

# Introducción a los Sistemas Operativos

---

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

FACYT- UC

PROFESORA: PHD. MIRELLA HERRERA

# DEFINICIÓN DE SISTEMA OPERATIVO

# Definición de Sistema Operativo

---

## Algunas definiciones de Sistema Operativo

- ❑ “El software del que depende el resto del software para hacer el computador funcional”
- ❑ “El único programa que se ejecuta todo el tiempo en el computador”
- ❑ “Un programa que administra todos los otros programas en el computador”

# Definición de Sistema Operativo

---

## Dos funciones principales

### Administrar los recursos físicos

- ☐ Manejar varios dispositivos
  - Procesador, memoria, discos, redes, pantallas, cámaras, etc.
- ☐ De manera eficiente, confiable, tolerando y enmascarando las fallas, etc.

### Proveer un ambiente de ejecución para las aplicaciones corriendo sobre el computador (programas como Word, Emacs, etc.)

- ☐ Provee recursos virtuales y sus interfaces
  - Archivos, directorios, threads, procesos, etc.
- ☐ Simplifica la programación mediante abstracciones de alto nivel
- ☐ Provee al usuario con un ambiente estable



# Definición de Sistema Operativo

---

- ❑ Ejemplos: Unix, Linux, MS-DOS, Windows, FreeBSD, etc.
- ❑ El intérprete de órdenes, los sistemas de ventanas, los editores, los compiladores, etc., son **programas del sistema, NO son el S.O.**
- ❑ El S.O. se ejecuta en **modo núcleo**
- ❑ Los programas del sistema se ejecutan en **modo usuario**

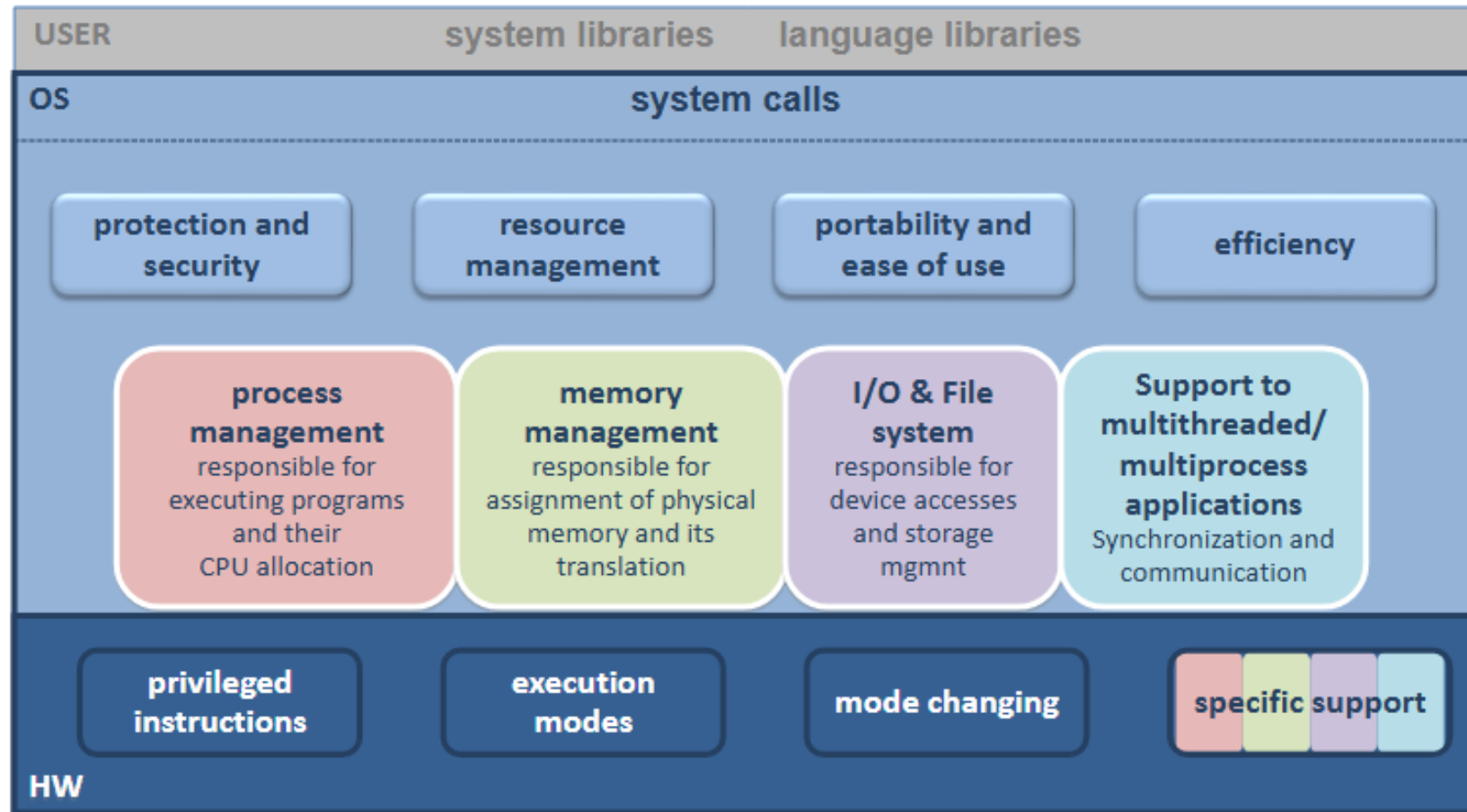
# Gestor de Recursos

---

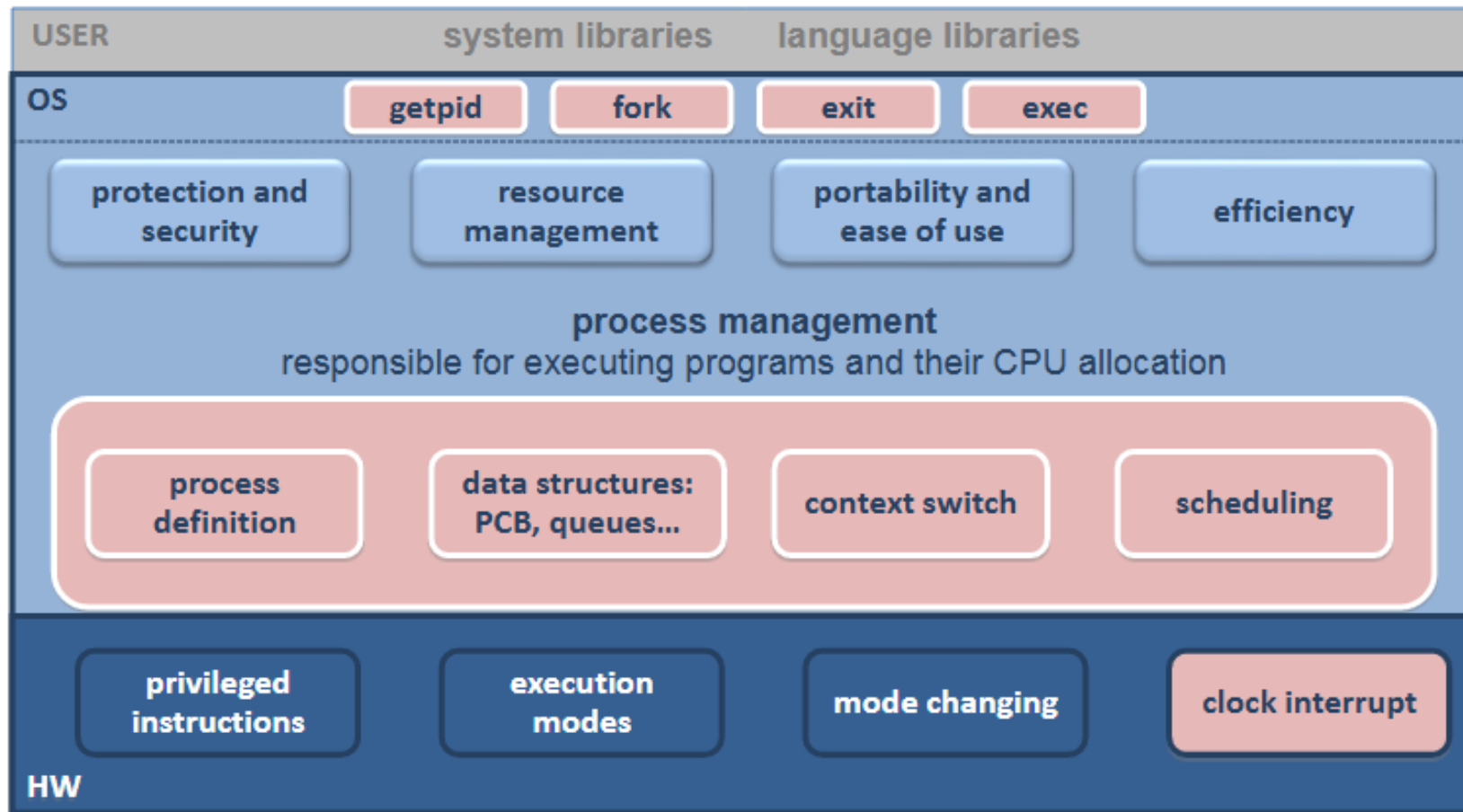
## Visión: Gestor de los diferentes recursos del sistema

- ☐ Gestor del Procesador y procesos
- ☐ Detección de fallas e interbloqueos
- ☐ Gestor de la Memoria
- ☐ Gestor de dispositivos de E/S
- ☐ Gestor de Información
- ☐ Protección y Seguridad
- ☐ Compartición

# Gestor de Recursos

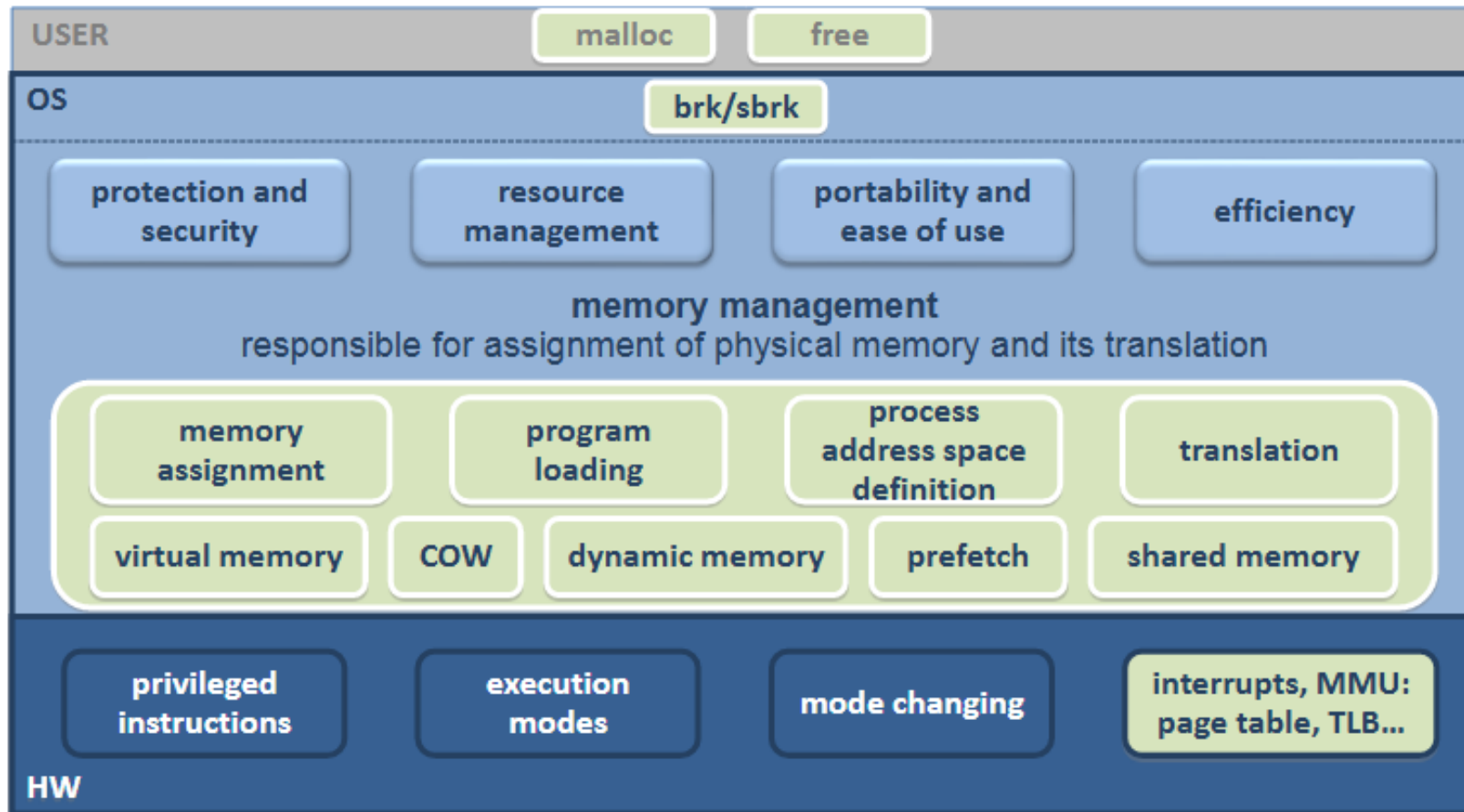


# Gestor de Recursos

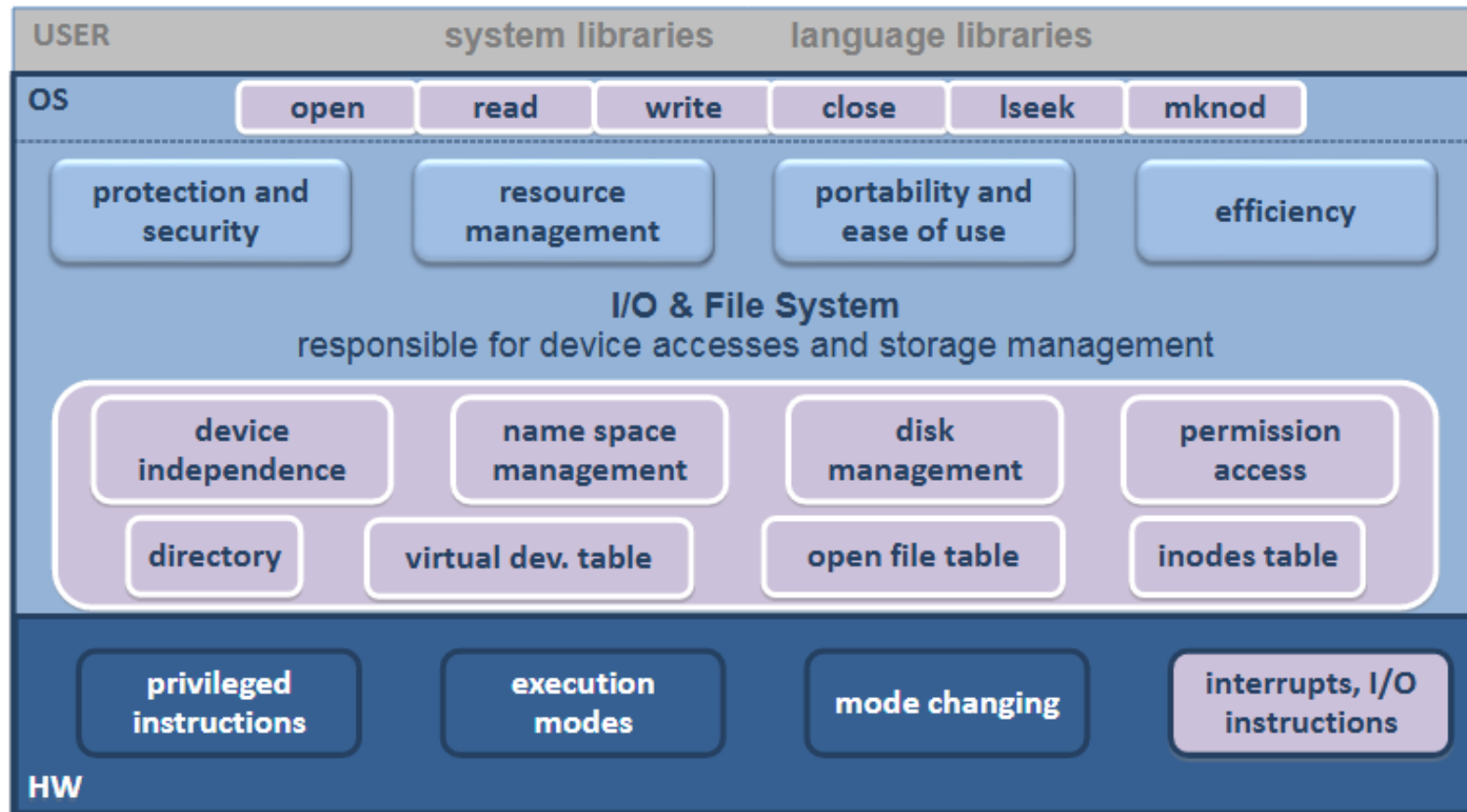




# Gestor de Recursos

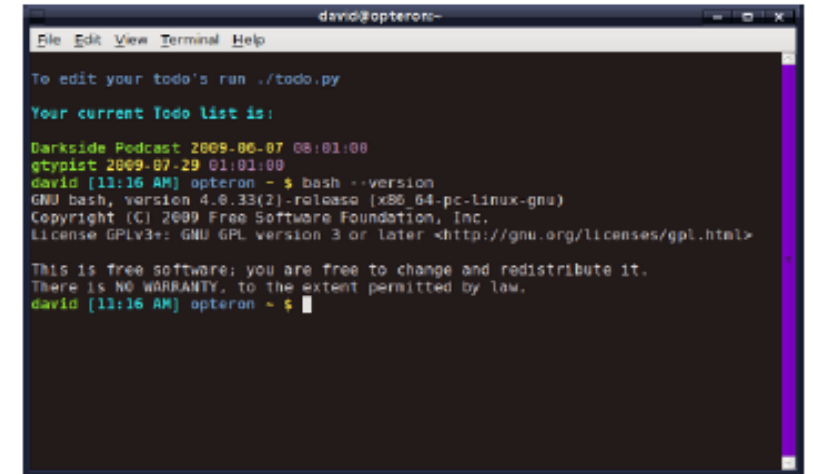


# Gestor de Recursos



# Entorno de trabajo del SO

- ❑ El SO suele proporcionar utilidades básicas para que el usuario pueda realizar tareas comunes:
  - Trabajar con archivos y discos
  - Ejecutar aplicaciones = cargador de programas
  - Imprimir
  - Administrar el sistema: backups, usuarios...
  
- ❑ CLI = *Command Line Interface*
  - Incorpora un lenguaje sencillo para dar instrucciones al SO:
    - Cargar programas, trabajar con archivos, etc.
  - Se le llama shell, intérprete de órdenes, consola...



```
david@opteron-  
File Edit View Terminal Help  
To edit your todo's run ./todo.py  
Your current todo list is:  
Darkside Podcast 2009-06-07 08:01:00  
gtypist 2009-07-29 01:01:00  
david [11:16 AM] opteron - $ bash --version  
GNU bash, version 4.0.33(2)-release (x86_64-pc-linux-gnu)  
Copyright (C) 2009 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
  
This is free software; you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
david [11:16 AM] opteron - $
```

# Entorno de trabajo del SO

□ *Graphical User Interface (GUI)*

□ API del SO: llamadas al sistema

- API = *Application Programming Interface*
- El SO ofrece a los desarrolladores y a los procesos un conjunto de servicios públicos, accesibles mediante una API = llamadas al sistema (systemcalls)
  - Llamada write() de UNIX. Escribe un bloque de datos en un fichero o en un dispositivo de E/S.



```
int write ( int fd, void* buffer, size_t size )
```

↑  
Resultado:  
número de bytes  
escritos

↑  
Descriptor de fichero  
(identifica el fichero  
donde vamos a escribir)

↑  
Apuntador a la zona  
de memoria donde  
están los datos

↑  
Longitud de los  
datos en bytes

# ELEMENTOS DE DISEÑO

# Elementos de Diseño del S.O.

---

**Estructura:** ¿Cómo se organiza el SO?

**Compartir:** ¿Cómo se comparten los recursos entre los usuarios?

**Nombres:** ¿Cómo denominan los usuarios y/o programas a los recursos?

**Protección:** ¿Cómo se protege un usuario/programa de los demás?

**Seguridad:** ¿Cómo autenticar, controlar el acceso y asegurar la privacidad?

**Rendimiento:** ¿Por qué el sistema es lento?

**Confiabilidad y Tolerancia a Fallas:** ¿Cómo manejamos las fallas?

**Extensibilidad:** ¿Cómo se agregan nuevas funcionalidades?

# Elementos de Diseño del S.O.

---

**Comunicación:** ¿Cómo podemos intercambiar información?

**Concurrencia:** ¿Cómo se crean y controlan las actividades paralelas?

**Escalabilidad, crecimiento:** ¿Qué sucede cuando se incrementan las demandas o los recursos?

**Persistencia:** ¿Cómo los datos pueden existir más allá del proceso que los crea?

**Compatibilidad:** ¿Cómo se pueden hacer cosas nuevas?

**Distribución:** Acceder al mundo de la información

**Contabilidad:** ¿Quién paga las cuentas y como se controla el uso de los recursos?

# Ventajas de los S.O.

---

## **Provee niveles de abstracción**

- ☐ Pueden realizarse acciones simples como: “Escribir XYZ en el archivo ABC.txt que está en la carpeta /home/usuario”, en lugar de “Cargar el registro XFY con ID de segmento a HDD de tipo 0x4333,... etc”

## **Protege a los usuarios de otros usuarios y en ocasiones de sí mismos**

- ☐ Yo no puedo leer tus archivos, tu no puedes leer los míos

## **Comparte recursos eficientemente**

- ☐ Puede brindar al usuario la sensación que sólo sus aplicaciones se están ejecutando sobre la máquina



# Beneficios de los S.O.

---

¿Qué ganamos interponiendo esta interfaz entre los programas y el hardware?

- ❑ **Usabilidad** la interfaz es más cómoda que el hw
- ❑ **Seguridad** se ocultan vulnerabilidades del interior del hardware
- ❑ **Portabilidad** independencia del hardware
- ❑ **Interoperabilidad** podemos compartir información con otros sistemas que usen la misma interfaz
- ❑ **Mantenibilidad** podemos hacer mejoras o adaptaciones dentro del SO sin obligar a hacer cambios en los programas de usuario
- ❑ **Productividad** por todo lo anterior

# CONCEPTOS BÁSICOS

# Conceptos Básicos

---

## El SO Software reactivo

El SO no se activa por sí solo. Se activa cuando ocurre un evento que tiene que atender = comportamiento reactivo

### ☐ Tipos de eventos:

- Interrupciones del hardware
- Llamadas al sistema
- Excepciones

### ☐ Cada tipo de evento activa una RSI diferente, indexada a través del vector de interrupciones

# Conceptos Básicos

## Interrupciones

- ❑ Cuando llega una señal de interrupción a la CPU, ésta suspende lo que está haciendo y ejecuta una rutina de servicio de interrupción (RSI)
- ❑ Antes de ejecutar la RSI, hay que guardar el estado de la CPU, para que pueda reanudar lo que estaba haciendo después de completar la RSI

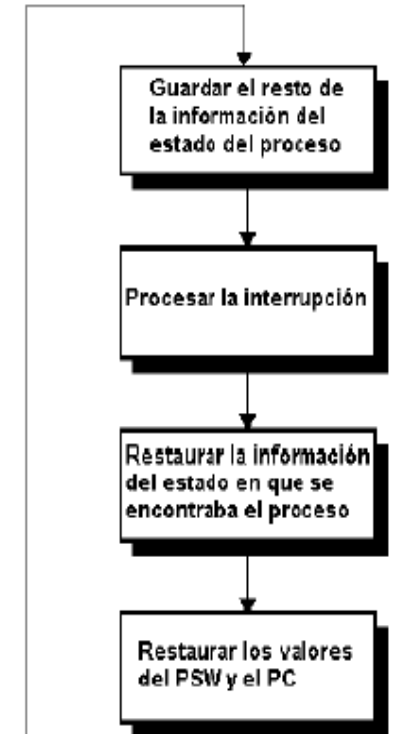
## ¿Cómo sabe la CPU qué dispositivo ha interrumpido?

- ❑ Método primitivo: preguntando a todos los dispositivos = polling
- ❑ Método avanzado: el dispositivo envía un número por el bus = interrupciones vectorizadas

## Hardware



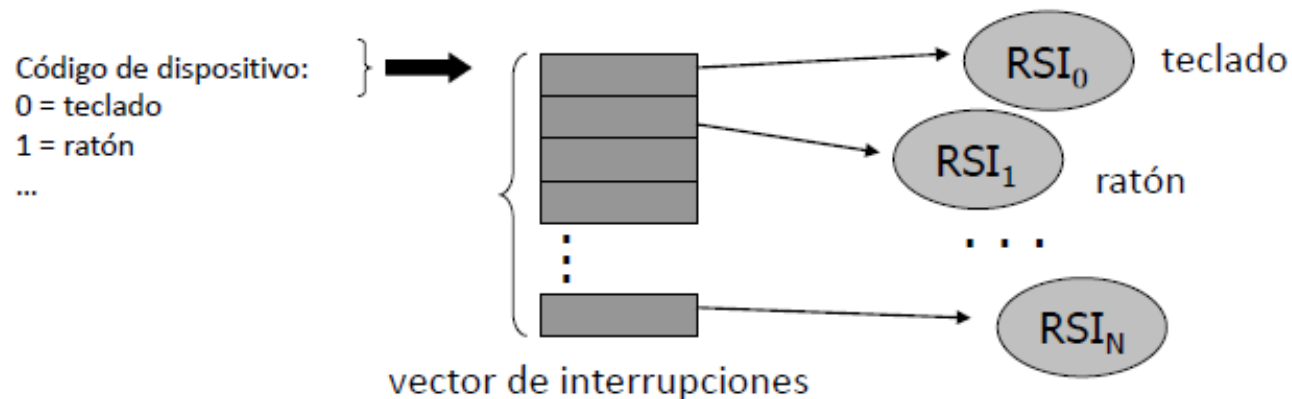
## Software



# Conceptos Básicos

## Sistema de Interrupciones Vectorizadas

- ❑ Vector de interrupciones: Una zona de la memoria principal contiene las direcciones de todas las RSI
- ❑ El dispositivo que interrumpe envía un número por el bus de datos. El número sirve de índice en el vector de interrupciones. La CPU ejecuta la RSI correspondiente



# Conceptos Básicos

---

## Sistema de Interrupciones de acuerdo a su origen:

### ❑ Interrupciones por Hardware

- Interrupciones de dispositivos de E/S: producidas por dispositivos de E/S como el teclado, el ratón y el disco duro. El SO detiene la ejecución normal del programa y atiende la interrupción de E/S antes de devolver el control al programa
- Excepciones de hardware: errores de hardware como la falta de energía, fallos en la memoria o problemas en el bus de datos. El SO detiene la ejecución normal del programa y maneja la excepción de hardware antes de devolver el control al programa
- Interrupciones de reloj: El reloj del sistema genera interrupciones por hardware periódicas que se utilizan para sincronizar el sistema operativo y planificar la ejecución de los procesos
- Interrupciones de señalización de errores: Los dispositivos de hardware notifican al SO de errores de memoria o errores de la unidad central de procesamiento (CPU)

# Conceptos Básicos

---

## Sistema de Interrupciones de acuerdo a su origen:

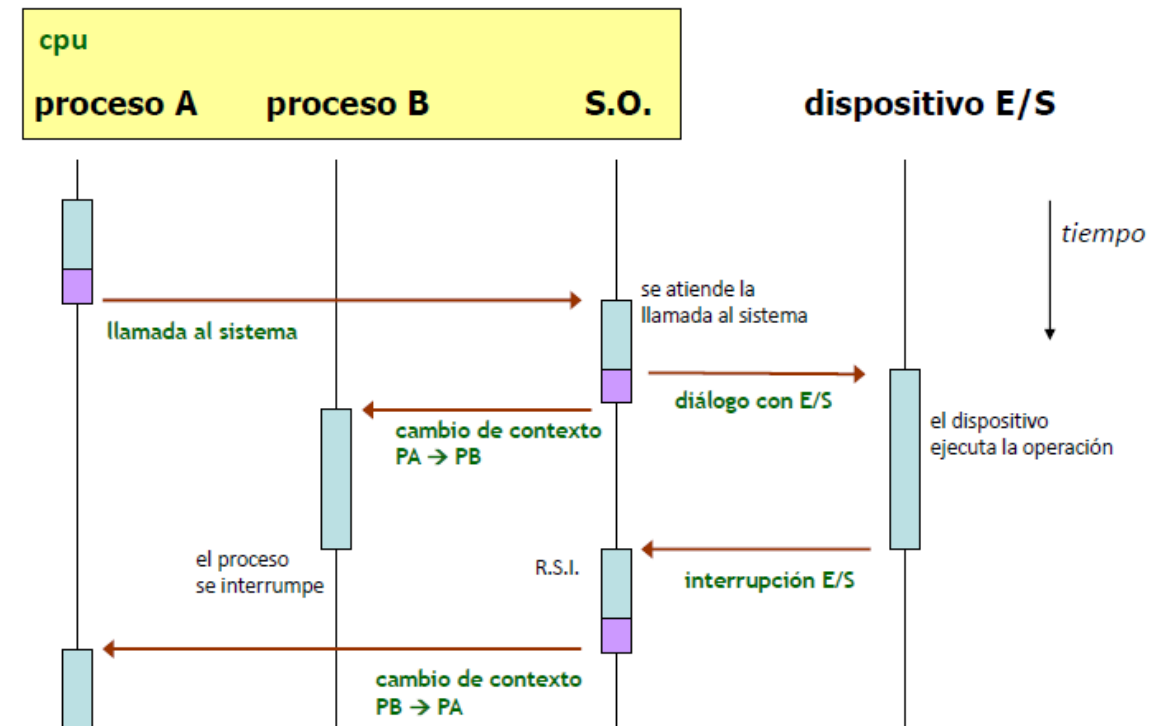
### ❑ Interrupciones software:

- Llamadas al sistema (system calls): en una llamada al sistema, un programa solicita al SO la creación de un archivo o la lectura de un archivo. El SO detiene la ejecución normal del programa, atiende la llamada al sistema y después devuelve el control al programa
- Excepciones: son interrupciones por software que se generan cuando ocurre una división por cero o un acceso a una dirección de memoria inválida. El SO detiene la ejecución normal del programa y maneja la excepción antes de devolver el control al programa
- Interrupciones de temporizador: son interrupciones por software que se generan cuando expira un temporizador definido por el SO. El SO detiene la ejecución normal del programa, atiende la interrupción de temporizador y después devuelve el control al programa
- Señales (signals): son interrupciones por software que se generan cuando un proceso recibe una notificación de otro proceso o del sistema operativo. Se utilizan para notificar a un proceso de eventos importantes, como la finalización o la interrupción de una operación

# Conceptos Básicos

## Sincronización entre E/S y SO

- ❑ El proceso A invoca una operación de E/S a través de una llamada al sistema
- ❑ El SO atiende la llamada y dialoga con la E/S
- ❑ Como el proceso A queda en espera por la E/S, el SO decide traer a CPU al proceso B
- ❑ Cuando la E/S finaliza, genera una interrupción
- ❑ El SO atiende la interrupción y decide reanudar la ejecución del proceso A





# Conceptos Básicos

---

## Protección del hardware

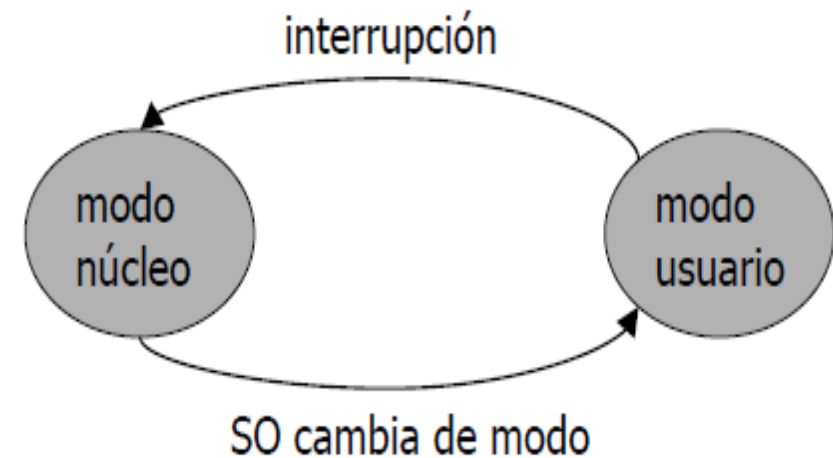
- ❑ Para que el S.O. funcione adecuadamente, hay que impedir que los programas de usuario puedan realizar libremente ciertas operaciones:
  - Acceso a la memoria del S.O. y de otros programas
  - Acceso directo a los dispositivos de E/S
  - Utilizar la CPU todo el tiempo que quieran
- ❑ Solución: modo dual de operación
  - La CPU define un repertorio de instrucciones privilegiadas.
  - Dos modos de operación del hardware:
    - Modo privilegiado/supervisor/sistema: Se pueden ejecutar todas las instrucciones.
    - Modo no privilegiado/usuario: Si se intenta ejecutar una instrucción privilegiada, la CPU interrumpe la ejecución y genera una excepción.

# Conceptos Básicos

---

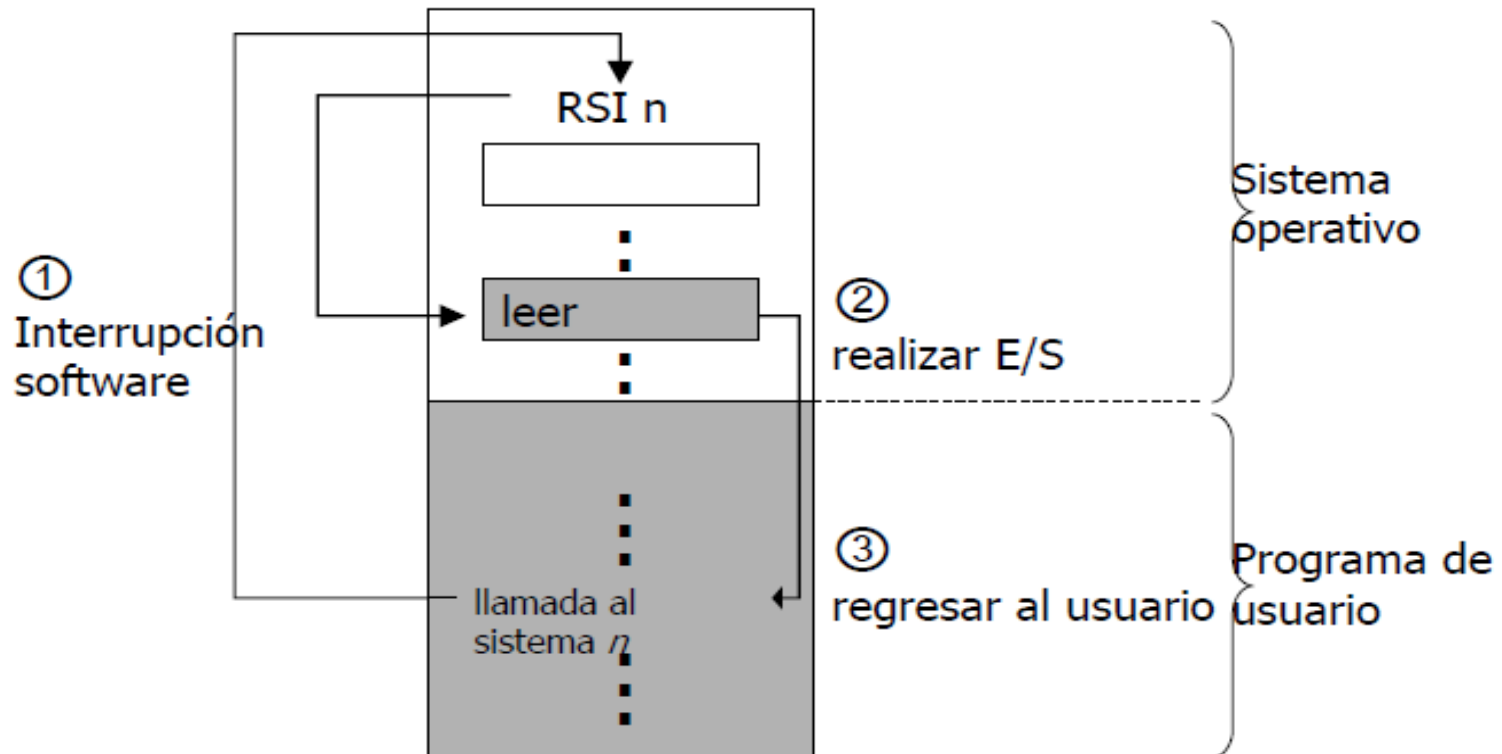
## Modo Dual de Operación: ¿Cuándo y cómo se cambia de modo?

- ❑ La CPU arranca en modo privilegiado.
- ❑ Cuando el S.O. cede el control al usuario, conmuta previamente a modo no privilegiado.
- ❑ Sólo se vuelve a modo privilegiado cuando el S.O. recupera el control, es decir, cuando ocurre una interrupción, una llamada al sistema o una excepción.



# Conceptos Básicos

Modo Dual de Operación: ¿Cuándo y cómo se cambia de modo?



# Conceptos Básicos

---

## Arranque típico de un SO

1. Cuando el equipo se enciende, la CPU inicia su ejecución en un punto fijo de la memoria
2. Hay una ROM con una pequeña rutina de arranque
3. La rutina localiza en qué dispositivo se encuentra el cargador del SO (boot loader) y lo carga en memoria
  - Nota: La ROM del equipo tiene código para leer y escribir sobre los dispositivos de E/S
4. El cargador instala el núcleo y se continúa el proceso de carga de módulos, servicios, etc. hasta que el SO queda totalmente operativo.

# ESTRUCTURA DE SISTEMAS OPERATIVOS

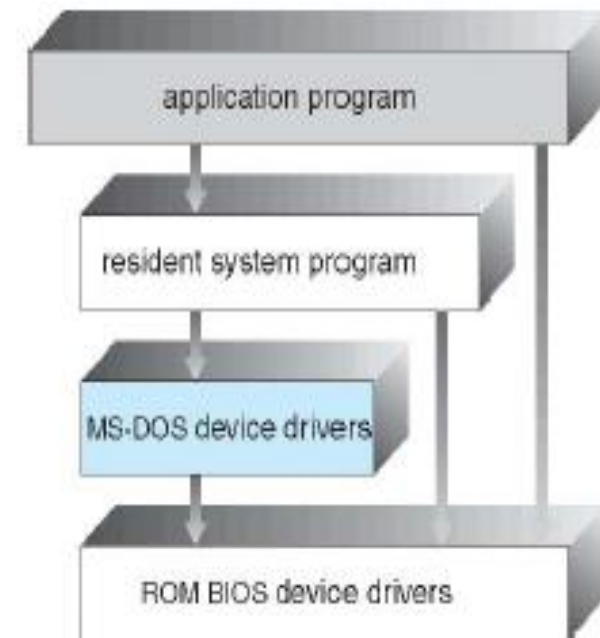
# Estructura de los Sistemas Operativos

## ¿Qué estructura interna tiene un SO?

Los SO de propósito general son programas muy extensos

### Estructura simple o monolítica: MS-DOS

- ❑ Escrito para proveer la mayor cantidad de funcionalidades en el menor espacio posible. No está dividido en módulos
- ❑ Aunque MS-DOS posee una estructura, sus interfaces y funcionalidades no están claramente separadas. Dispositivos de mano: móviles y tabletas



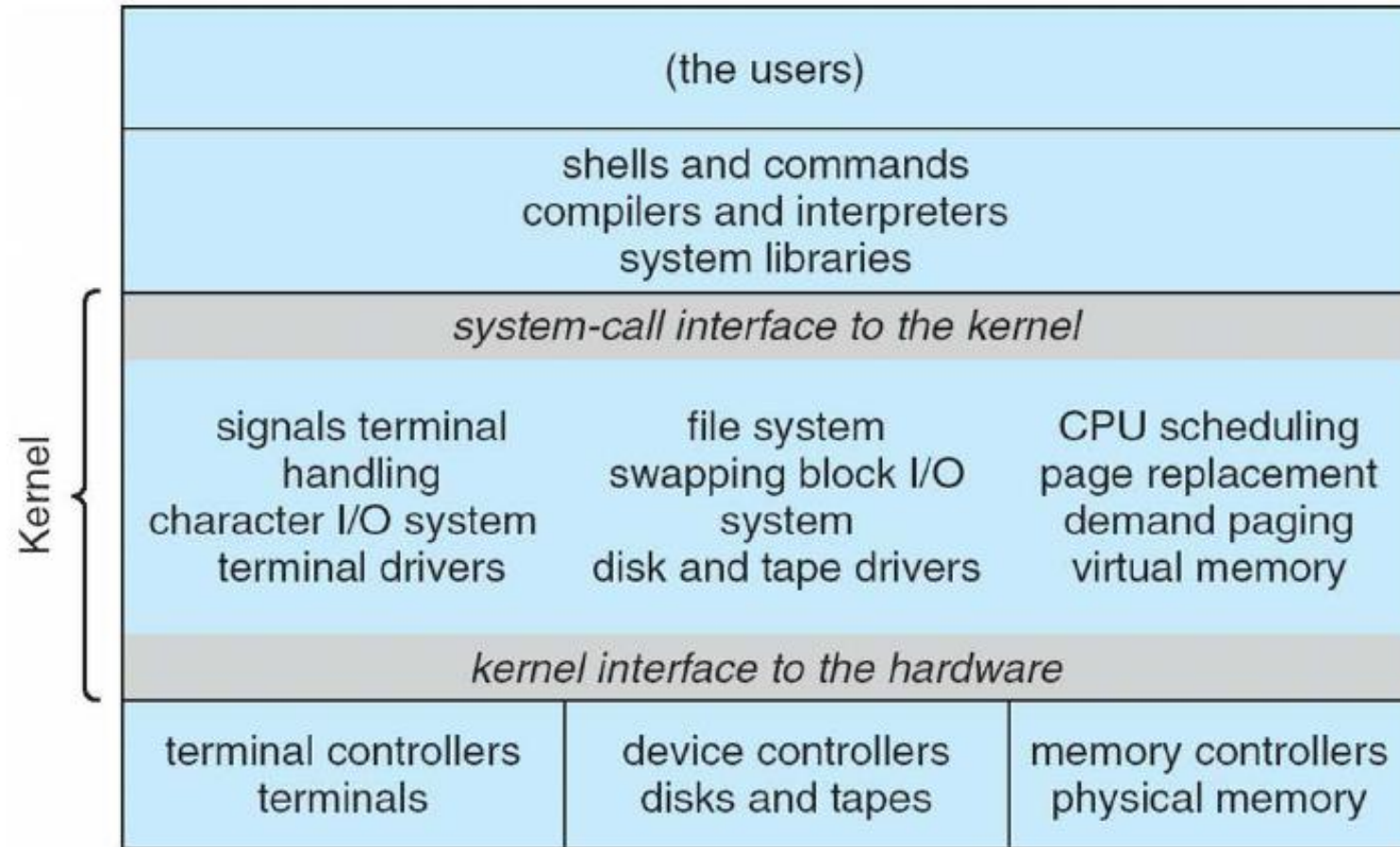
# Estructura de los Sistemas Operativos

---

## UNIX

- ❑ Limitado por las funcionalidades de hardware, el UNIX original poseía una estructura limitada
- ❑ El SO UNIX se estructuraba en dos partes separadas:
  - Los programas de sistema
  - El *kernel*
    - Consiste de todo aquello que resida por debajo de la interfaz de llamadas al sistema y por encima del hardware
    - Provee sistemas de archivos, planificación de CPU, manejo de memoria, y otras funciones del SO; de hecho una gran cantidad de funcionalidades para un solo nivel

# Estructura de los Sistemas Operativos



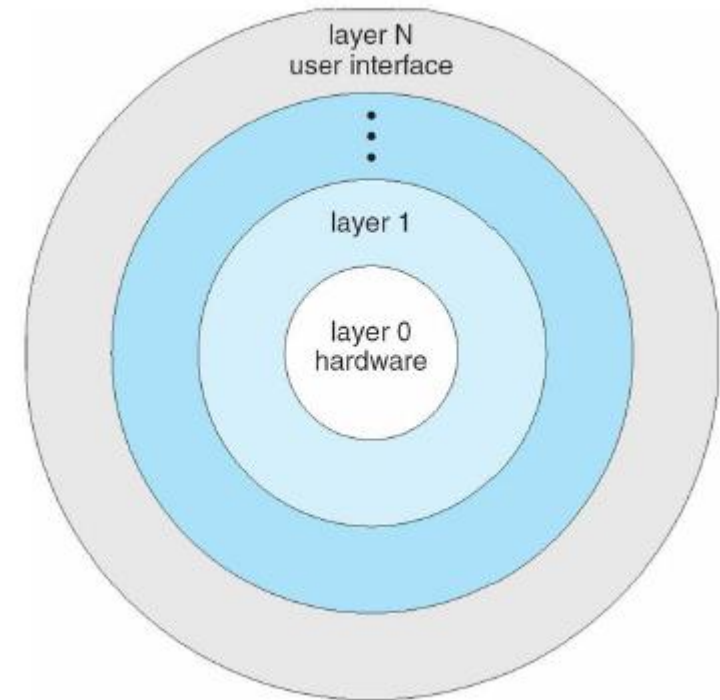


# Estructura de los Sistemas Operativos

---

## Por Capas

El SO es dividido en un número de capas (niveles), cada una encima de la otra. La capa más baja (nivel 0) es el hardware, mientras que la capa más alta (nivel n) es la interfaz de usuario. Haciendo uso de la modularidad producto de este enfoque, las capas son seleccionadas, así como las funciones y servicios que ellas ofrecen a sus superiores



# Estructura de los Sistemas Operativos

---

## MicroKernel

### Características:

La idea es mover una gran cantidad de funcionalidades al espacio de usuario

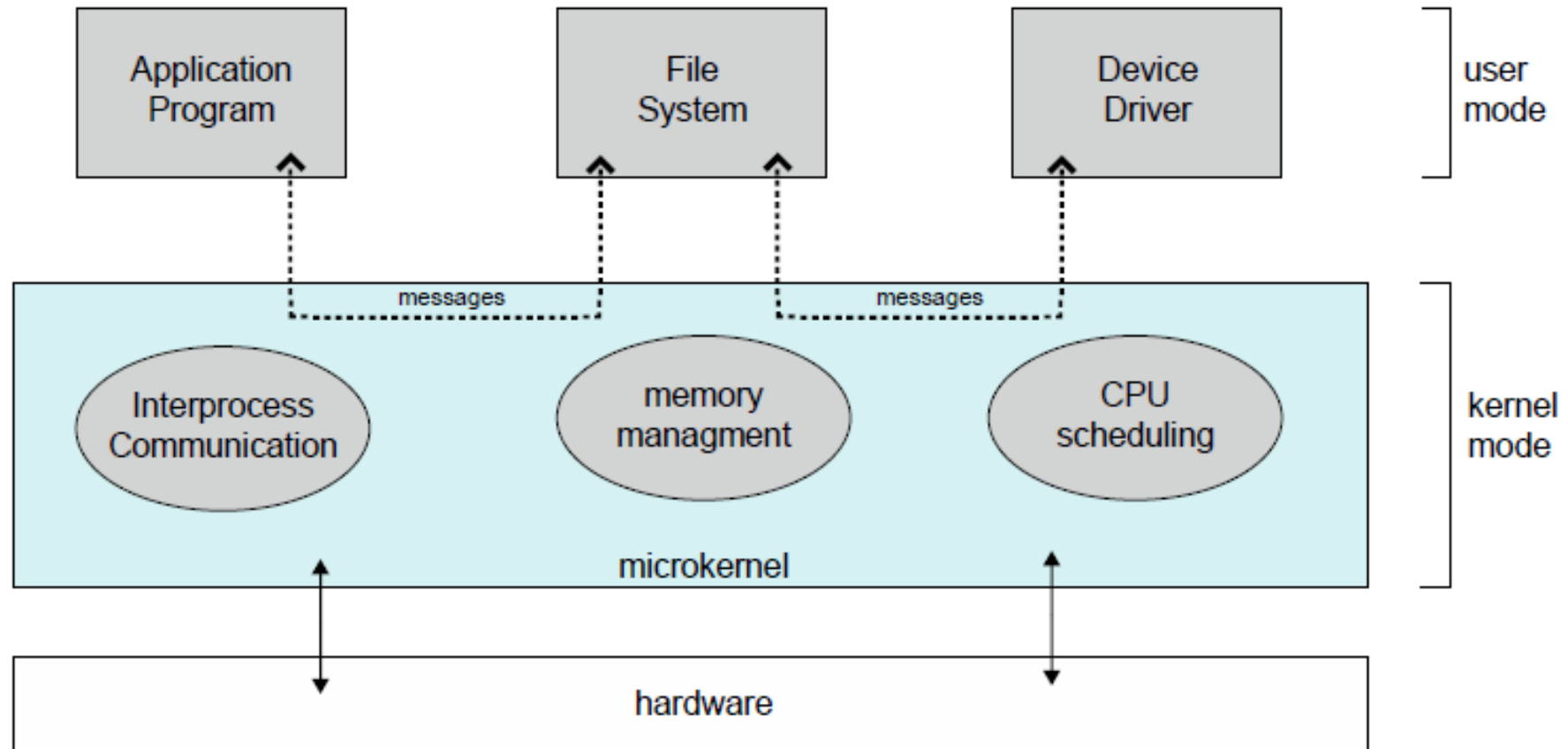
#### ¿Por qué?

- ❑ Mach es un ejemplo de microkernel, MINIX, QNX
- ❑ El kernel de Mac OS X (Darwin) está parcialmente basado en Mach
- ❑ La comunicación se lleva a cabo mediante el pase de mensajes entre los módulos que residen en el espacio de usuario

### Beneficios:

- ❑ Es sencillo extender las funcionalidades del microkernel
- ❑ Es fácil portar el SO a nuevas arquitecturas
- ❑ Mayor confiabilidad
- ❑ Menos código ejecutándose en el espacio del kernel
- ❑ Mayor seguridad
- ❑ **Desventajas:** Sobrecarga ocasionada por la constante comunicación entre los módulos que se ejecutan en el espacio de usuario

# Estructura de los Sistemas Operativos



# Estructura de los Sistemas Operativos

---

## Módulos

- ❑ Los SO modernos implementan el soporte de carga de módulos kernel
- ❑ Usan un enfoque orientado a objetos
- ❑ Cada componente principal se encuentra separado
- ❑ La comunicación se realiza mediante interfaces bien conocidas
- ❑ Es posible cargar o descargar funcionalidades del kernel en caliente
- ❑ Su arquitectura es muy similar a la estructura de capas pero más flexible
- ❑ Ejemplo: Linux, Solaris, etc.

# Estructura de los Sistemas Operativos

---

## Sistemas Híbridos

- ❑ La mayoría de los SO modernos no implementa un modelo puro
- ❑ Los modelos híbridos combinan diferentes enfoques con la finalidad de mejorar en desempeño, seguridad, etc.
- ❑ Linux y Solaris utilizan un enfoque monolítico con soporte de módulos
- ❑ Windows es básicamente monolítico, con una estructura de microkernel para diferentes subsistemas
- ❑ Mac OS x utiliza un enfoque híbrido de capas, una interfaz gráfica (Aqua), más un ambiente de programación (Cocoa)
- ❑ Por debajo, el kernel se estructura como un microkernel estilo Mach, y un conjunto de partes de UNIX BSD. Adicionalmente, el kernel soporta la carga y descarga de módulos, funcionalidad conocida como soporte de extensiones de kernel y extensiones de I/O

# Estructura de los Sistemas Operativos

---

## Android

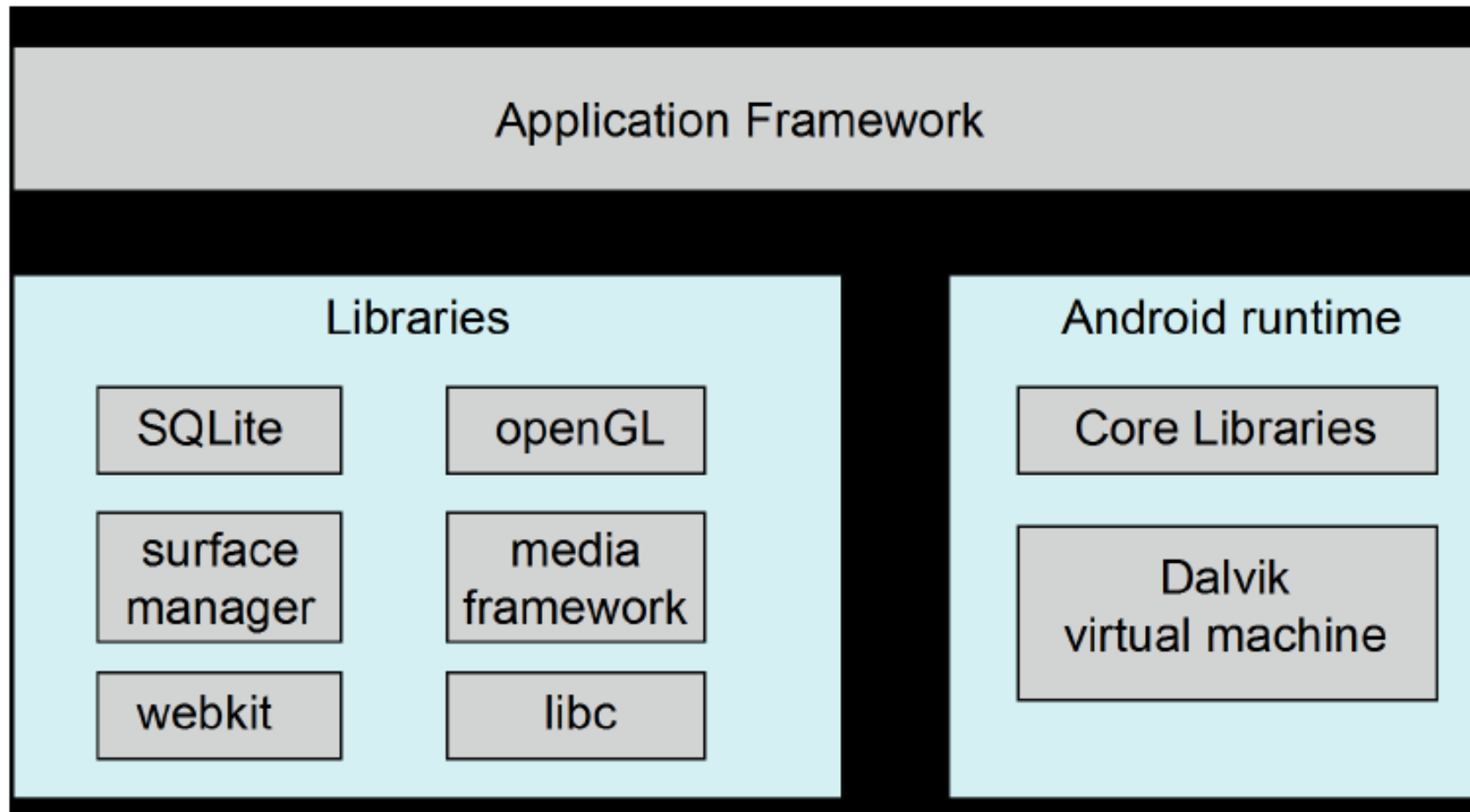
Desarrollado por Google en alianza con la Open Handset Alliance

### Código abierto

- ❑ Estructurado en capas
- ❑ Basado en el kernel de Linux, pero con modificaciones importantes
- ❑ Provee manejo de procesos, memoria, drivers de dispositivos
- ❑ Mejoras en el manejo de energía
- ❑ El ambiente de ejecución incluye un conjunto de bibliotecas y la máquina virtual Dalvik

# Tipos de Sistemas Operativos

---



# TIPOS DE SISTEMAS OPERATIVOS



# Tipos de Sistemas Operativos

---

## Entornos de Computación

- ☐ Ordenadores personales
- ☐ Dispositivos de mano: móviles y tabletas
- ☐ Sistemas empotrados (*embedded systems*)
- ☐ Servidores + multiprocesadores
- ☐ *Clusters* de servidores
- ☐ Supercomputadores
- ☐ Sistemas en la nube (*clouds*)

# Tipos de Sistemas Operativos

---

## Algunos Tipos

- ☐ Procesamiento por lotes (*batch processing*)
- ☐ Tiempo Compartido (*time sharing*)
- ☐ Tiempo Real (*real time*)
- ☐ Sistemas Multiusuario
- ☐ Máquinas Virtuales

# Tipos de Sistemas Operativos

---

## Procesamiento por Lotes (*batch processing*)

- ❑ Históricamente, fueron los primeros SO (principios de los 1950)
- ❑ Objetivo: automatizar la ejecución de trabajos y aumentar la utilización del procesador
- ❑ Los trabajos se agrupaban en lotes que se iban ejecutando en secuencia
- ❑ Los más primitivos eran secuenciales; la multiprogramación se incorporó a finales de los 50
- ❑ Primer lenguaje para dictar tareas al SO JCL (*Job Control Language*)

# Tipos de Sistemas Operativos

---

## Sistemas de Tiempo Compartido (*time sharing*)

- ❑ Inventados en los 1950, comercializados en los 60
- ❑ Avance sobre los sistemas por lotes, para conseguir interactividad
- ❑ Cada proceso dispone de una pequeña rodaja de tiempo periódica
- ❑ Los procesos se van turnando en la CPU
- ❑ Si la rodaja de tiempo es lo bastante pequeña (milisegundos), el usuario no percibe las pausas periódicas de su sesión

# Tipos de Sistemas Operativos

---

## Sistemas de Tiempo Real (*real time systems*)

- ❑ Diseñados para cumplir tareas que deben completarse en un plazo prefijado (sistemas de control industrial, sistemas multimedia...)
- ❑ Usan algoritmos de planificación de procesador especiales
- ❑ S.T.R. crítico = para industria y sistemas empuotrados en los que el cumplimiento de plazos es crítico. Suelen prescindir de servicios que afectan a los tiempos (ej. Memoria virtual)

# Tipos de Sistemas Operativos

---

## Sistemas Multiusuario

- ❑ Un sistema multiusuario reconoce que hay varios perfiles de acceso, con privilegios distintos:
  - Permisos de acceso a ficheros y aplicaciones
  - Cuotas de espacio o de tiempo de procesador
  - Prioridad en el acceso a los recursos
- ❑ Importante: multiusuario  $\neq$  multitarea (puede haber sistemas multitarea que no son multiusuario)

# Tipos de Sistemas Operativos

---

## Multiprocesadores

- ❑ Desde unos pocos hasta miles de procesadores
- ❑ Varios modelos de acceso a memoria:
  - UMA: Memoria compartida
  - NUMA: Memoria no compartida
- ❑ Varios modelos de ejecución de procesos:
  - SMP: Multiprocesamiento simétrico = una tarea se puede ejecutar en cualquier procesador
  - AMP: Multiprocesamiento asimétrico = hay especialización de tareas (ej. un procesador ejecuta el SO y otro los procesos de usuario)

# Tipos de Sistemas Operativos

---

## Sistemas Distribuidos

- Un Sistema Distribuido consiste en un conjunto de computadores conectados en red y que se utiliza como si fuera un único sistema con múltiples procesadores + una gran memoria compartida + un gran almacenamiento secundario
- No existe un «sistema distribuido universal», pero sí hay servicios con características de sistema distribuido.
  - Servicios en la nube (Dropbox, Amazon...)
  - La WWW



# Tipos de Sistemas Operativos

---

## Máquinas Virtuales

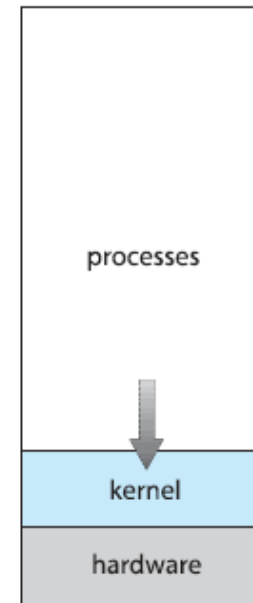
- ❑ Emulación por software de una máquina física
- ❑ Sobre la máquina virtual pueden ejecutarse programas implementados para la máquina física emulada
- ❑ Ventaja: no necesitamos el sistema original
- ❑ Inconveniente: la emulación es más lenta
- ❑ Ejemplos de máquinas virtuales
  - IBM VM: (años 1960) sobre un sistema por lotes, ofrecía a cada usuario su propia máquina virtual no multiprogramada; las m.v. se planificaban con tiempo compartido
  - Java: los programas compilados en Java corren sobre una máquina virtual (JVM)
  - VMware: capaz de ejecutar al mismo tiempo varias sesiones Windows, Linux, Mac OS X, etc. sobre plataforma PC o Mac.

# Tipos de Sistemas Operativos

## Usos de las Máquinas Virtuales

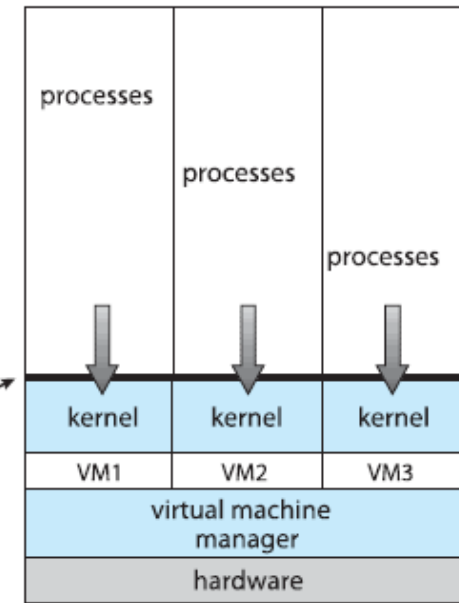
- ❑ Crear entornos protegidos: cada máquina virtual está aislada de las otras
- ❑ Independencia de la plataforma (ej. Java)  
Emulación por software de una máquina física
- ❑ Pervivencia de sistemas antiguos (ej. emuladores MSDOS, consolas de juegos...)
- ❑ Desarrollo: se pueden escribir y probar aplicaciones para un hardware que no tenemos. Sobre la máquina virtual pueden ejecutarse programas implementados para la máquina física emulada

Sin virtualización



(a)

Con virtualización



(b)

# Logros de los Sistemas Operativos

---

## Algunos logros históricos de los SO

- ☐ Multiprogramación
- ☐ Memoria paginada
- ☐ Memoria virtual
- ☐ Sistemas de archivos
- ☐ Control del acceso concurrente
- ☐ Protección y seguridad
- ☐ Interfaz uniforme con la E/S