Sistemas Operativos 1.0

UNITEC

Tema #2
Tour por un OS

Basado en los slides de
William Stallings

Un sistema operativo es…

- Un programa que controla la ejecución de aplicaciones
- Una interface entre las aplicaciones y el hardware

Objetivos principales de un OS:

- Conveniencia
- Eficiencia
- Capacidad de evolucionar

Harware y la infraestructura de Soft.

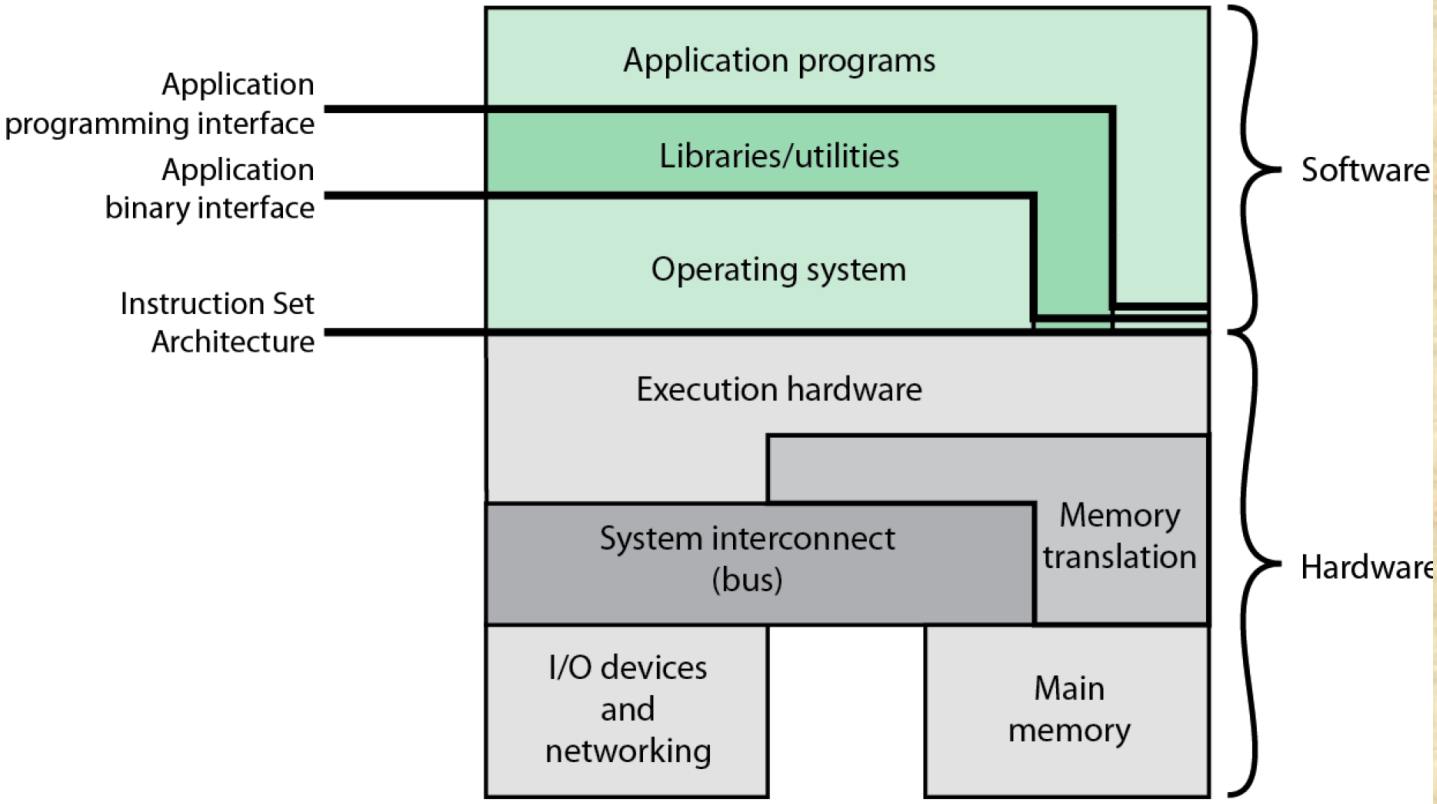


Figure 2.1 Computer Hardware and Software Infrastructure

Servicios de un OS

- Desarrollo de Programas
- Ejecución de programas
- Acceso a dispositivos de I/O
- Acceso controlado a archivos
- Acceso al sistema
- Detección y respuesta de errores
- Contabilización de uso

Interfaces Primarias

- Instruction set architecture (ISA)
- Application binary interface (ABI)
- Application programming interface (API)

El rol de un OS

- Una computadora es un conjunto de recursos para el movimiento, almacenamiento y procesamiento de data
- El OS es responsable de administrar esos recursos

El OS como software

- Funciona de la misma manera que cualquier software
- Programa, o conjunto de programas ejecutados por el procesador
- Frecuentemente libera el control y depende del procesador para que le permita retomar el control

OS como administrador de recursos

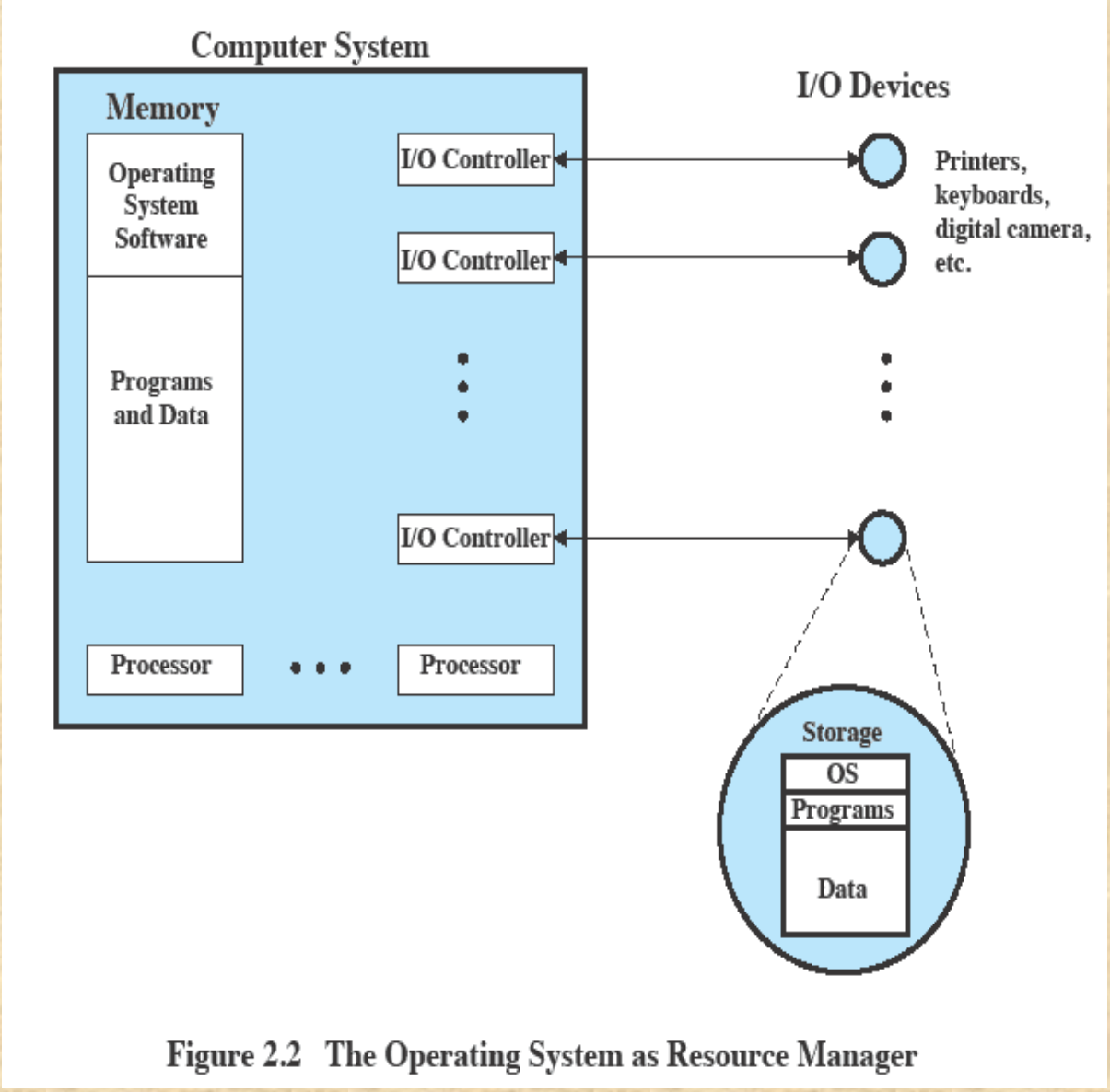


Figure 2.2 The Operating System as Resource Manager

Features necesarios en el hardware

Protección de memoria para el OS

- Mientras el programa del usuario está ejecutando, no debe alterar la memoria del OS

Timer

- Evita que un proceso monopolice el sistema

Instrucciones privilegiadas

- Sólamente pueden ser ejecutadas por el OS

Interrupts

- Le da al OS más flexibilidad en controlar los programas del usuario

Modo dual de operación

Modo usuario

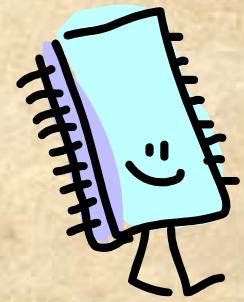
- Aplicaciones del usuario ejecutan en modo usuario
- Se protegen ciertas áreas de memoria
- No se permite la ejecución de ciertas instrucciones

Modo kernel

- OS ejecuta en modo kernel
- Se pueden ejecutar instrucciones privilegiadas
- Se puede accesar toda la memoria

El OS introduce cierto overhead (sobrecosto)

- Tiempo del procesador alterna entre la ejecución de procesos del usuario y ejecución del OS
- Sacrificios:
  - Cierta cantidad de memoria asignada al OS
  - Algo de tiempo de procesador lo consume el OS
- A pesar del overhead, el OS mejora la utilización del sistema completo



Qué ocurre en un sistema batch multiprogramado

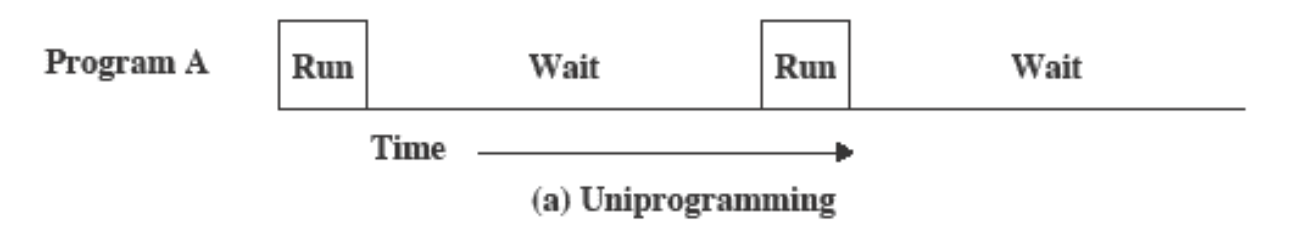
Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Figure 2.4 System Utilization Example

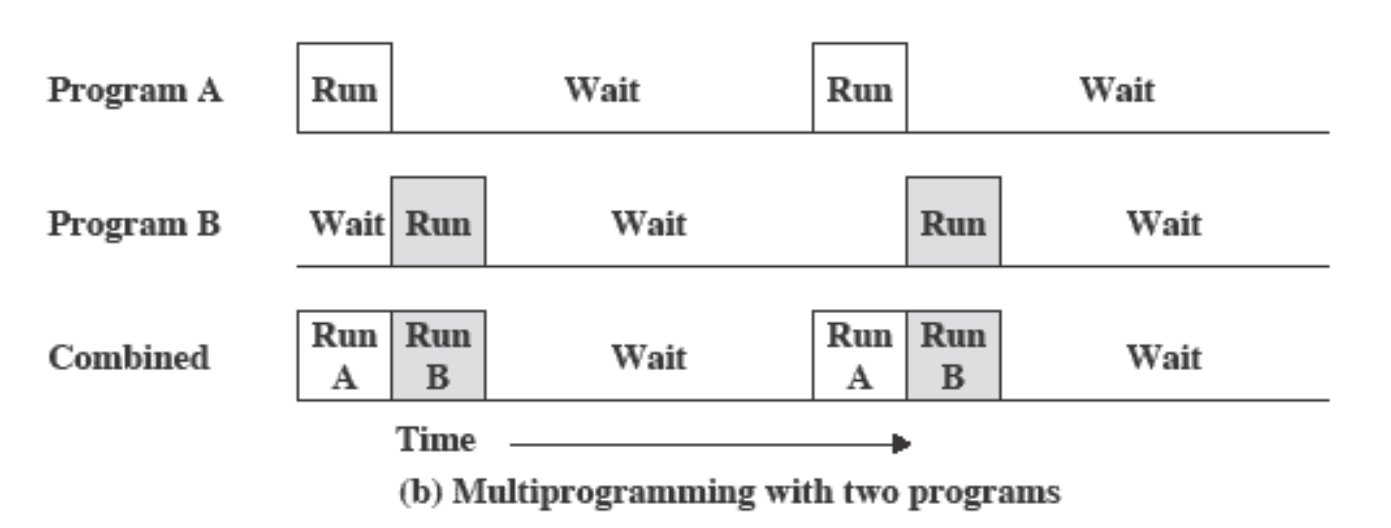
- Procesador está casi siempre desocupado (idle)
  - Dispositivos de I/O son más lentos que el CPU

Uniprogramación



- El procesador pasa cierto tiempo ejecutando hasta que ocurre una instrucción de I/O; debe esperar hasta que la operación concluya antes de proceder

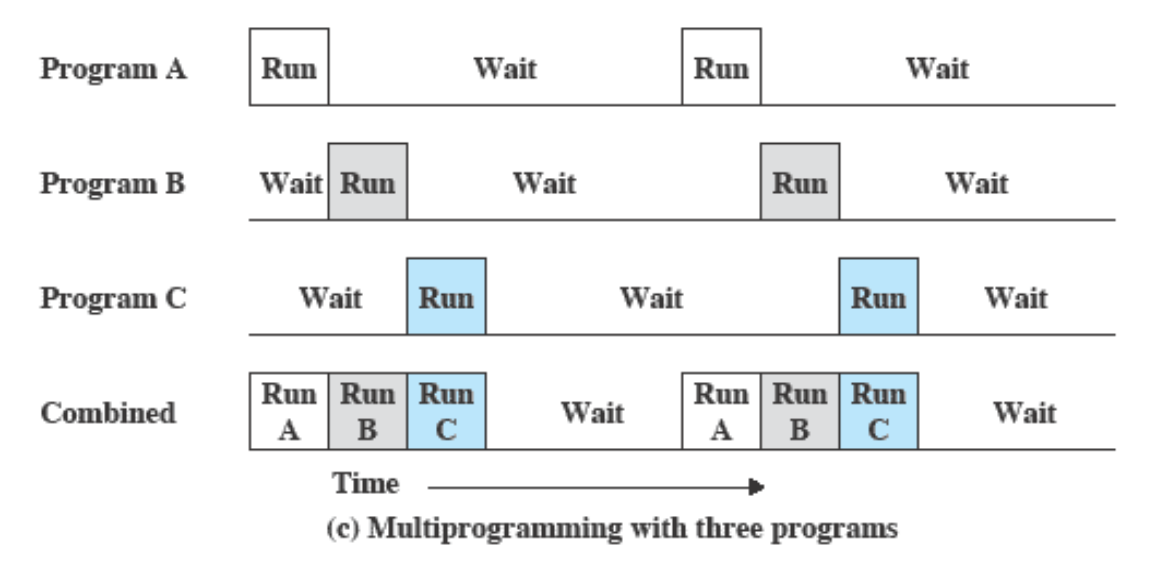
Multiprogramación



(b) Multiprogramming with two programs

- Debe haber suficiente memoria para mantener el OS y un proceso de usuario
- Cuando un proceso necesita esperar operaciones de I/O, el procesador puede intercambiarse al otro proceso, que no está esperando I/O

Multiprogramación



(c) Multiprogramming with three programs

- Multiprogramación
  - También conocida como multitasking
  - Se incrementa la memoria para tener tres, cuatro o más procesos en memoria, y multiplexar entre ellos

Ejemplo de multiprogramación

Table 2.1 Sample Program Execution Attributes

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Histogramas de utilización

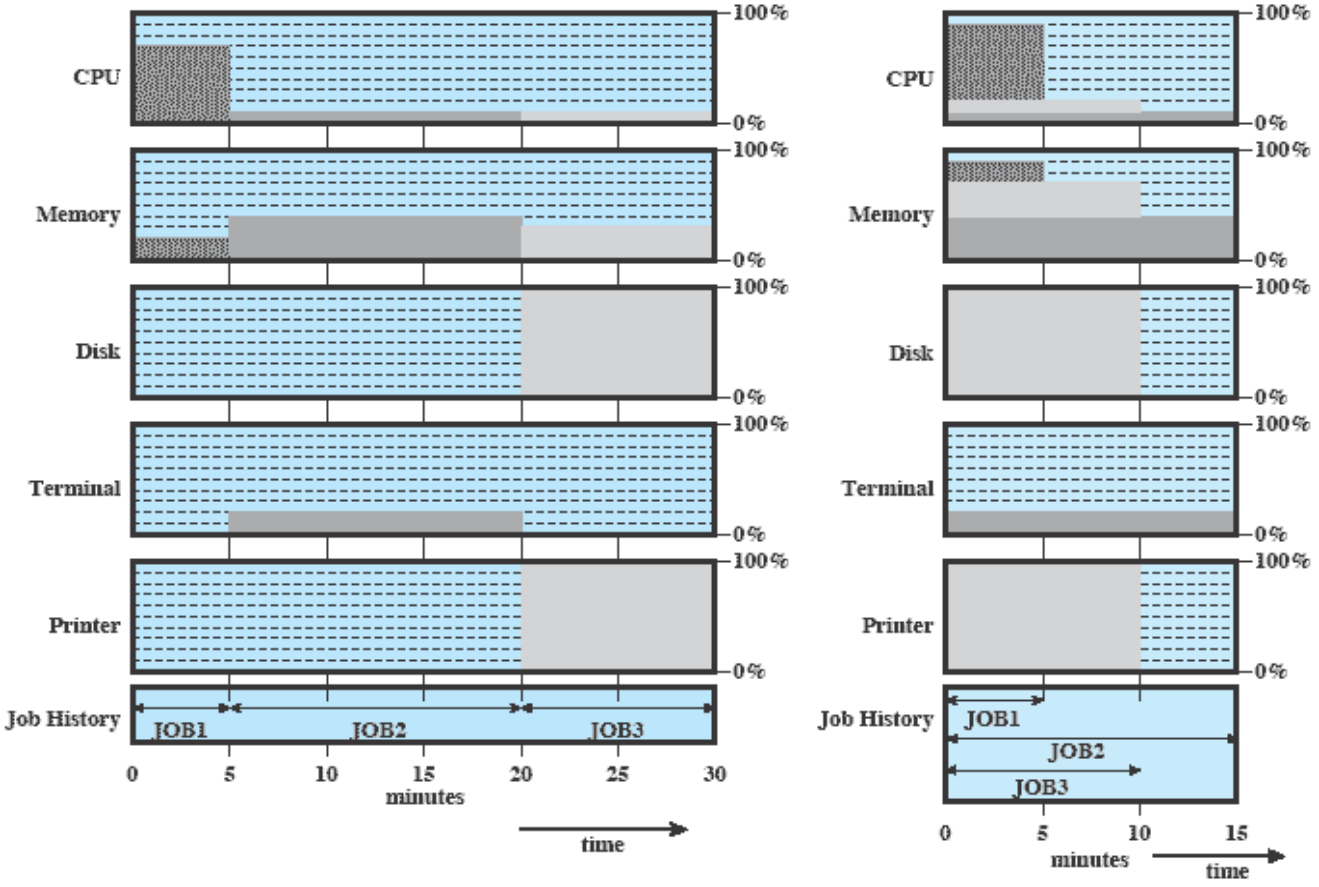


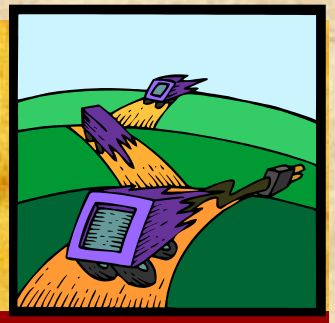
Figure 2.6 Utilization Histograms

Efectos en el uso de recursos

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Table 2.2 Effects of Multiprogramming on Resource Utilization

Procesos



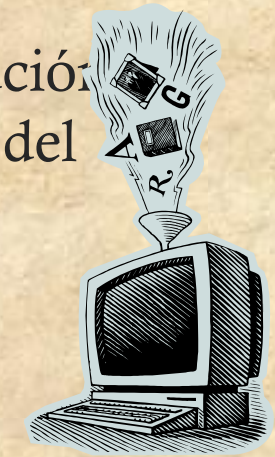
- Concepto fundamental para la estructura del OS

Un *proceso* puede definirse como:

- Un programa en ejecución
- Una instancia de un programa corriendo
- La entidad que puede ser asignada y ejecutada en un procesador
- Una unidad de actividad caracterizada por un hilo de ejecución, un estado actual y un conjunto de recursos del sistema asociados

Componentes de un proceso

- Un proceso tiene tres componentes:
  - Un programa ejecutable
  - Los datos necesarios por el programa
  - El contexto de ejecución o estado de proceso del programa



- El contexto de ejecución es muy importante:
  - Datos internos que el OS usa para supervisar y controlar el proceso
  - Incluye el contenido de los registros del CPU
  - Incluye información sobre la prioridad del proceso, o si el proceso está esperando algún evento de I/O

Administración de procesos

- El estado completo de un proceso está contenido en su contexto
- Nuevos features pueden ser diseñados e incorporados en el OS expandiendo el contexto para incluir nueva información para soportar el feature

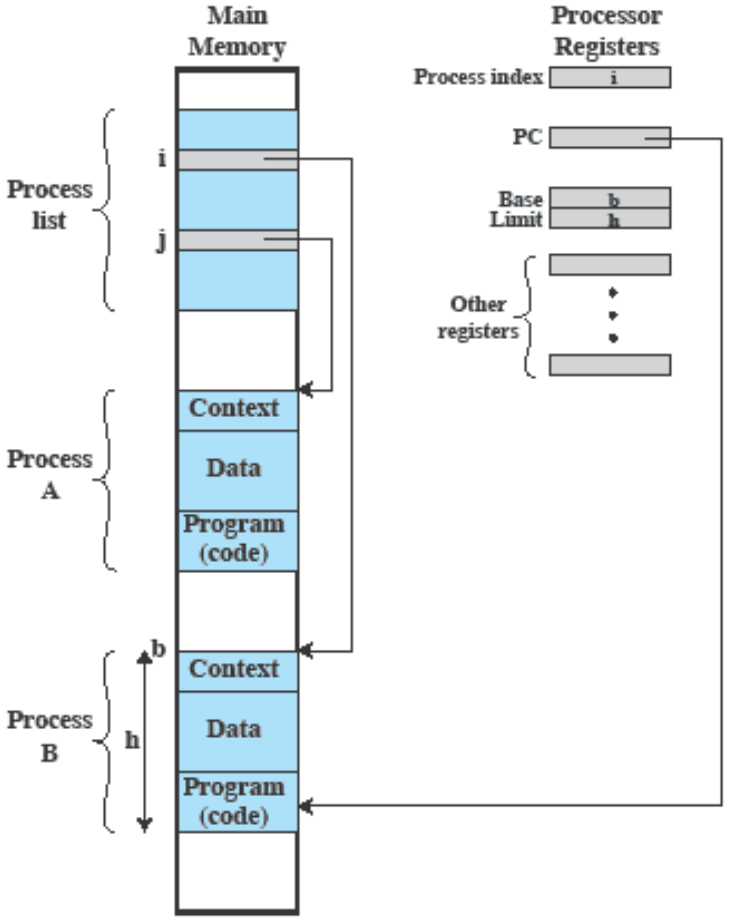


Figure 2.8 Typical Process Implementation

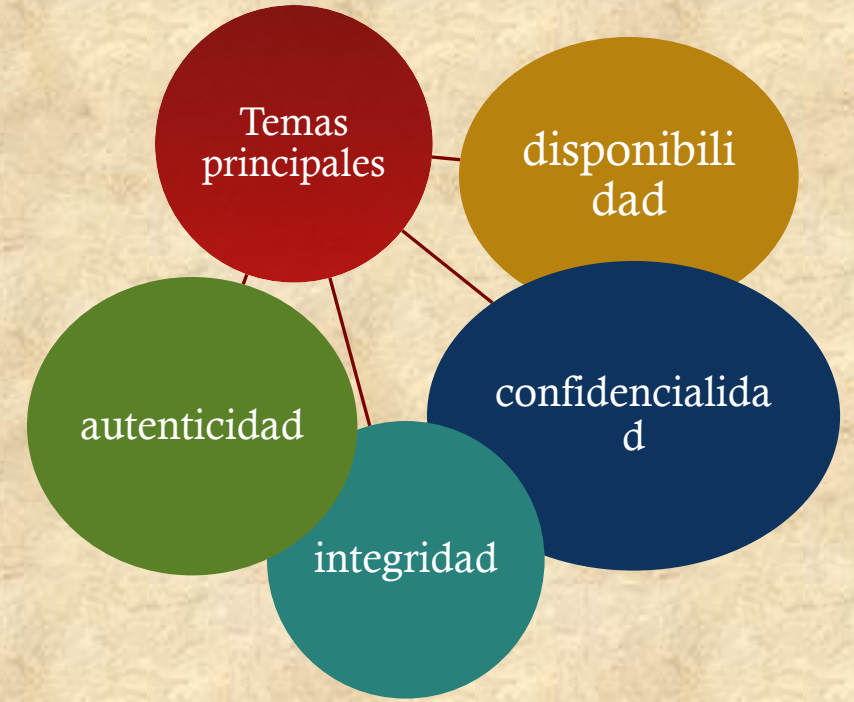
Administración de memoria

- El OS tiene cinco responsabilidades en el manejo de almacenamiento:

Aislar procesos	Asignación automatica	Soportar programación modular	Protección y control de acceso	Almacenamiento de largo plazo

Protección y seguridad

- La naturaleza de la amenaza que involucra una organización depende de sus circunstancias
- El problema radica en controlar el acceso a los sistemas y a la información contenida en ellos



Planificación y Administración de Recursos

- Responsabilidad primaria del OS es administrar recursos
- Las políticas de asignación de recursos deben considerar:

Elementos clave de un OS

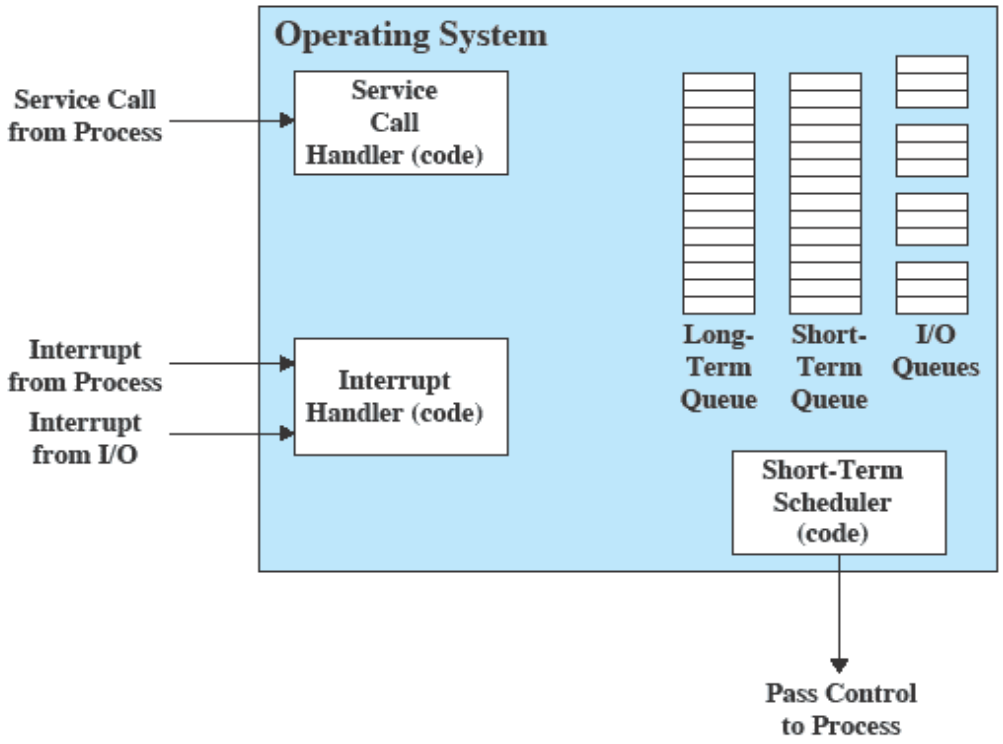


Figure 2.11 Key Elements of an Operating System for Multiprogramming

Arquitecturas de un OS

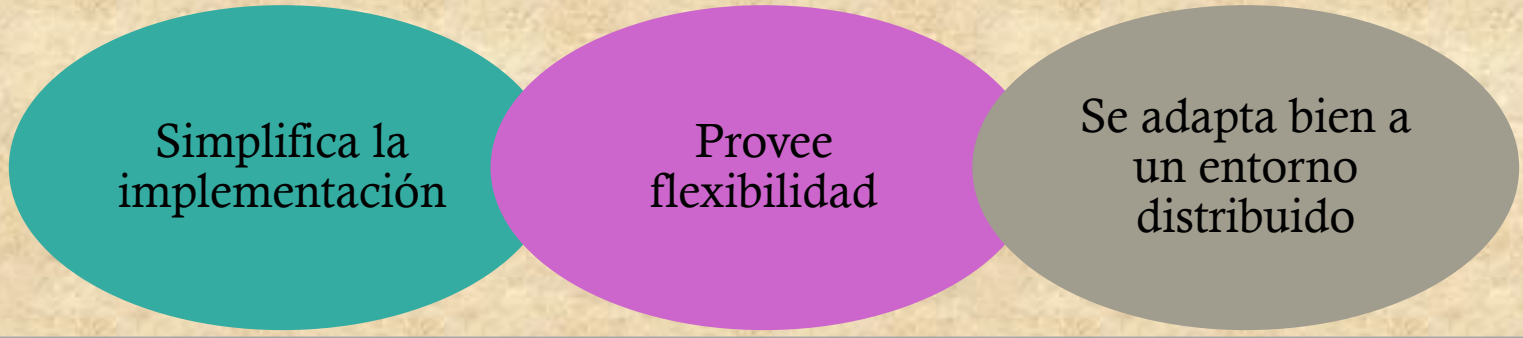
Diferentes formas de organizar el OS dependiendo de las demandas actuales:

- Arquitectura de microkernel
- Multithreading
- Multiprocesamiento simétrico
- Sistemas operativos distribuidos
- Diseño orientado a objetos

Arquitectura microkernel

- Asigna unas pocas funciones esenciales al kernel:

- El enfoque:



Multithreading

- Técnica en la que un proceso se divide en hilos (threads) que pueden correr en forma concurrente

Thread

- Unidad planificable de trabajo
- Incluye un contexto de CPU y su propia área de datos para habilitar subrutias
- Se ejecuta secuencialmente y es interrumpible (reentrante)

Proceso

- Colección de uno o más threads y sus recursos asociados
- Programador tiene mayor control sobre la modularidad de la aplicación y sus eventos

Multiprocesamiento simétrico (SMP)

- Término se refiere a una arquitectura de hardware y también al comportamiento del OS que explota esa arquitectura
- Varios procesos pueden correr en paralelo
- Múltiples procesadores son transparentes para el usuario
  - Estos procesasores comparten la misma memoria y facilidades de I/O
    - Arquitecturas NUMA
  - Todos los procesadores pueden realizar las mismas funciones
- El OS se preocupa de planificar los hilos o procesos en CPUs individuales, y de la sincronización entre CPUs

Ventajas del SMP

Performance	Más de un proceso corriendo simultáneamente
Disponibilidad	Falla de un proceso no detiene todo el sistema
Crecimiento	Performrmance de un sistema puede mejorar agregando otro CPU
Escalabilidad	Se pueden crear diferentes sistemas con capacidades diferentes

Multiprogramación

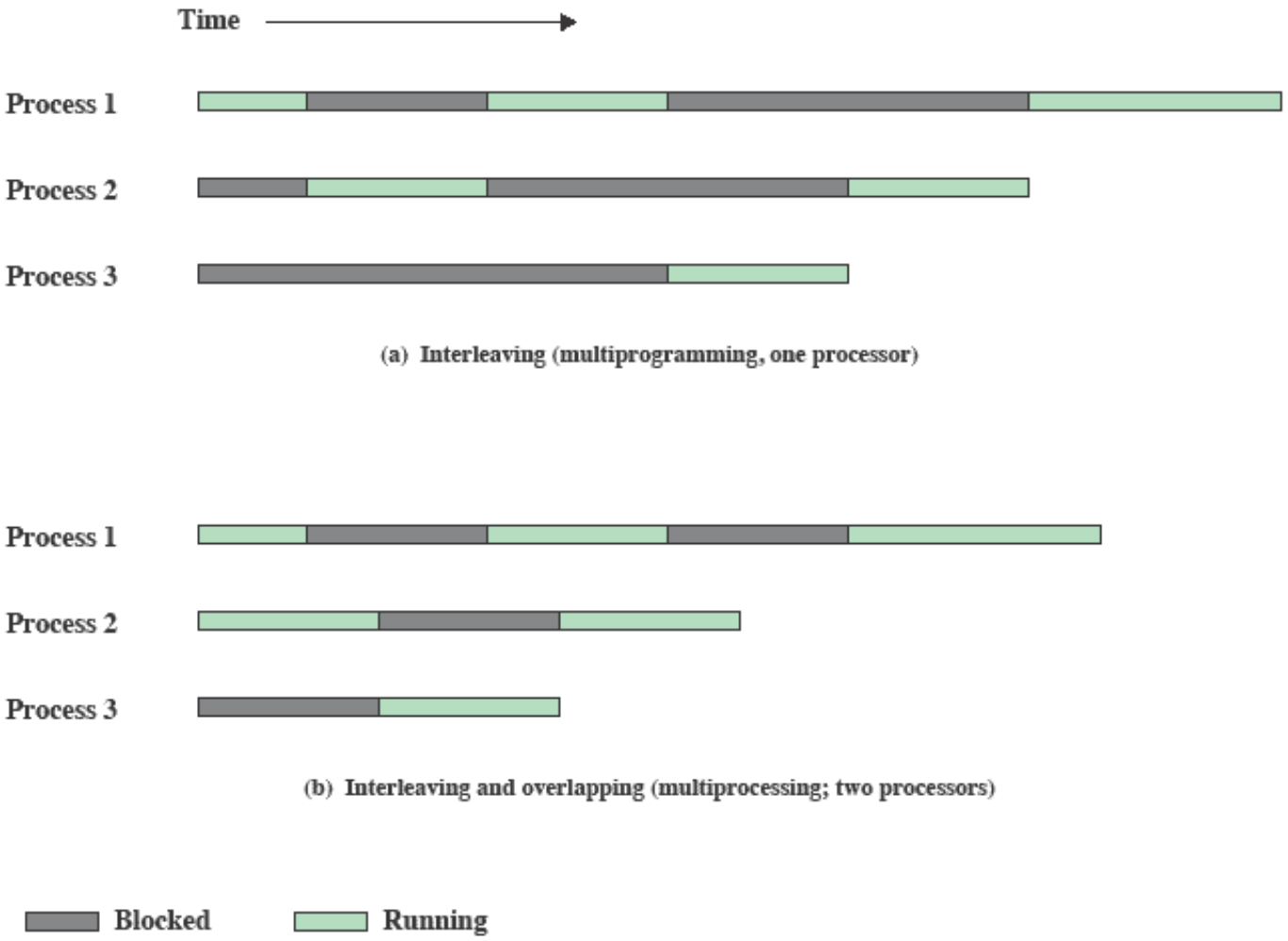


Figure 2.12 Multiprogramming and Multiprocessing

Diseño de OS

Sistema operativo distribuído

- Crea la ilusión de
  - Un único espacio de memoria
  - Único espacio de memoria secundaria
  - Facilidades de acceso unificadas
- El estudio de OS distribuídos está retrasado respecto a las otras arquitecturas

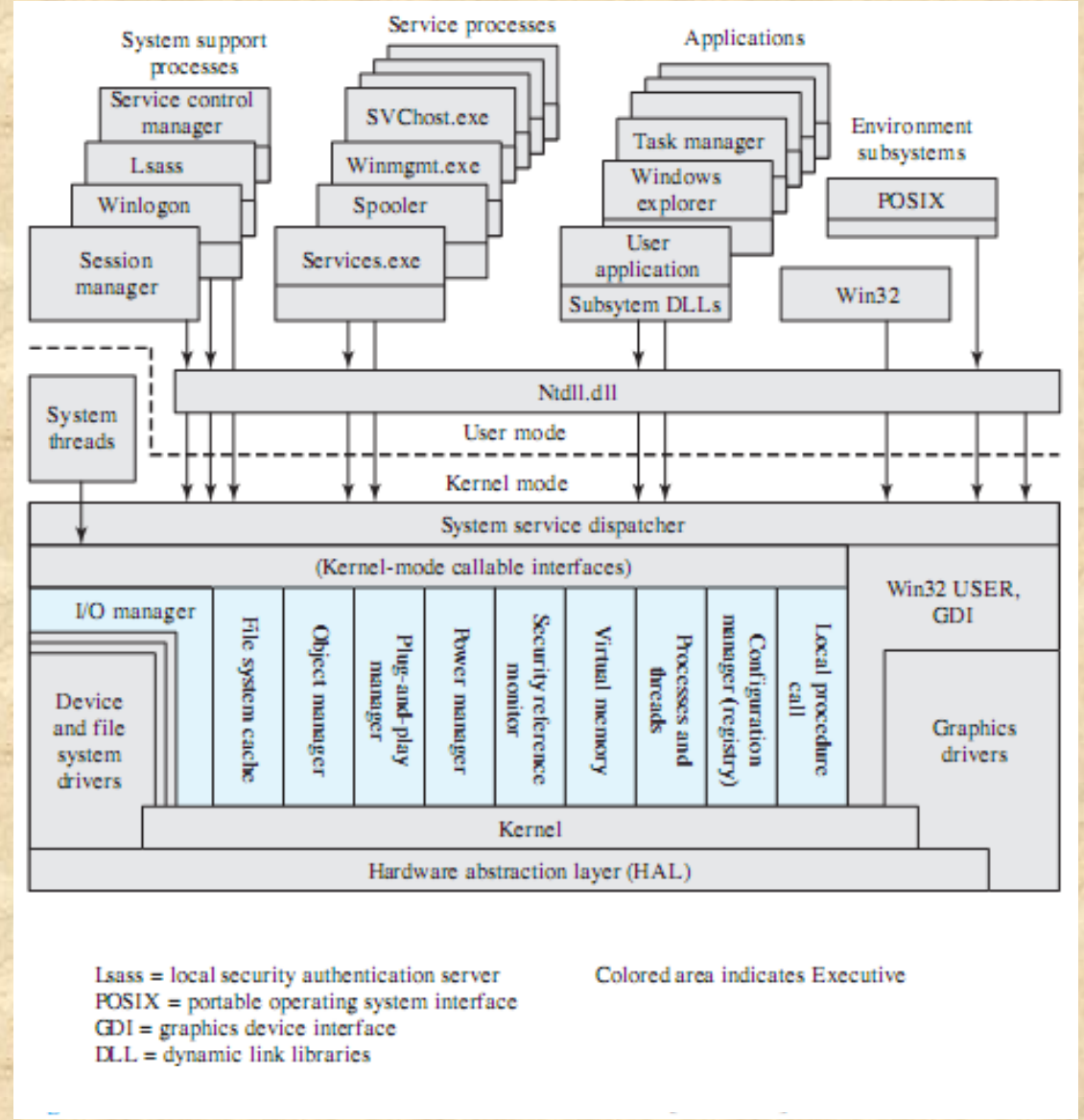
Diseño orientado a objetos

- Utilizado para agregar extensiones modulares al kernel
- Permite a los programadores personalizar un OS sin afectar la integridad del sistema
- Facilita el desarrollo de herramientas y sistemas distribuídos

Historia de MS Windows

- **MS-DOS 1.0** liberado en 1981
  - 4000 lineas de assembler
  - Corría en 8 Kbytes de memoria
  - Usaba procesador Intel 8086
- **Windows 3.0** liberado en 1990
  - 16-bit
  - GUI
  - Implementado como una capa sobre MS-DOS
- **Windows 95**
  - 32-bits
  - Dio paso al desarrollo de Windows 98 y Windows Me
- **Windows NT (3.1)** liberado en 1993
  - 32-bits con soporte de Windows, DOS y OS/2
- **Windows 2000**
  - Servicios y funciones para procesamiento distribuído
  - Active Directory
  - Plug and play y manejo de energía
- **Windows XP** liberado en 2001
  - Su meta era reemplazar las versiones de Windows basadas en MS-DOS, con tecnología NT
- **Windows Vista** liberado en 2007
- **Windows Server 2k8** liberado en 2008
- **Windows 7** liberado en 2009, as´´I como **Windows Server 2008 R2**
- **Windows Azure**
  - Cloud computing

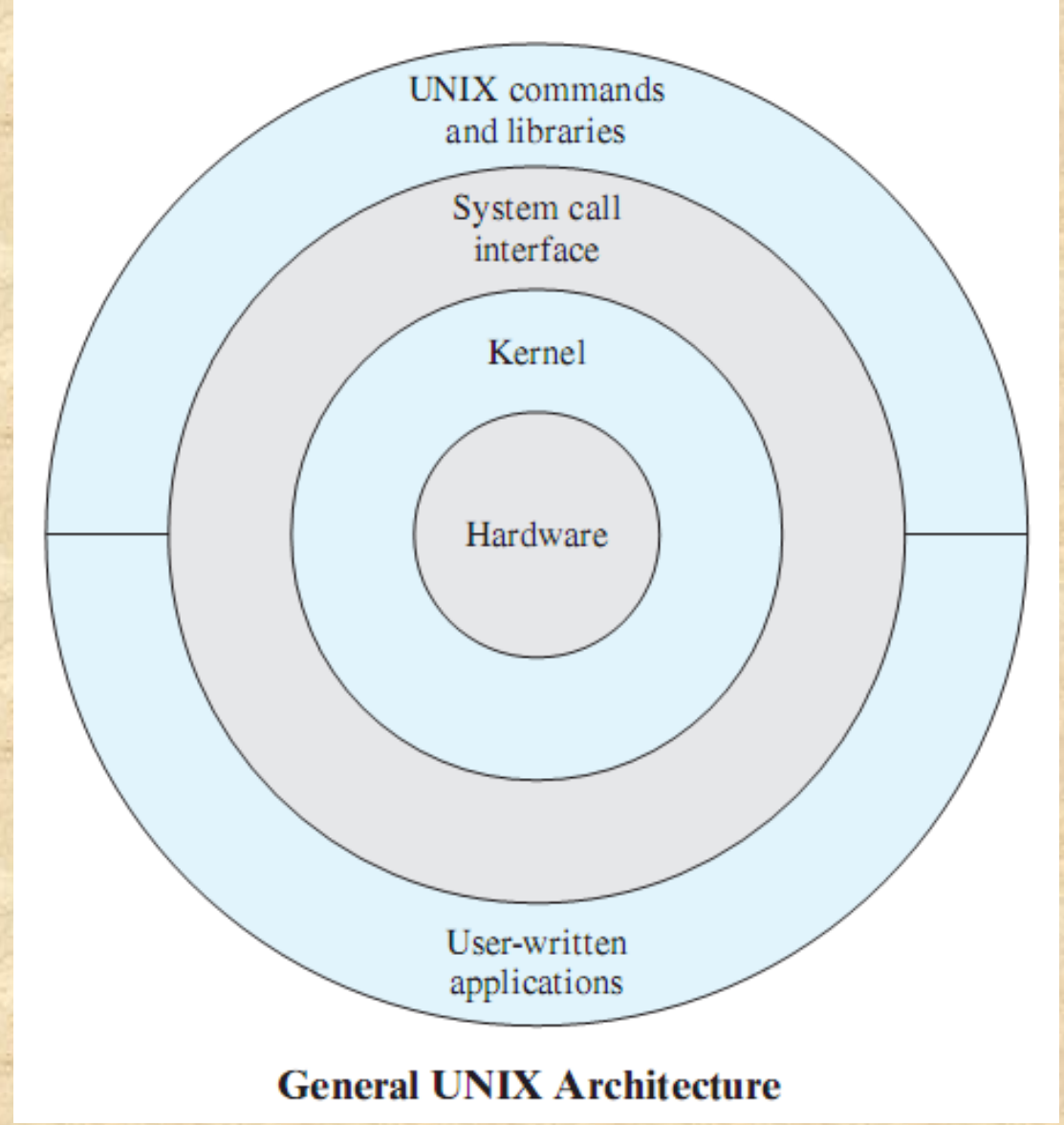
Arquitectura de Windows



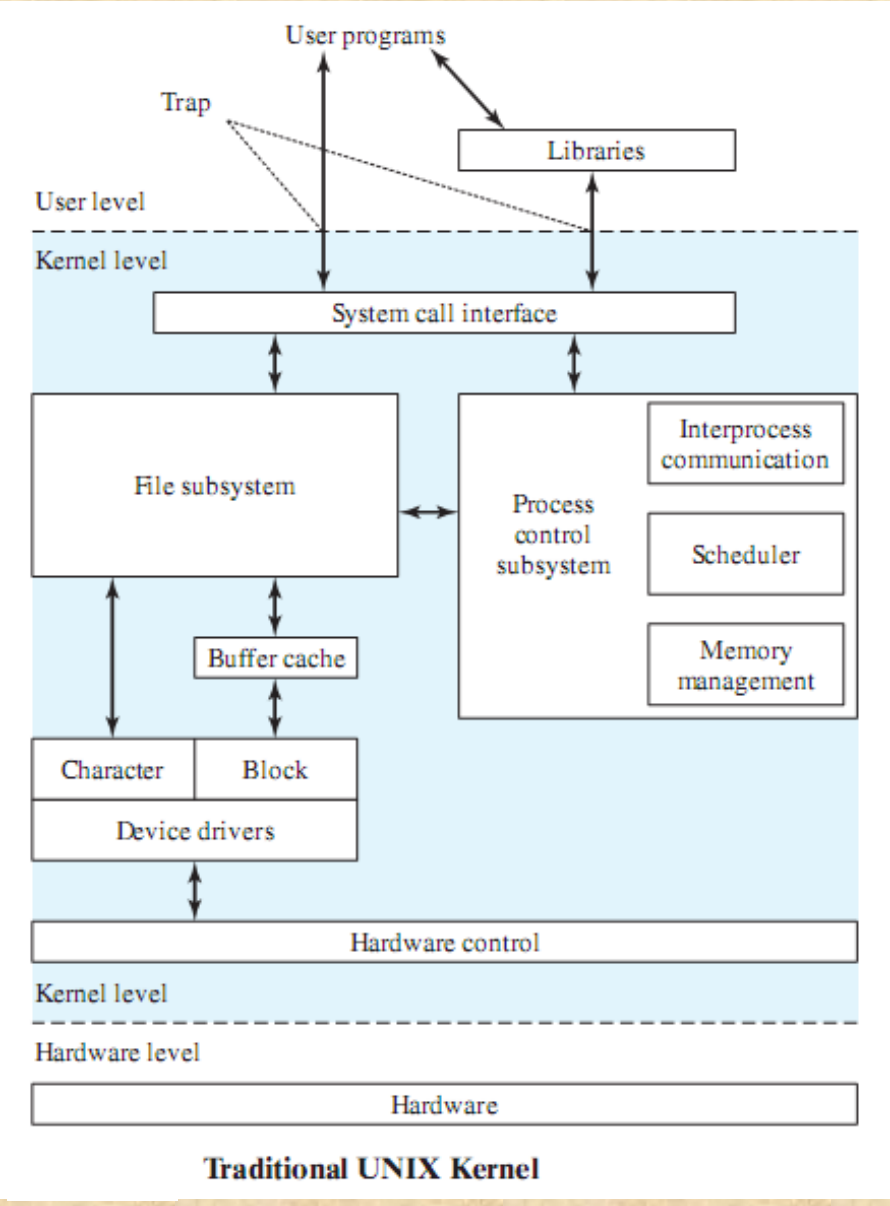
Unix tradicional

- Desarrollado en Bell Labs y fue operacional en una DEC PDP-7 en 1970
- Incorporando ideas de Multics
- PDP-11 fue importante porque mostró que UNIX podia ser implantado en otras computadoras
- Siguiente paso fue reescribir UNIX en C
  - Demostrando las ventajas de un lenguaje de alto nivel para el OS
- Descrito en una revista técnica en 1974
- Primera version disponible fuera de Bell Labs fue Version 6 en 1976
- Version 7, liberada en 1978 es el ancestor de los UNIX modernos
- UNIX BSD (Berkeley Software Distribution) fue muy importante, sin basarse en AT&T UNIX

Arquitectura de UNIX



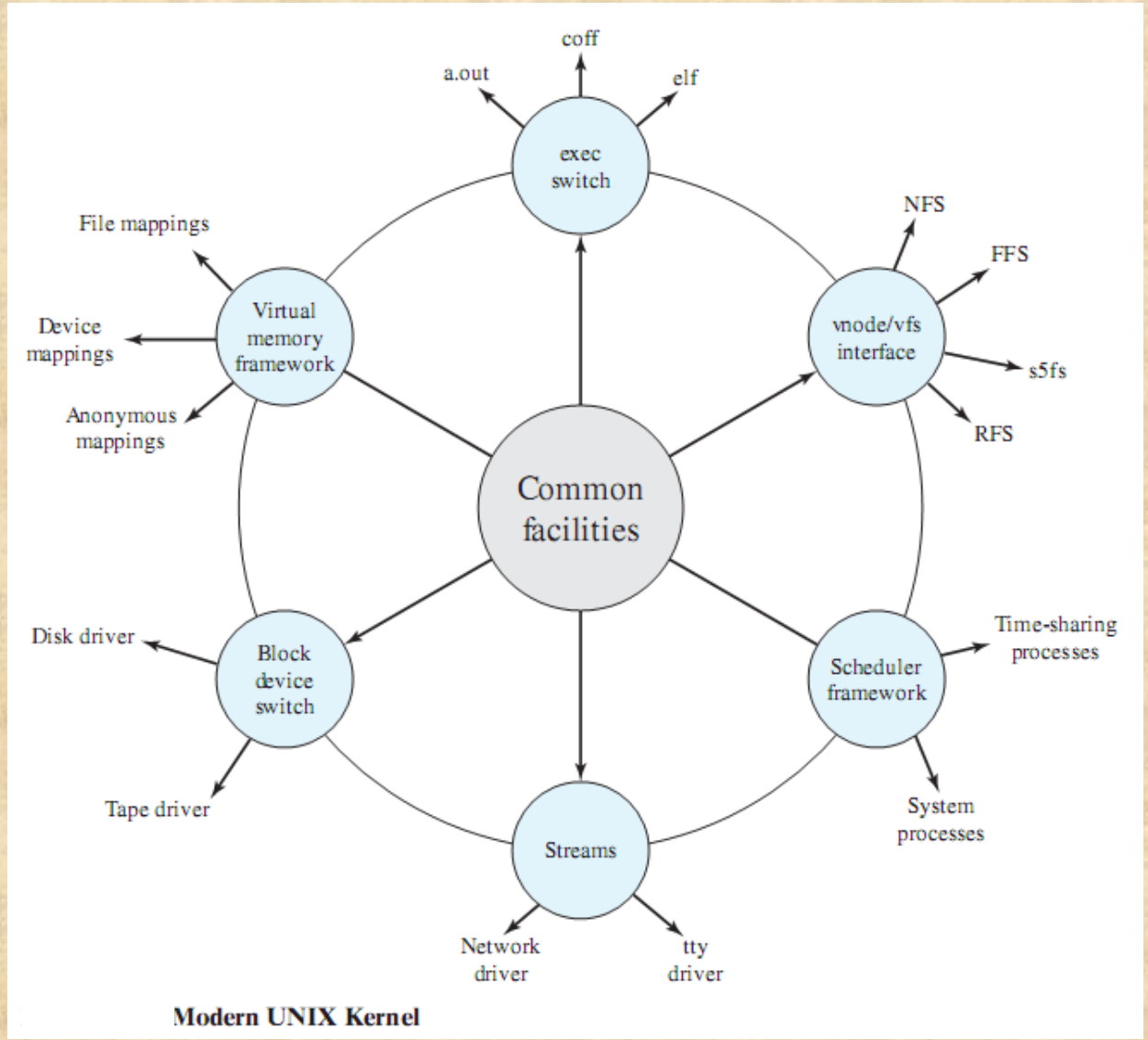
General UNIX Architecture



Traditional UNIX Kernel

Kernel de un UNIX tradicional

Kernel de un UNIX moderno



LINUX

- Comenzo como una variante de UNIX para el IBM PC basado en MINIX
- Escrito por Linus Torvalds, un estudiante finlandés
- Linux fue posteado en Internet en1991
- Hoy es un sistema UNIX complete, que corre en varias plataformas
- Es libre y su código fuente está disponible
- Su éxito se basa en la disponibilidad de paquetes de software libre
- Altamente modular y fácilmente configurable

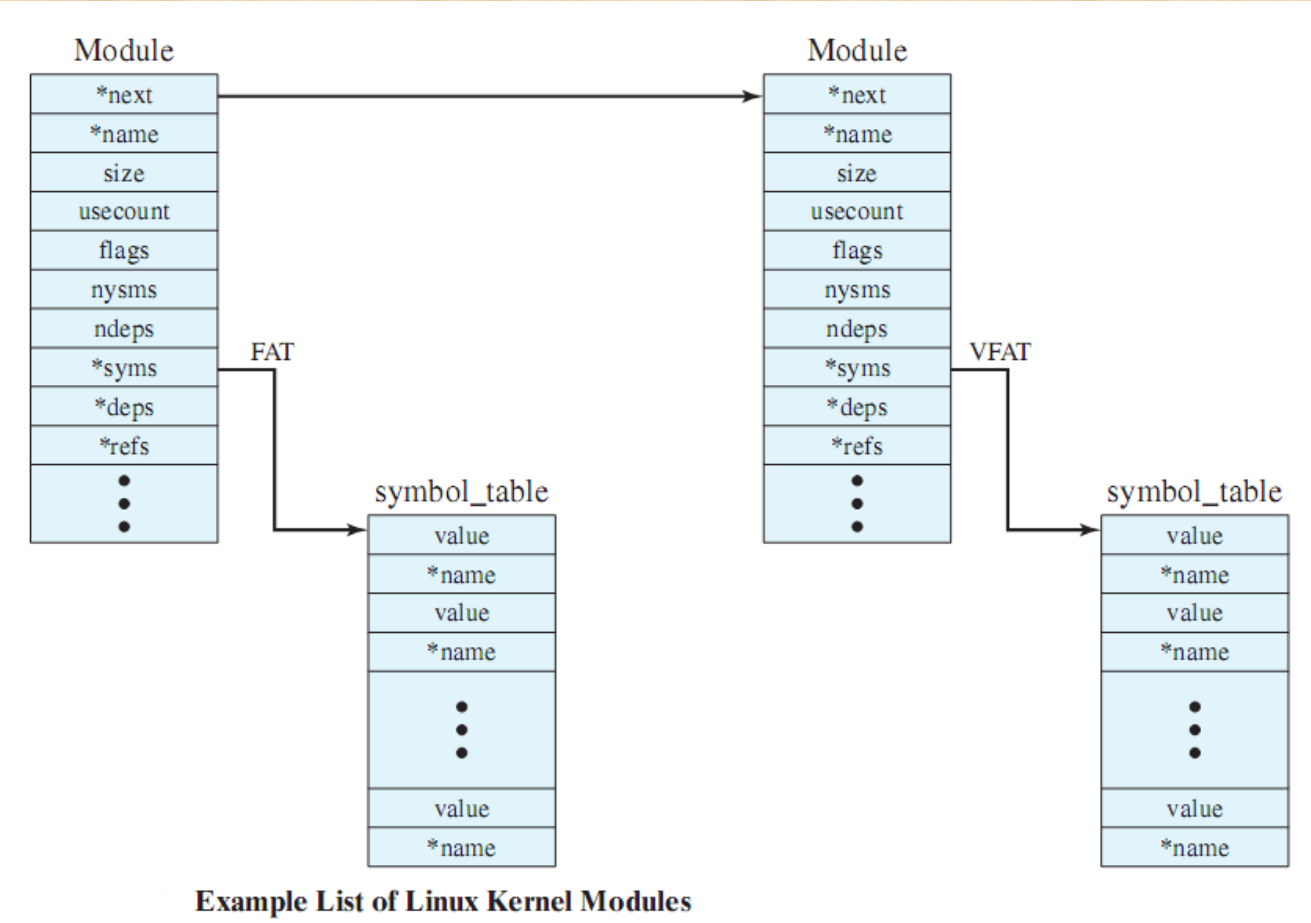
Kernel monolítico modular

- Casi toda la funcionalidad del OS está en un único bloque de código que corre como un solo proceso dentro de un espacio de direccionamiento
- Todos los componentes funcionales del kernel tienen acceso a todas sus estructuras y rutinas internas
- Linux está estructurado como una colección de módulos

Loadable Modules

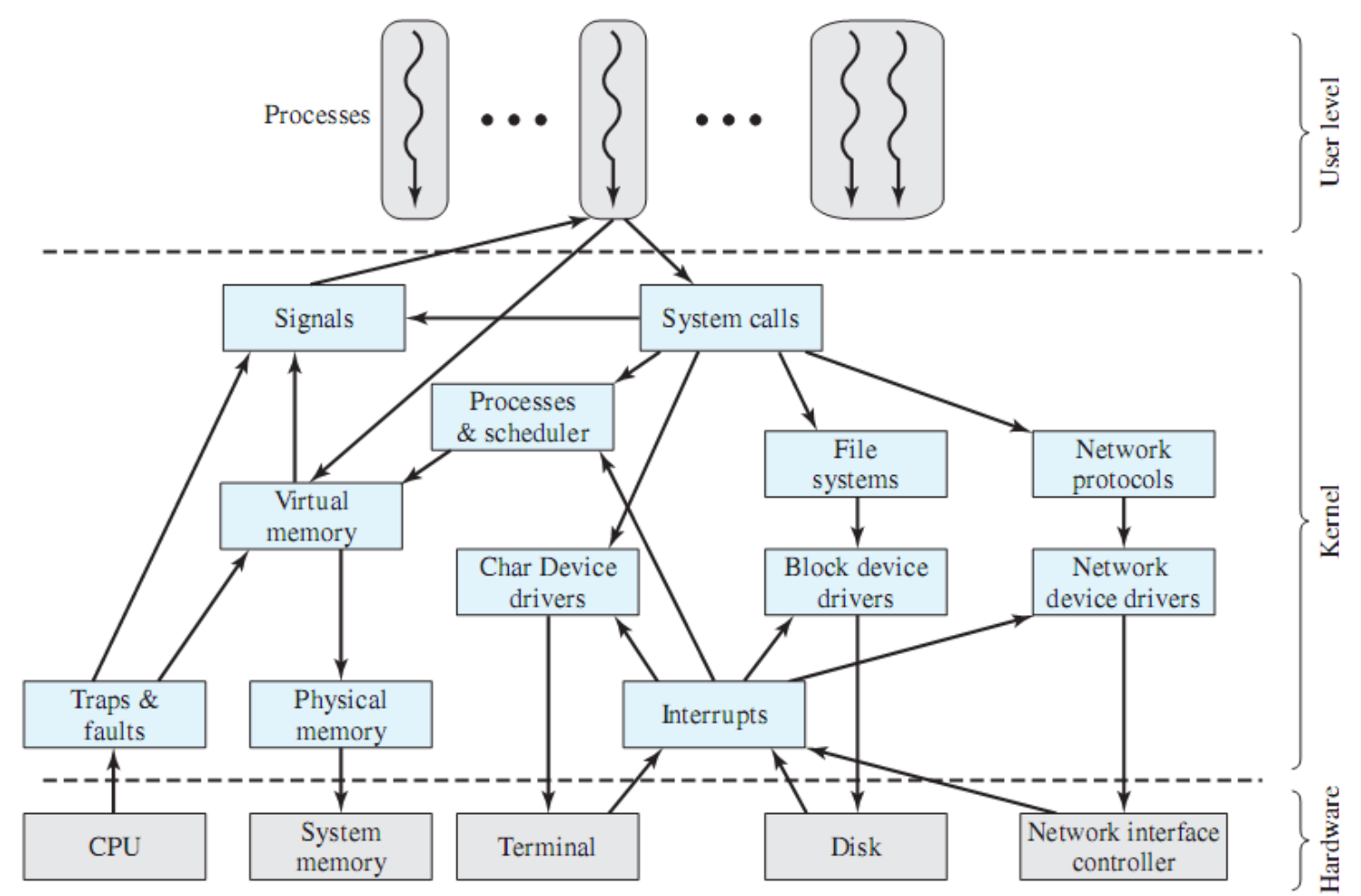
- Bloques relativamente independientes
- Un modulo es un archivo objeto cuyo código puede ser linkeado y deslinkeado del kernel en runtime
- Un modulo se ejecuta en modo kernel a nombre del proceso actual
- Dos características importantes:
  - Dynamic linking
  - Stackable modules

Módulos del kernel de Linux



Example List of Linux Kernel Modules

Componentes del kernel de Linux



Linux Kernel Components

Señales (signals) en Linux

SIGHUP	Terminal hangup	SIGCONT	Continue
SIGQUIT	Keyboard quit	SIGTSTP	Keyboard stop
SIGTRAP	Trace trap	SIGTTOU	Terminal write
SIGBUS	Bus error	SIGXCPU	CPU limit exceeded
SIGKILL	Kill signal	SIGVTALRM	Virtual alarm clock
SIGSEGV	Segmentation violation	SIGWINCH	Window size unchanged
SIGPIPT	Broken pipe	SIGPWR	Power failure
SIGTERM	Termination	SIGRTMIN	First real-time signal
SIGCHLD	Child status unchanged	SIGRTMAX	Last real-time signal

Table 2.5 Some Linux Signals