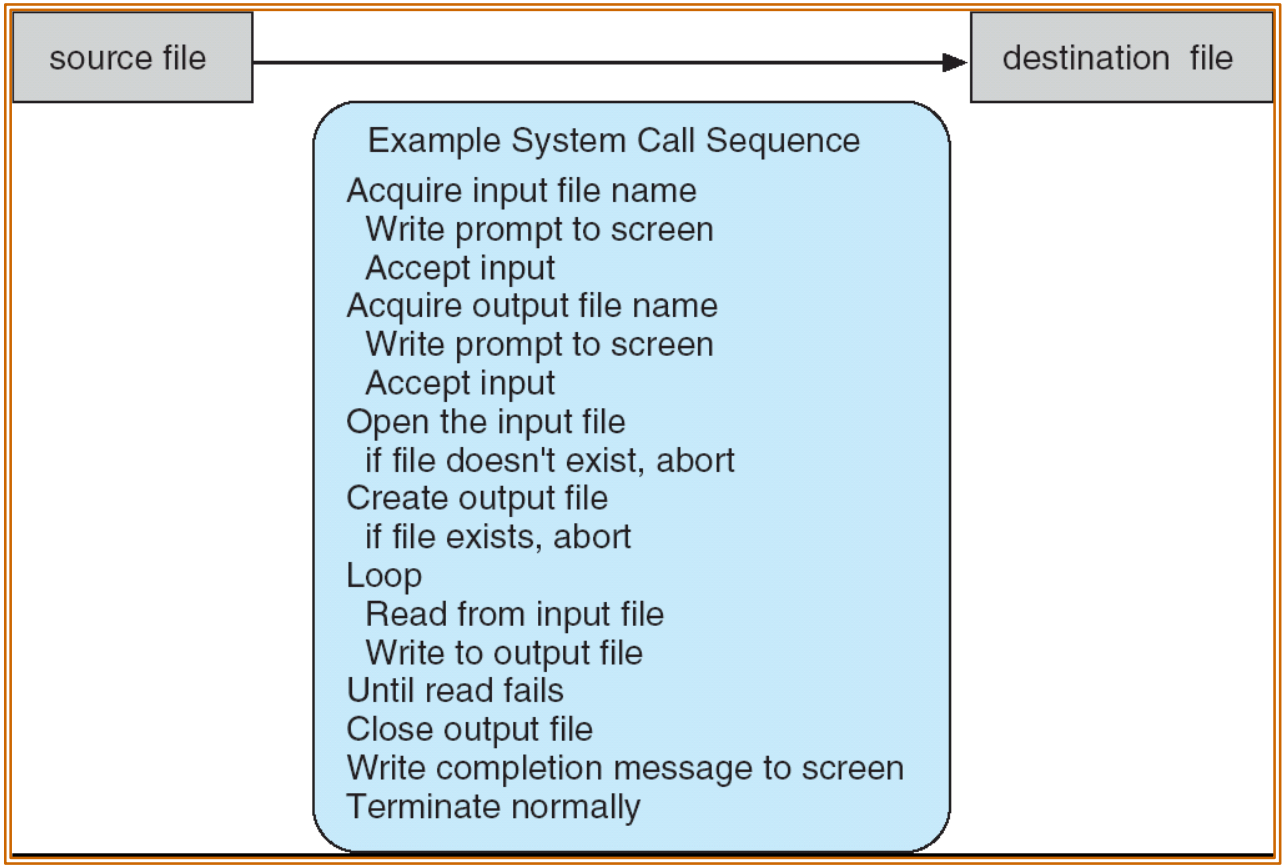
# Llamadas al Sistema

- Una **llamada al sistema** es un método o función que puede invocar un proceso para **solicitar un cierto servicio** al S.O.
- El S.O. actúa como **intermediario**, ofreciendo una **interfaz de programación** (API) que el programador puede usar en cualquier momento para solicitar recursos gestionados por el sistema operativo
- Normalmente están escritas en lenguaje **C** o **C++**
- Las APIs más comunes son **Win32** en Windows y **POSIX** en UNIX

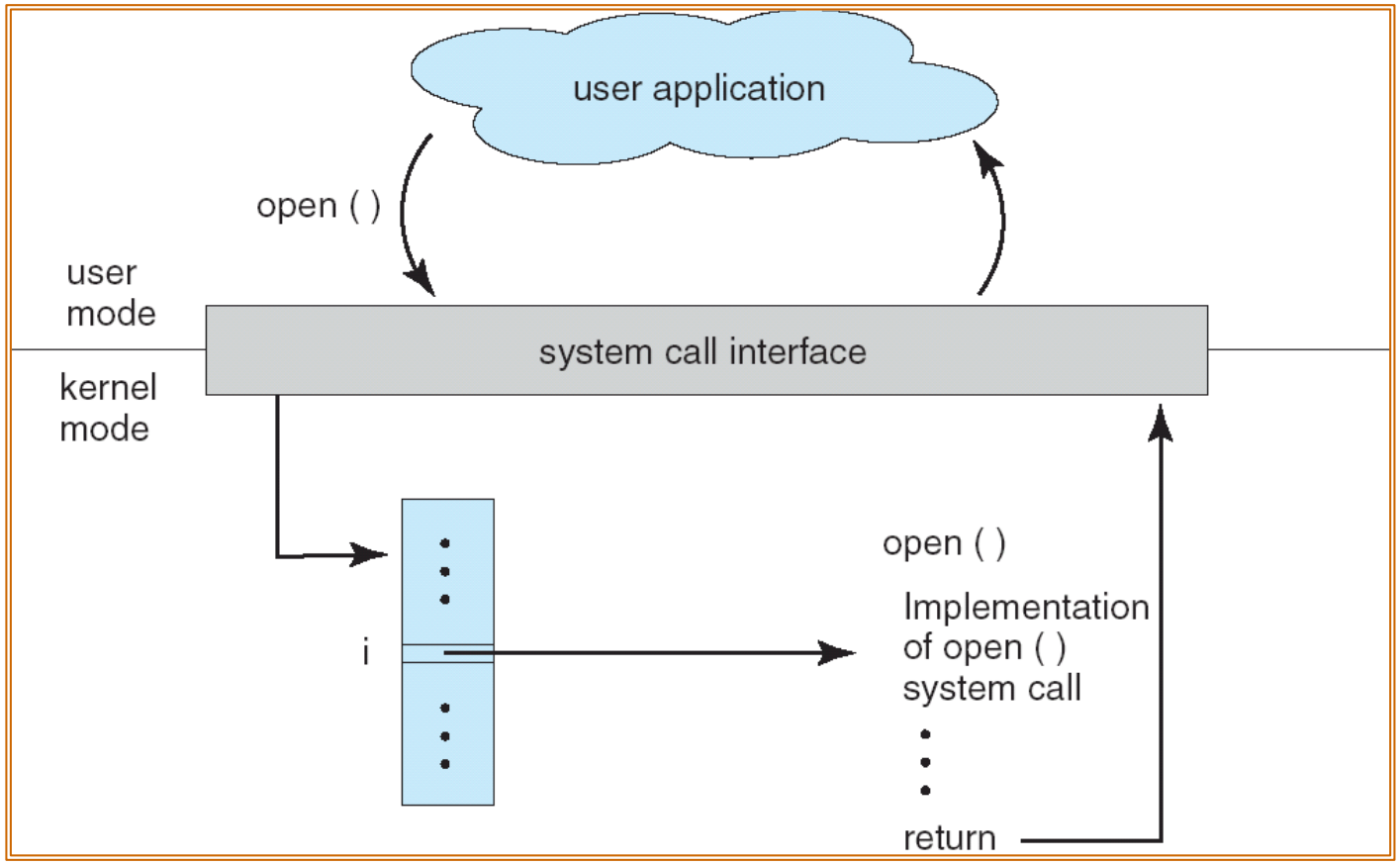
- Algunos ejemplos de llamadas al sistema:
  - **time**: permite obtener la fecha y hora del sistema
  - **write**: escribe un dato en un cierto dispositivo de salida
  - **read**: lee desde un dispositivo de entrada
  - **open**: obtiene un descriptor de un fichero del sistema
- Por ejemplo Linux 3.0 ofrece más de 340 llamadas al sistema.

# Ejemplo: secuencia de llamadas al sistema para copiar un archivo

- El programador no necesita saber cómo está implementada la llamada al sistema, sólo necesita utilizar la API y entender qué es lo que hará el S.O. como resultado de la llamada



# Llamadas al Sistema



- Tipos de llamadas al sistema. Colección de programas que proporcionan un entorno cómodo para desarrollar y ejecutar programas:
  - **Control de procesos**
    - Fin, abortar, cargar, ejecutar, crear, finalizar, obtener y establecer atributos, esperar, asignar y liberar memoria
  - **Gestión de archivos**
    - Crear, eliminar, abrir, cerrar, leer, escribir, reposicionar, obtener y establecer atributos

- Tipos de llamadas al sistema:
  - **Gestión de dispositivos**
    - Solicitar, liberar, leer, escribir, reposicionar, obtener y establecer atributos, conectar y desconectar dispositivos
  - **Mantenimiento de información**
    - Obtener y establecer hora, fecha, datos del sistema, atributos de un proceso, archivo o dispositivo
  - **Comunicaciones**
    - Crear, eliminar conexiones, enviar y recibir mensajes, …

# Estructura de un S.O.

- Hasta ahora hemos visto aspectos de los S.O. relacionados con su apariencia exterior (interfaz del programador).
- El interior de un sistema operativo puede tener diferentes diseños que se han probado en la práctica:
  - Sistemas monolíticos, sistemas de capas, microkernels, sistemas cliente-servidor, máquinas virtuales, exokernels…

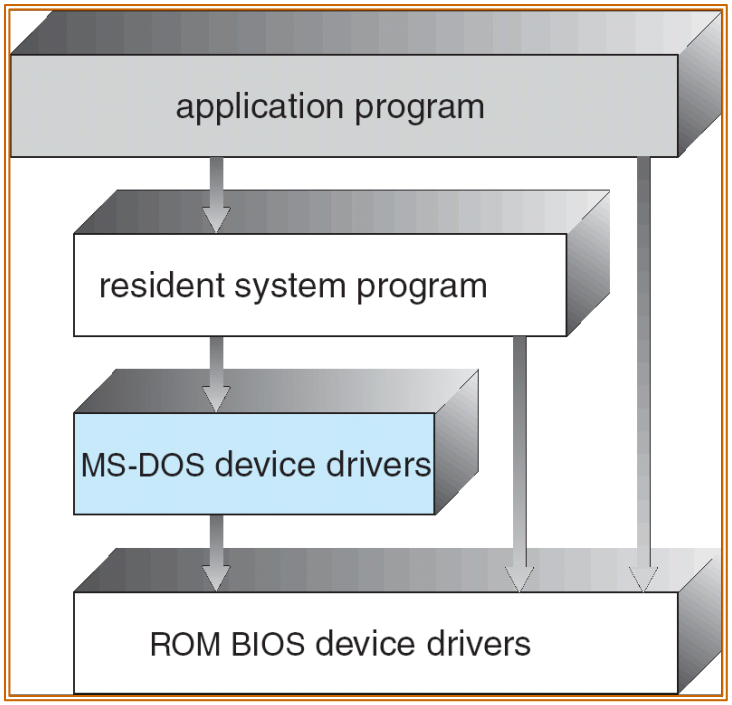
# Estructura de un S.O.

- La estructura interna del S.O. **varía** mucho de un sistema a otro.
- Depende de la **elección** del hardware y del tipo de sistema.
- **Objetivos**:
  - Del **usuario**: cómodo de usar, fácil de aprender, fiable, seguro, rápido.
  - Del **sistema**: fácil de diseñar e implementar, flexible, libre de errores y eficiente.

## Sistemas monolíticos

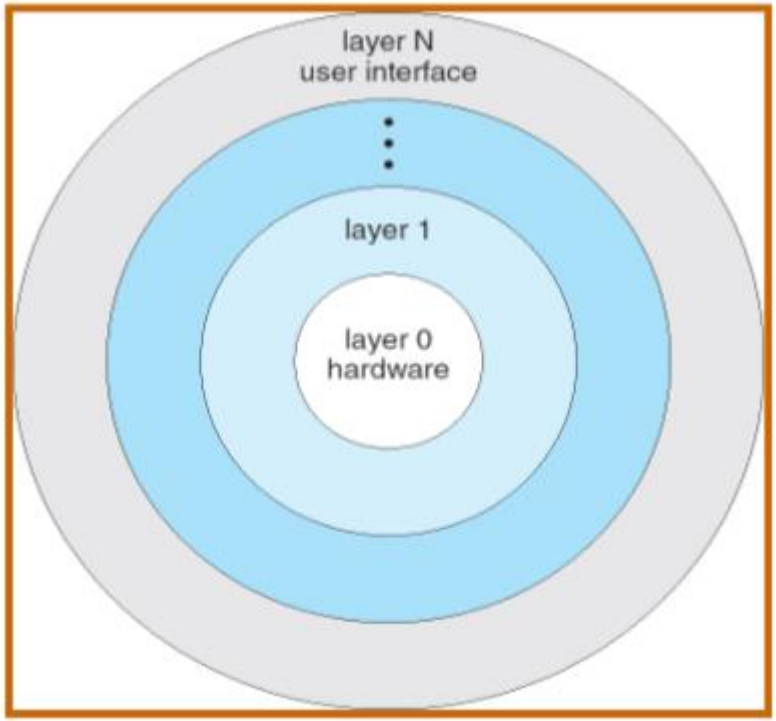
- No se tiene una estructura **definida**
- El sistema es escrito como una colección de procedimientos que pueden ser **invocados** por cualquier otro
- No existe "**ocultación de información**"
- Todo procedimiento es **público** y accesible a cualquiera, aunque es posible tener buenos diseños y obtener una buena eficiencia

- Sistemas monolíticos. Ej: MS-DOS (y Linux)



# Sistemas de capas

- Se organiza el diseño en una jerarquía de **capas** construidas una encima de otra.
- Cada capa sólo utiliza funciones de la capa **inferior**.
- Difícil determinar **qué** tiene que ir en **cada** capa.
- La **capa 0** es el hardware y la **N** es la de procesos de usuario.

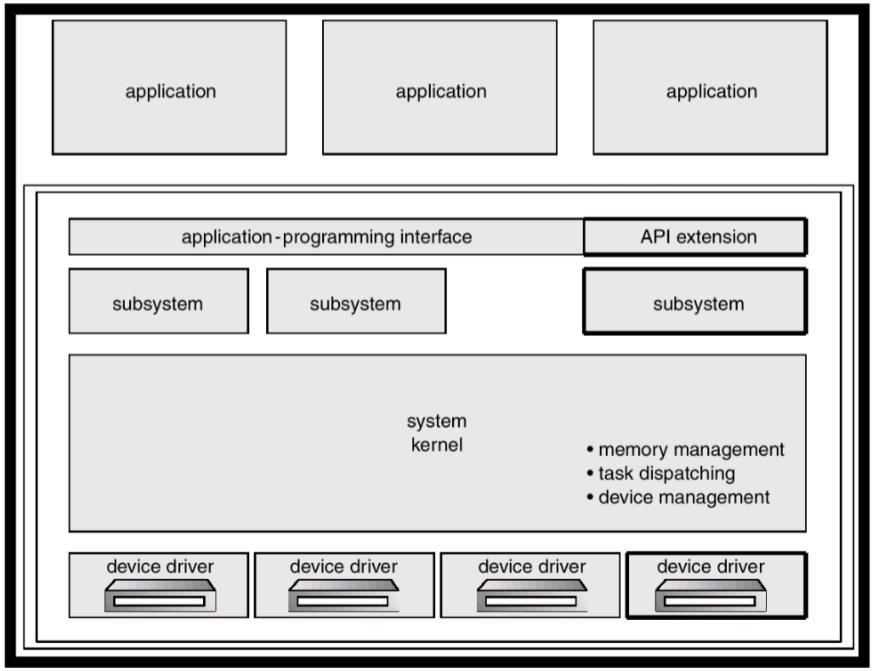


# Sistemas de capas

- Ventajas:
  - Modularidad
  - Depuración y verificación de cada capa por separado
- Desventajas:
  - Alto coste de definición de cada capa en la etapa de diseño
  - Menos eficiente respecto al sistema monolítico debido al **overhead** al pasar por cada capa

# Sistemas de capas

- Ej: OS/2 (IBM)



## Microkernels

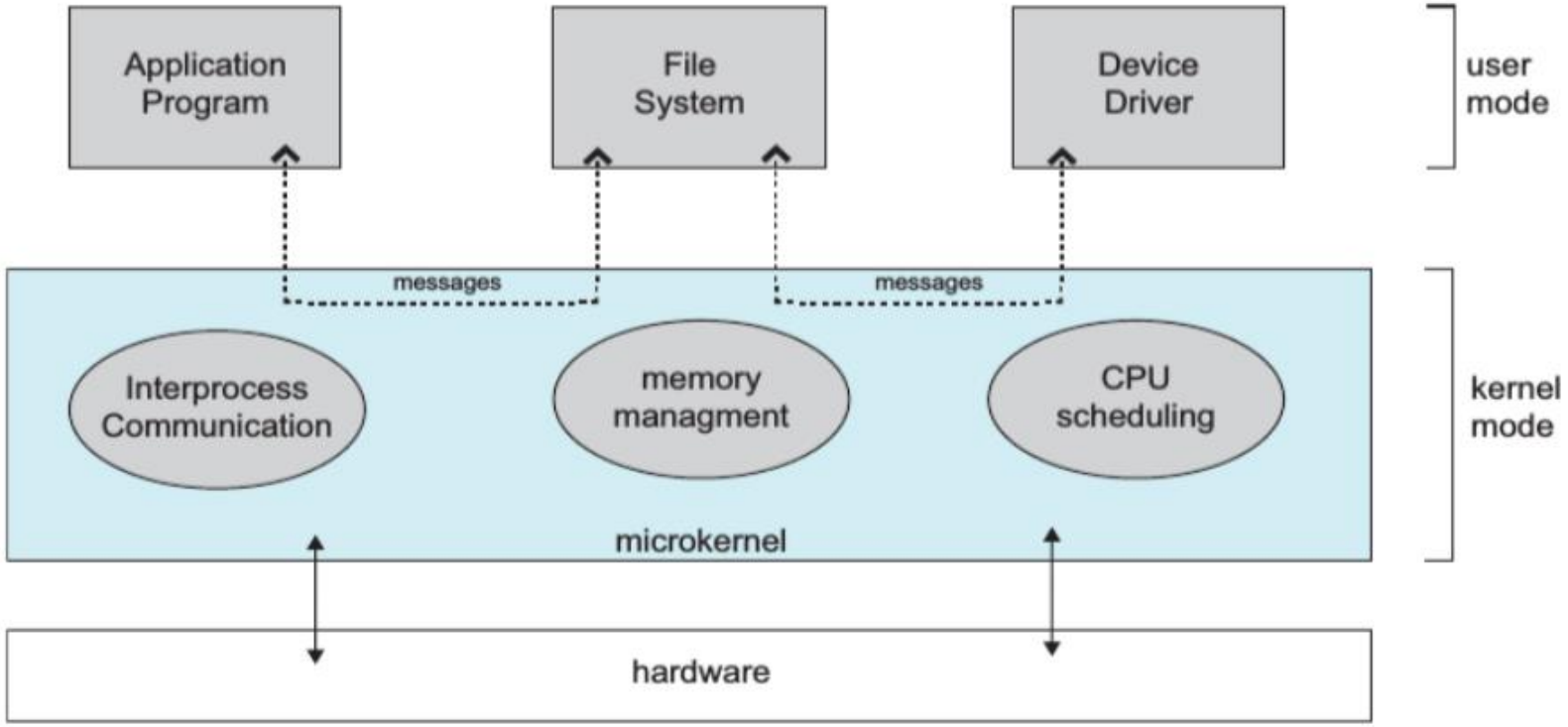
- Consta de un núcleo que proporciona un **manejo mínimo de procesos**, memoria y una capa de comunicación entre procesos.
- La **capa de comunicación** es la funcionalidad principal del sistema.
- Los demás servicios son construidos como **procesos separados** al microkernel que ejecutan en modo usuario.
- El acceso a los **servicios** del sistema se realiza a través del paso de mensajes.

# Microkernels

- **Ventajas**:
  - Más fácil de ampliar y de portar a nuevas arquitecturas
  - No es necesario modificar el núcleo para incorporar un nuevo servicio
  - Es más seguro debido a que los servicios se ejecutan en modo usuario
  - El diseño es simple y funcional, lo que típicamente conlleva a tener un sistema más confiable.
- **Desventajas**:
  - Sobrecarga en las comunicaciones entre el espacio de usuario y el kernel

# Microkernels

- Ej: Symbian (Nokia) – Windows 10



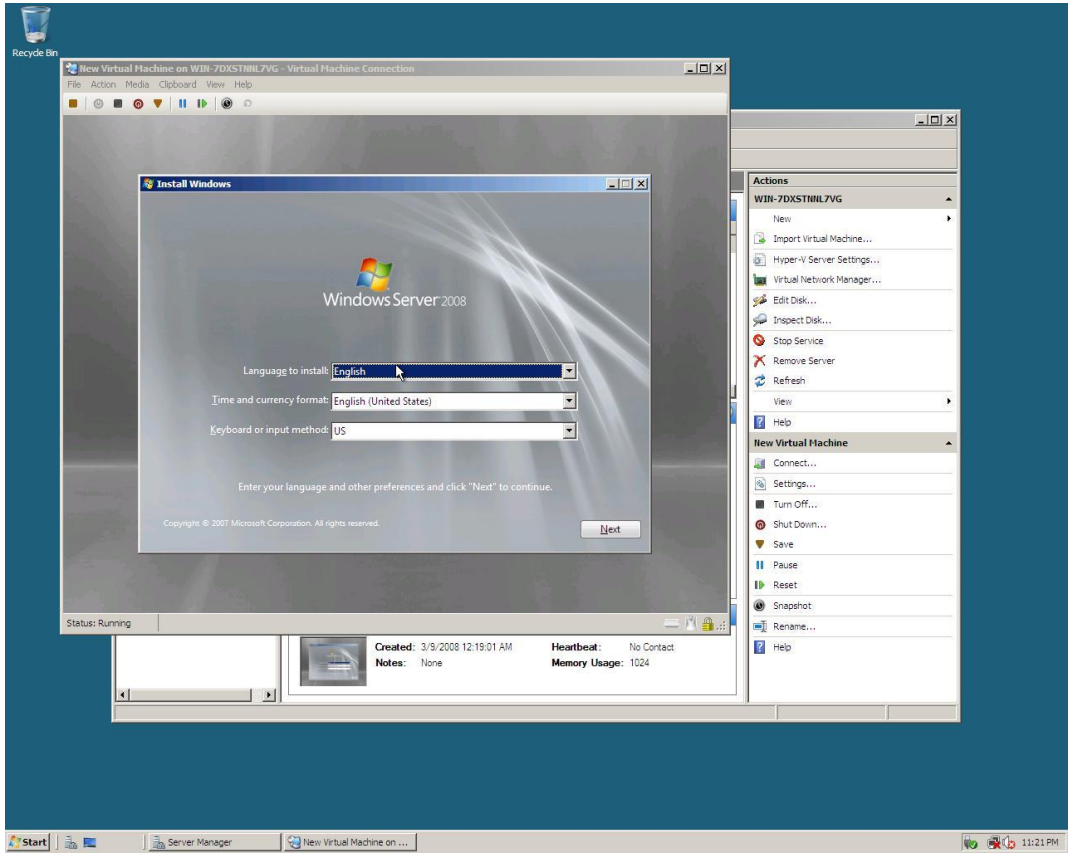
## Máquinas virtuales

- Pueden ser vistas como **extensiones** de los sistemas multiprogramados pero a más bajo nivel.
- Los procesos no solamente trabajan sobre el S.O. como si fueran el único proceso en el sistema sino que tienen una **copia virtual** del hardware de la CPU
- Las máquinas virtuales corren como procesos a **nivel de usuario** y el administrador de MVs (hypervisor) implementa un modo usuario virtual y un modo administrador virtual

# Máquinas virtuales

- También se implementan **discos virtuales** sobre los discos reales para las máquinas virtuales
- Dos modos básicos
  - Tipo 1: el administrador corre directamente sobre el hardware
  - Tipo 2: el administrador corre como un proceso sobre un sistema operativo normal

- Máquinas virtuales
  - **Ventajas**:
    - Seguridad
    - Facilidad de desarrollo
    - Flexibilidad
    - Alta disponibilidad
  - **Desventajas**
    - Más lento que un sistema real



## Modelo cliente-servidor

- Puede ser ejecutado en la **mayoría de las computadoras**, ya sean grandes o pequeñas.
- Sirve para **toda clase de aplicaciones**. Se considera de propósito general y cumple con las mismas actividades que un S.O. convencional.
- El núcleo tiene como misión **establecer la comunicación** entre los clientes (solicitan servicios) y los servidores (proporcionan servicios).

# Ejercicios

1. Dados los números binarios 01001000 y 01000100, indica cuál es mayor. ¿es necesario convertir los números al sistema decimal para compararlos?
2. ¿Cuántos caracteres diferentes se pueden representar utilizando el sistema de numeración binario con tres dígitos? ¿Y con ocho dígitos?
3. Realiza las siguientes conversiones:
   - ¿Cuantos bytes son 1 GB?
   - ¿Cuantos MB son 10 GB?
   - ¿Cuántos GB son 1.000.000 MB?

# Ejercicios

4. ¿Cuántos Bytes caben en un disco de 210 MB? ¿y cuántos bits?
5. ¿Cuántos disquetes de 3 ½, de capacidad 1,44 Mb, podrías copiar en un disco de 2 Gb?
6. En el Código ASCII, el símbolo “ ? ”, escrito en binario, es 00111111. ¿Cómo se representa en el sistema hexadecimal? ¿y en el sistema decimal?

# Resumen

- Los S.O. se pueden ver desde **dos puntos de vista**:
  - Como **administradores de recursos**. La función del S.O. es administrar las diferentes partes del sistema de forma eficiente.
  - Como **máquinas extendidas**. La función del S.O. es proveer a los usuarios abstracciones que sean más convenientes de usar que la máquina actual.
- Los S.O. tienen una **larga historia**
  - Empezando desde los días en que reemplazaron al **operador**, hasta los sistemas modernos de **multiprogramación**.

# Resumen

- Los S.O. tienen una **larga historia**
  - Entre los puntos importantes tenemos a los sistemas de **procesamiento por lotes**, los sistemas de **multiprogramación** y los sistemas de **computadora personal**.
- Relación con el **hardware**
  - Dado que los S.O. **interactúan** de cerca con el hardware, es útil tener cierto conocimiento al respecto para comprenderlos.
  - Las computadores están **compuestas** de procesadores, memorias y dispositivos de I/O conectados mediante buses.

- Los conceptos básicos en los que se basa un S.O.
  - **Procesos**, administración de **memoria**, administración de **I/O**, sistema de **archivos** y **seguridad**.
- Corazón de cualquier S.O.
  - Es el conjunto de **llamadas al sistema** que puede manejar. Estas llamadas indican lo que realmente **hace** el S.O.
- Estructuración de los S.O.
  - Sistema monolítico, jerarquía de capas, microkernel, cliente-servidor, máquina virtual o exokernel.

# Sistemas Operativos y Redes

# Presentación de la asignatura

Service Science, Management and Engineering

Grado en Ciencia, Gestión e Ingeniería de Servicios

**Profesor:**

**David Granada**

david.granada@urjc.es