

Sistemas Operativos I

Introducción

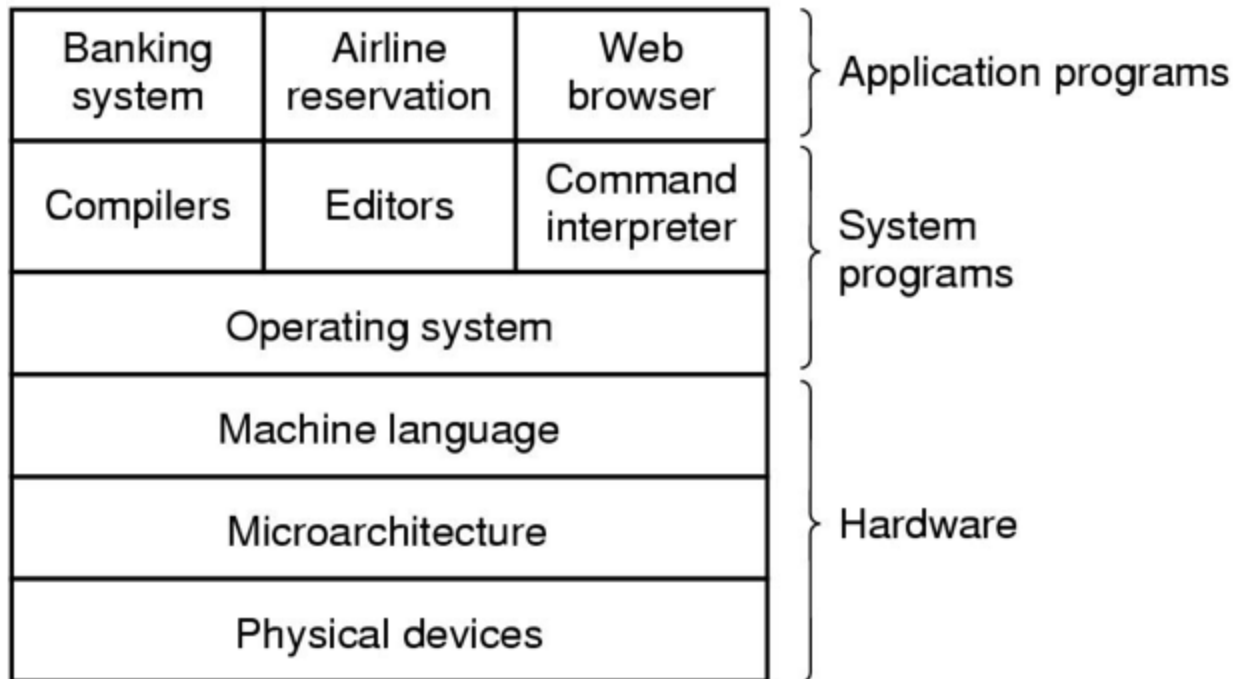
- Qué es un SO?
- Historia de los SO
- Revisión del hardware
- Conceptos de SO
- Llamadas al sistema
- Estructura de los SO

¿Qué es un Sistema Operativo?

Sistema Operativo

- Es una máquina extendida
 - Oculta los detalles tediosos y complejos que se deben realizar
 - Presenta al usuario una "máquina virtual" que es mucho más simple de usar
- Es un administrador de recursos
 - Cada programa obtiene tiempo de uso de recursos
 - Cada programa obtiene espacio de utilización de un recurso-

Componentes de un Sistema

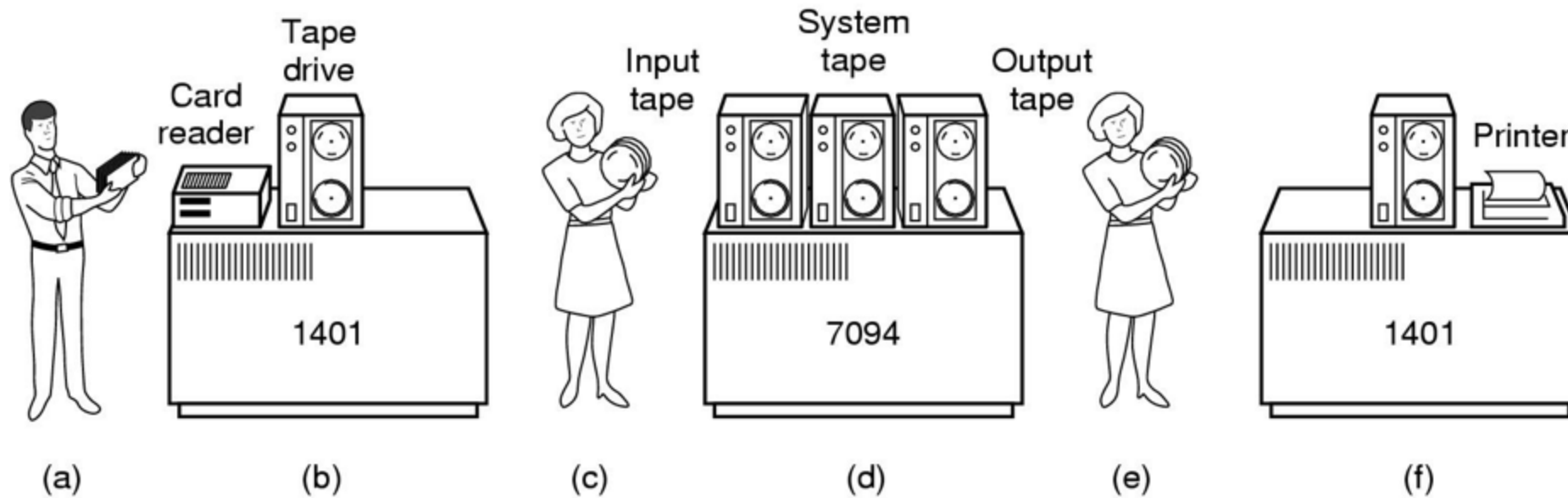


Historia

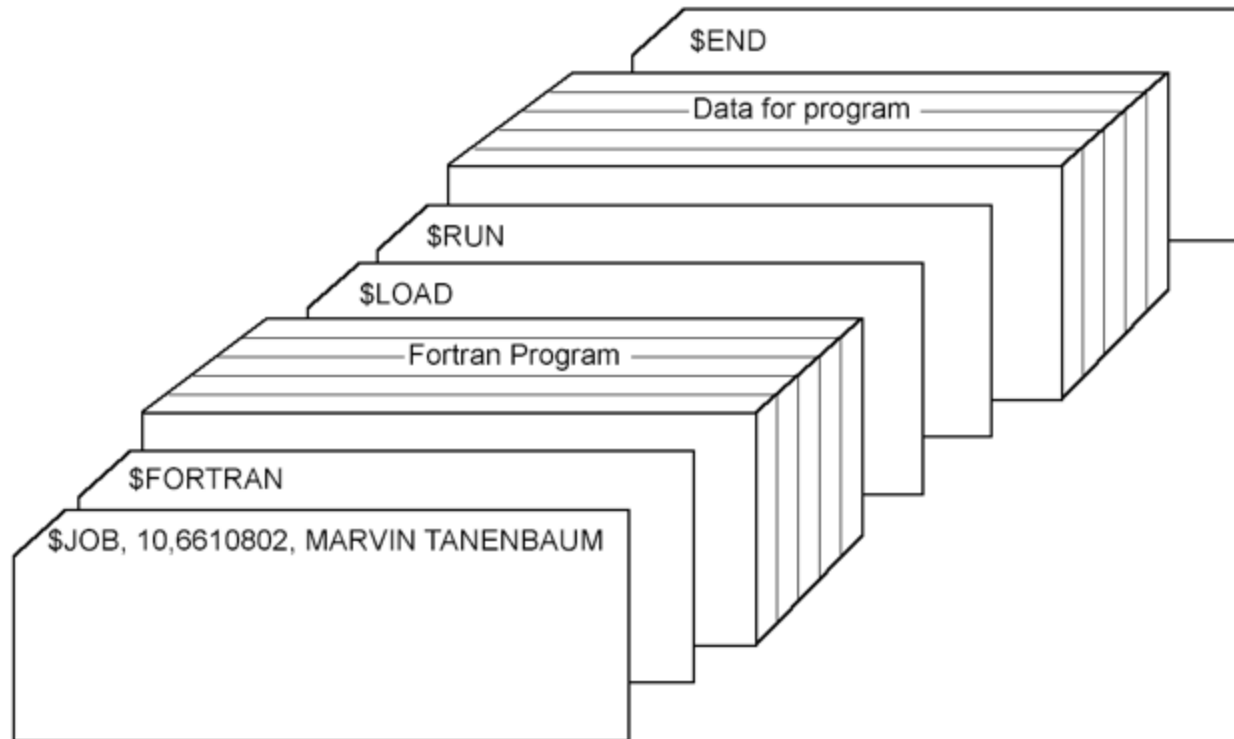
Generaciones

- Primera generación 1945 - 1955: válvulas, tableros
- Segunda generación 1955 - 1965: transistores, sistemas batch
- Tercera generación 1965 - 1980: circuitos integrados, multiprogramación
- Cuarta generación 1980 - presente: microprocesadores, computadoras personales

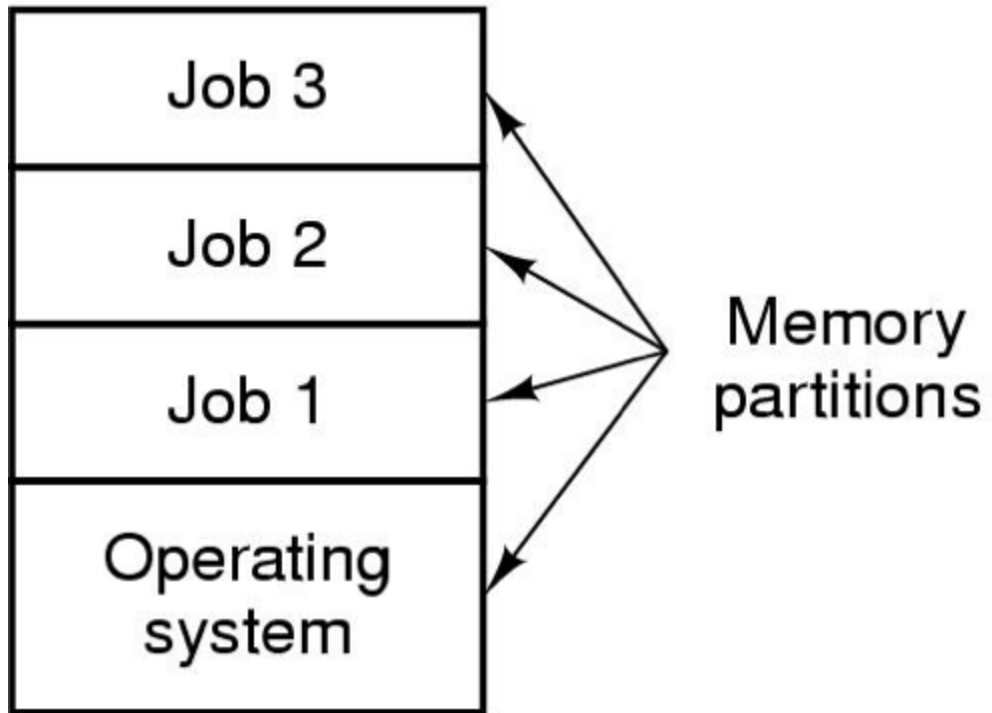
Sistemas Batch



Sistemas Batch



Multiprogramación

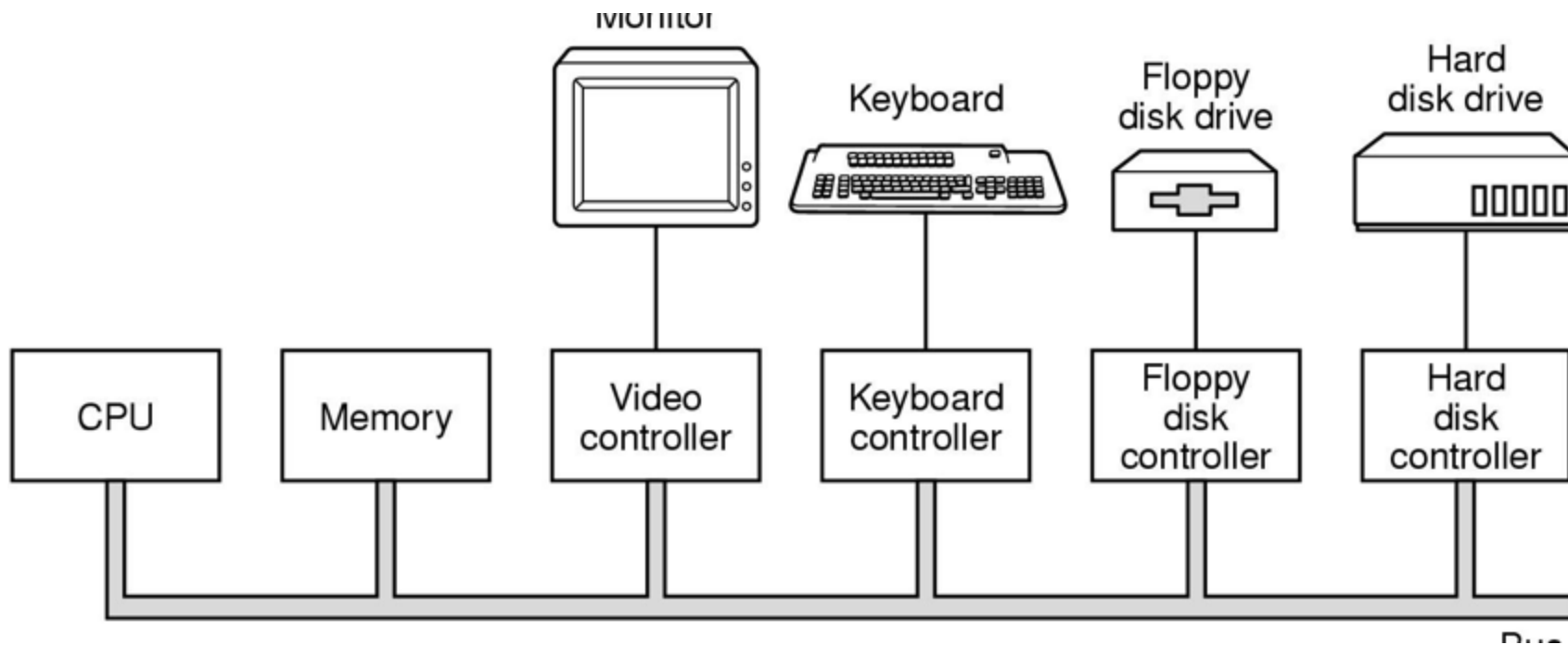


Diversidad de SO

- Mainframes
- Servidores
- Multiprocesador
- Computadora Personal
- Tiempo Real
- Embebidos

Revisión del Hardware

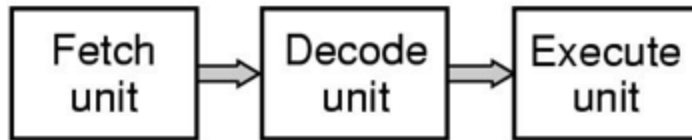
Componentes de una Computadora



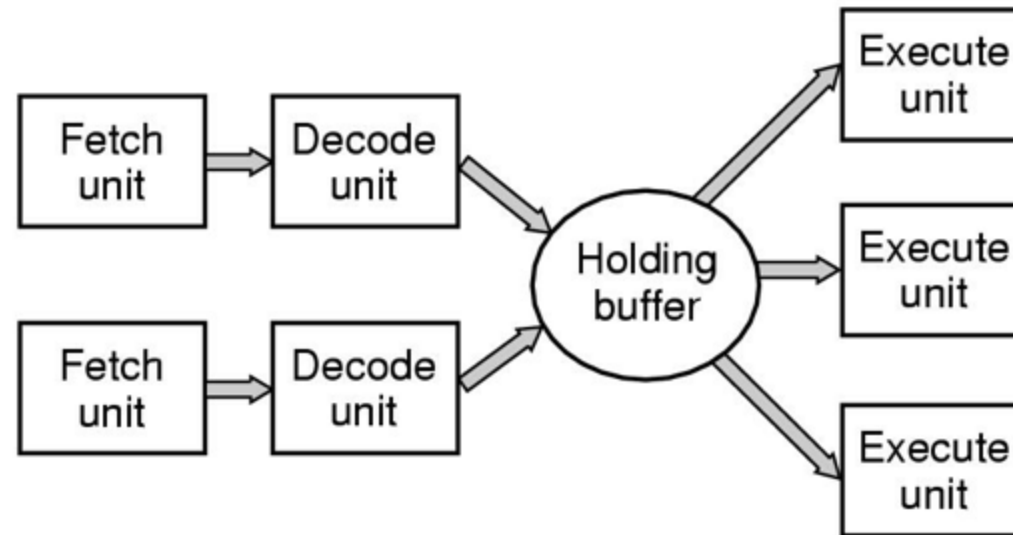
CPU

- Tomar instrucción de memoria, decodificarla y ejecutar
- Conjunto de instrucciones específico
- Registros de propósito general, contador de programa, puntero a la pila, PSW

Tipos de CPU



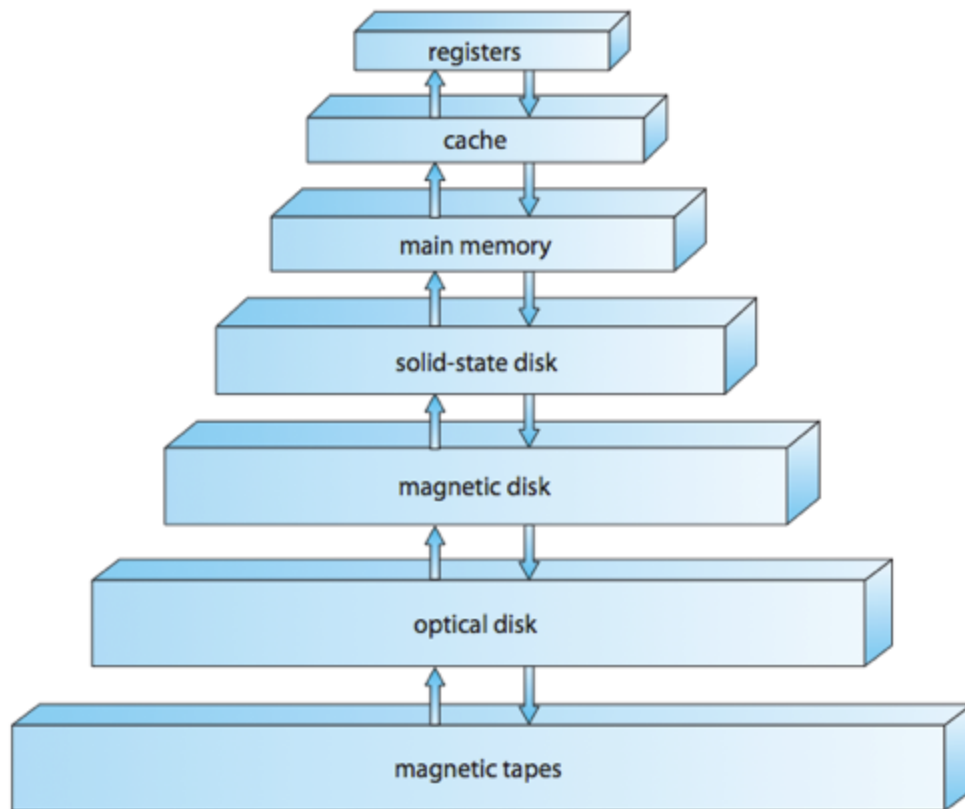
(a)



(b)

- (a) Pipeline 3 etapas
- (b) Superescalar

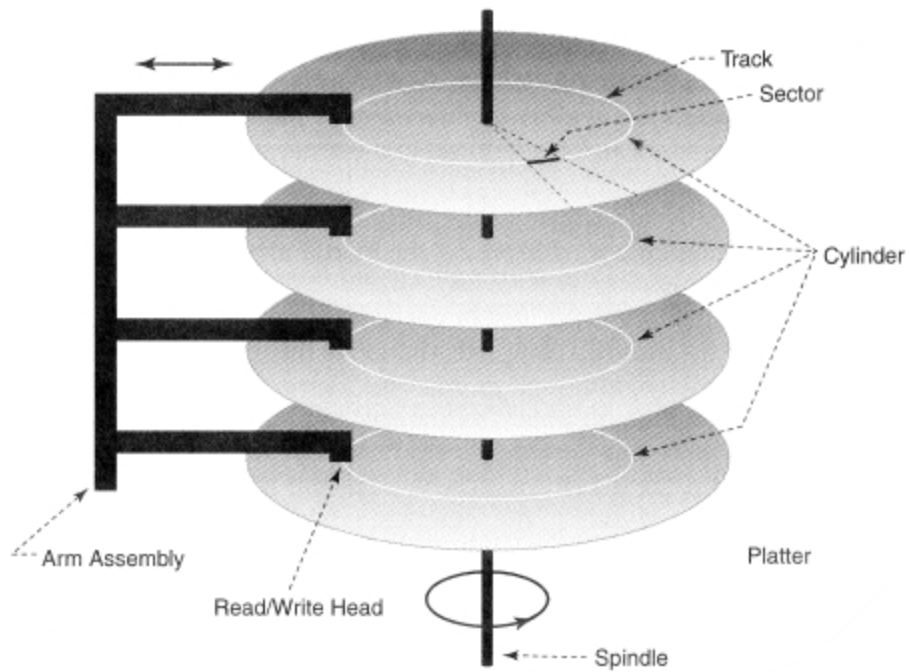
Almacenamiento



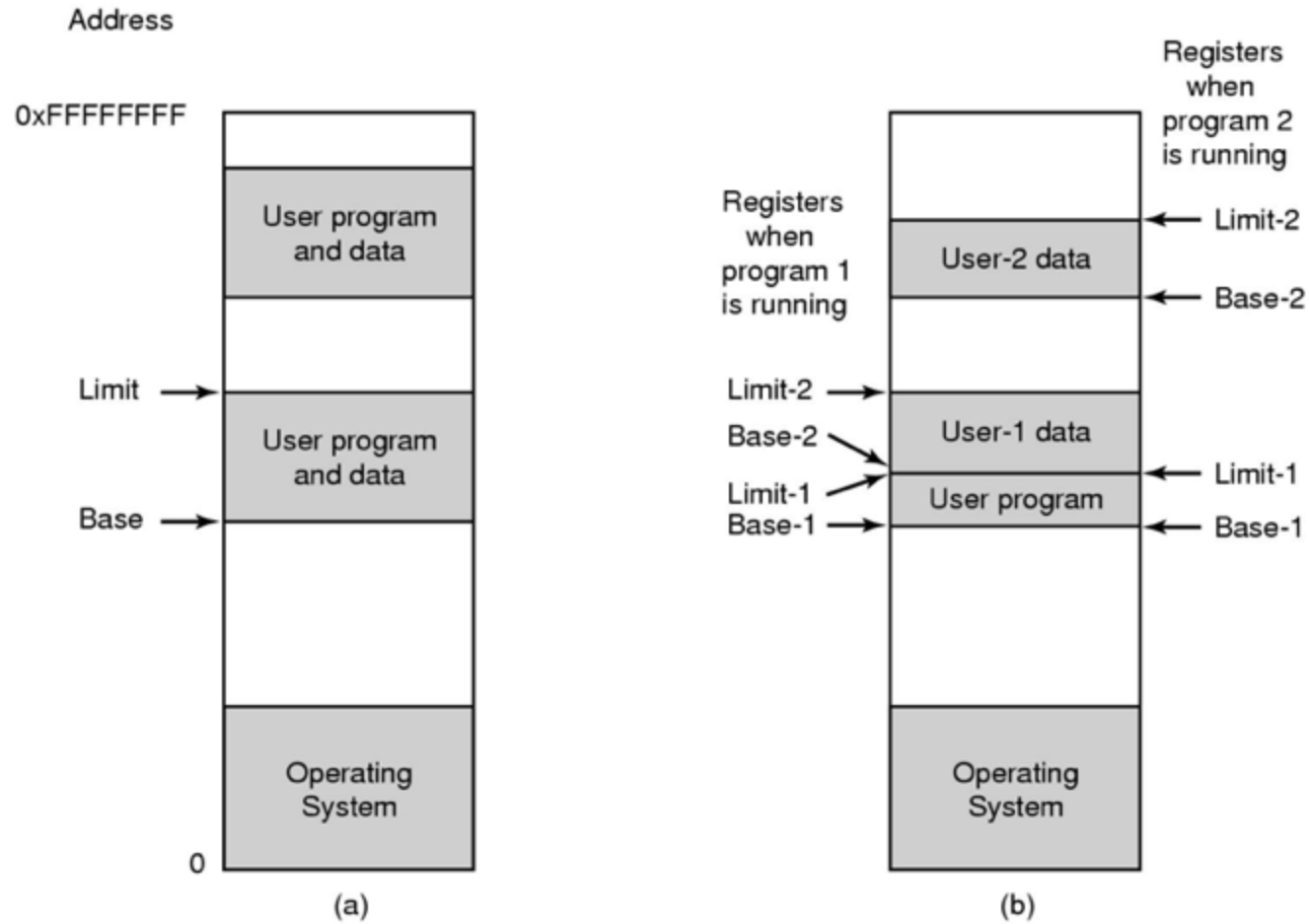
Almacenamiento

Nivel	1	2	3	4
Tipo	reg	cache	mem ppal	disco
Tamaño	< 1Kb	> 16Mb	16Gb	128Gb
Tecnología	CMOS	CMOS/SRAM	CMOS/DRAM	SSD
T Acceso (ns)	0.25	0.5	100	35000
Bandwith (Mb/s)	100000	10000	5000	250
Manejado	compilador	hardware	SO	SO

Discos Magnéticos



Memoria Principal

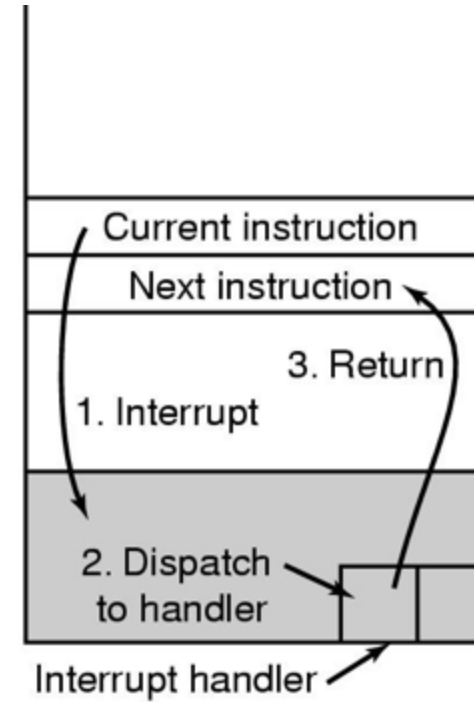
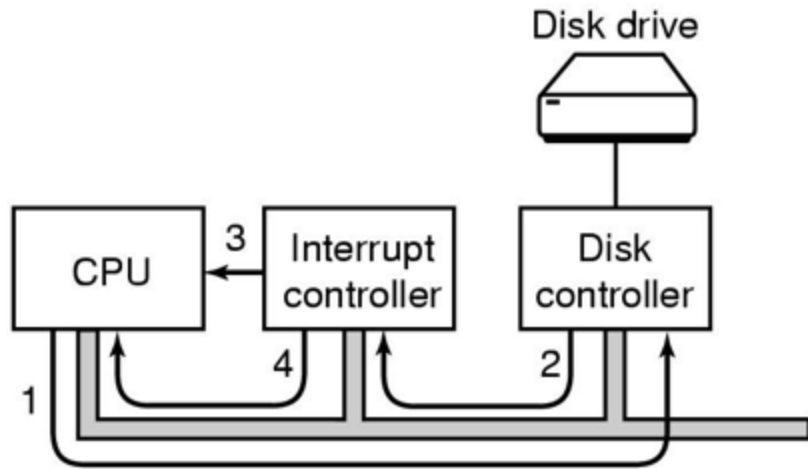


Entrada/Salida

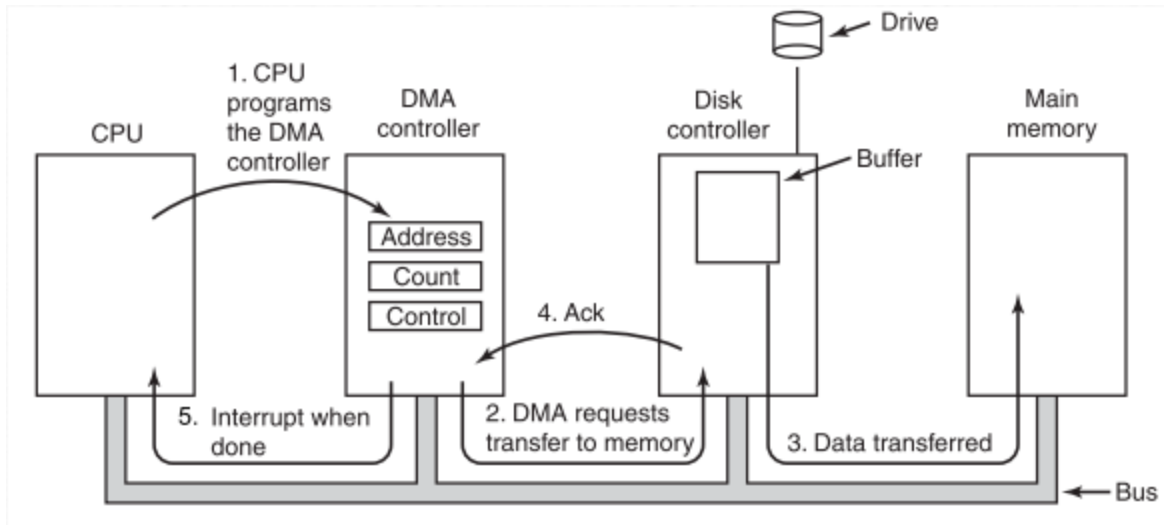
Solicitudes de E/S

- Busy waiting
- Interrupciones
- DMA

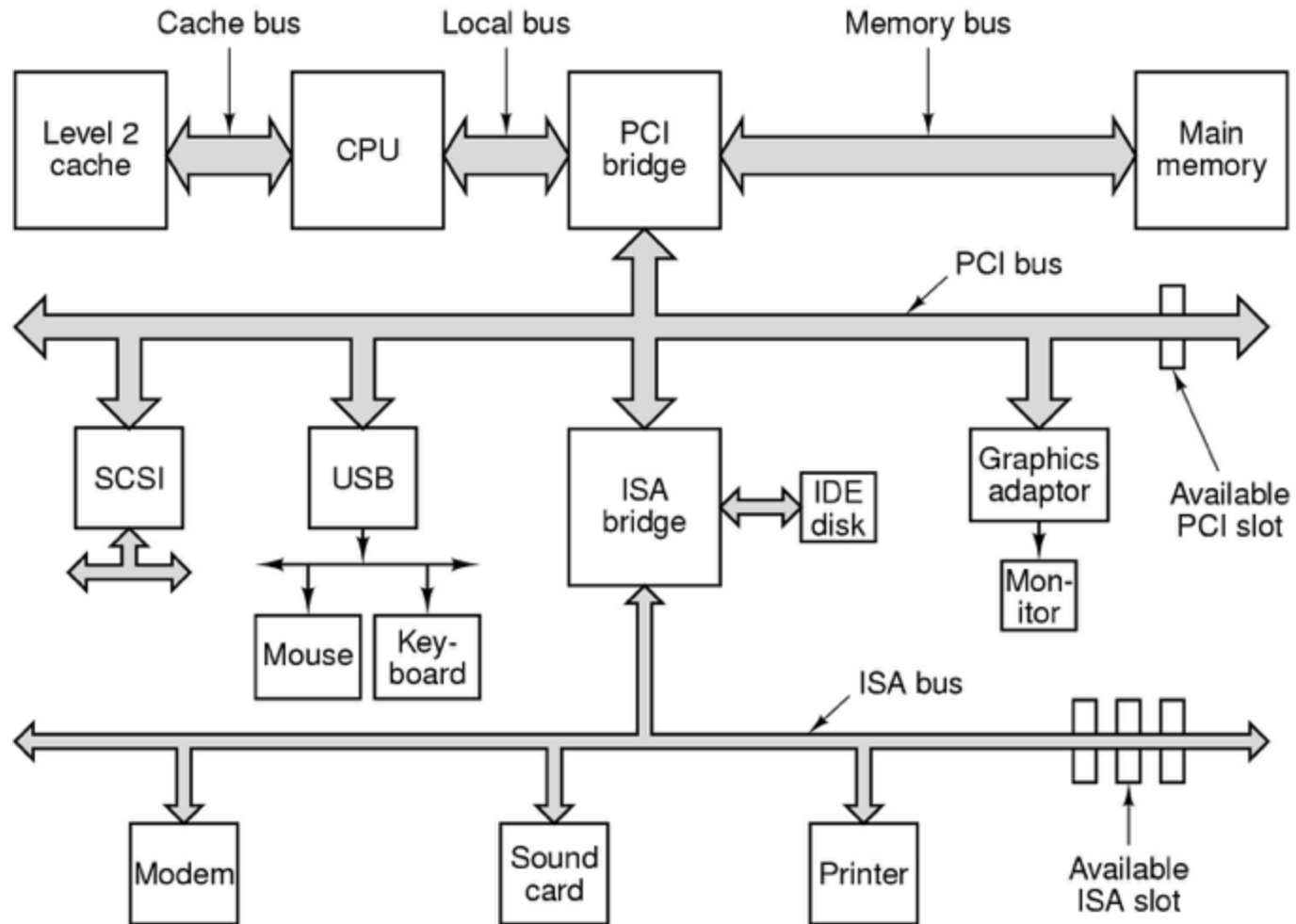
Interrupciones



DMA

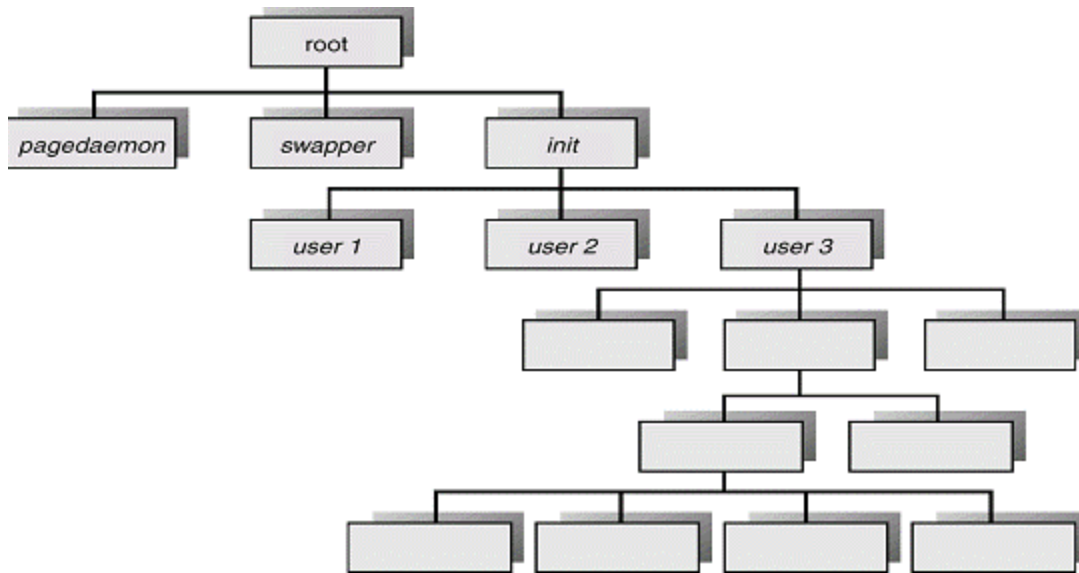


Complejidad del Hardware



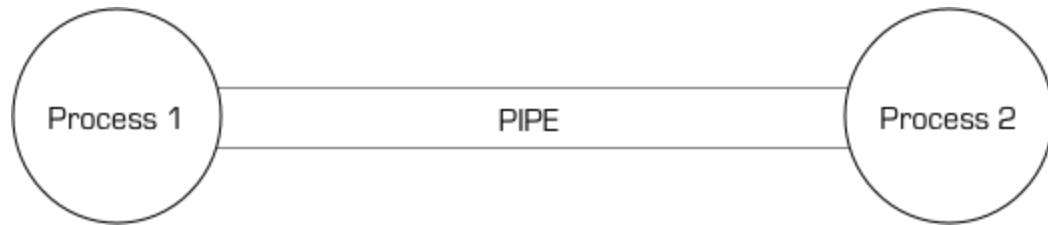
Conceptos

Procesos

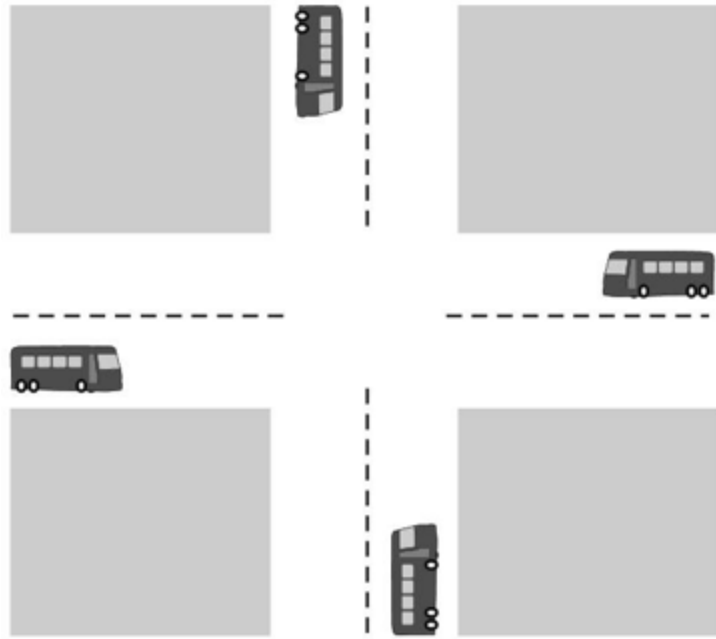


Arbol de procesos

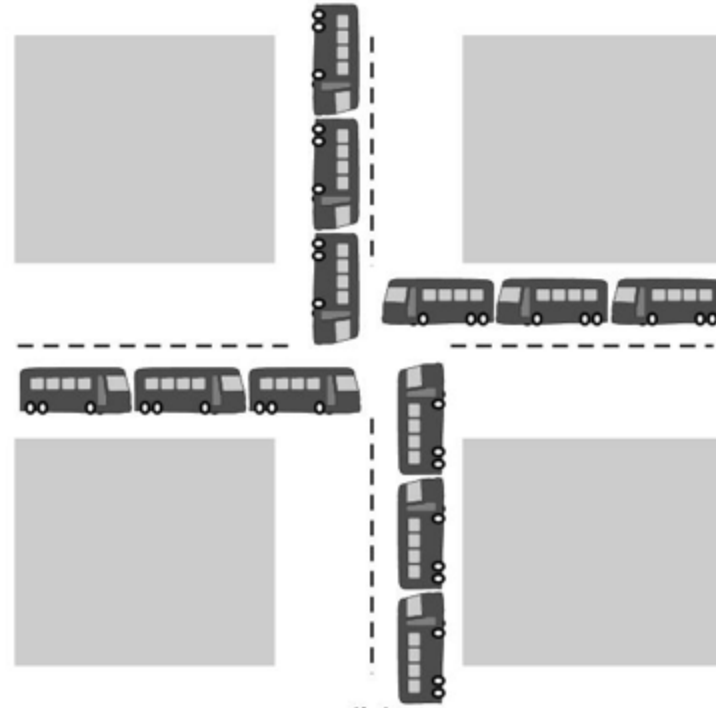
Comunicación Entre Procesos



Deadlocks

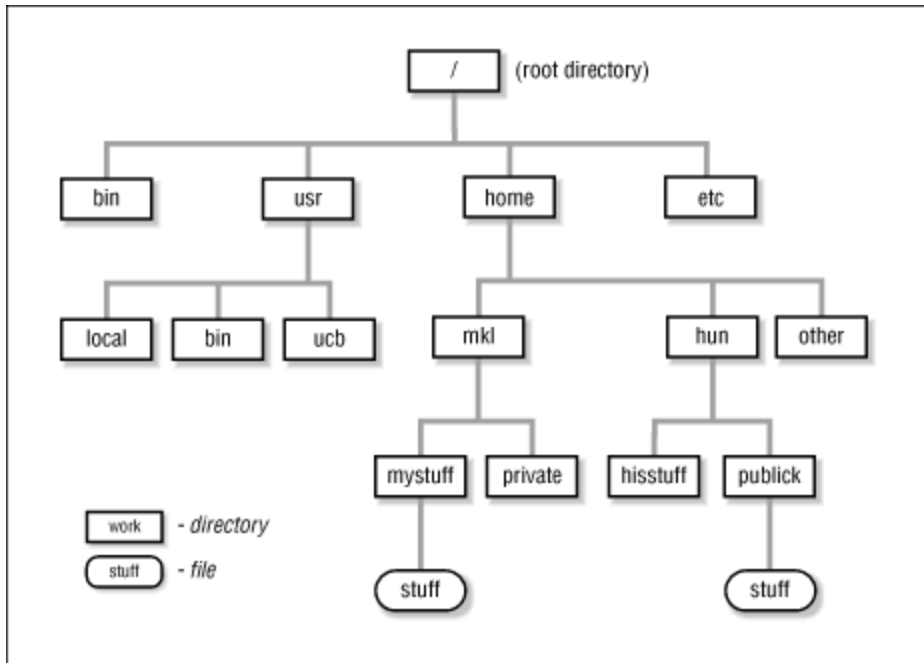


(a)

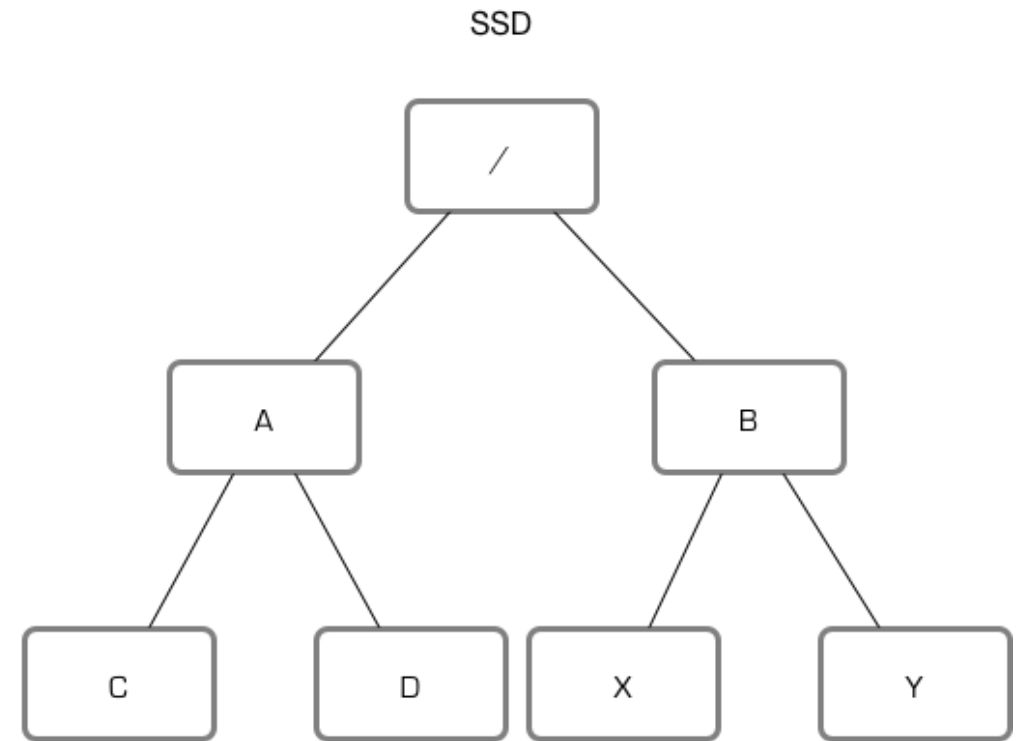
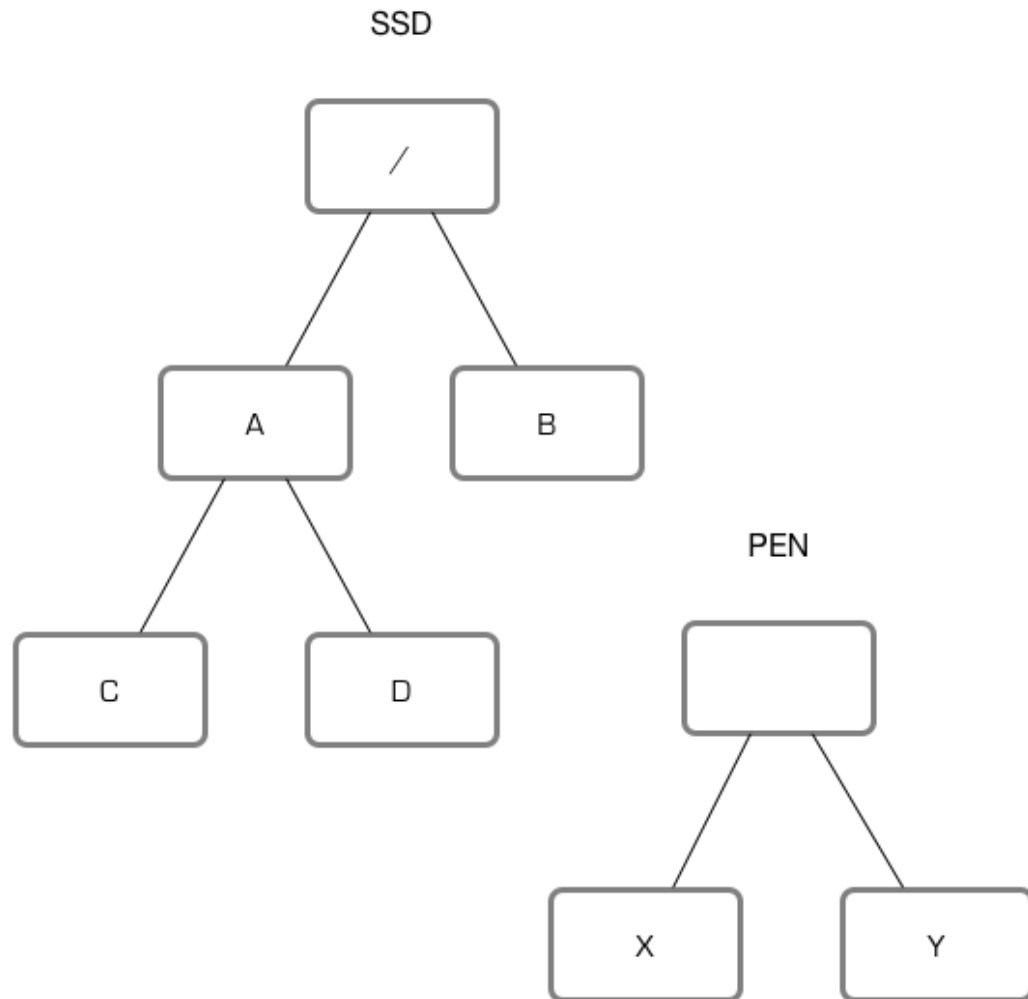


(b)

Sistemas de Archivos



Volumenes y Puntos de Montado

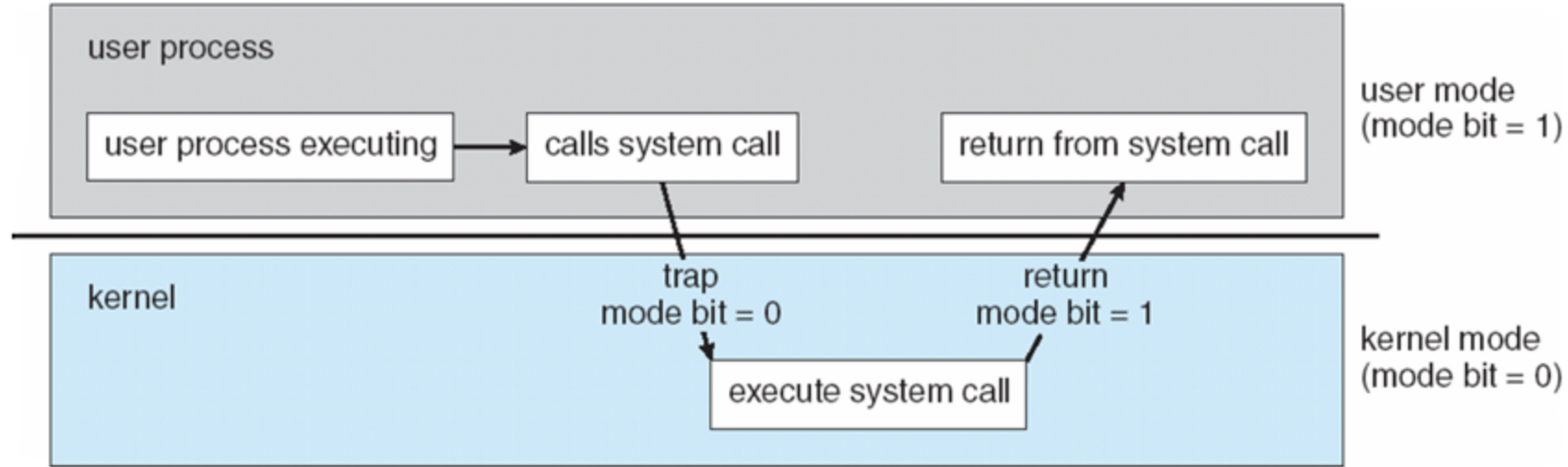


Modo Protegido

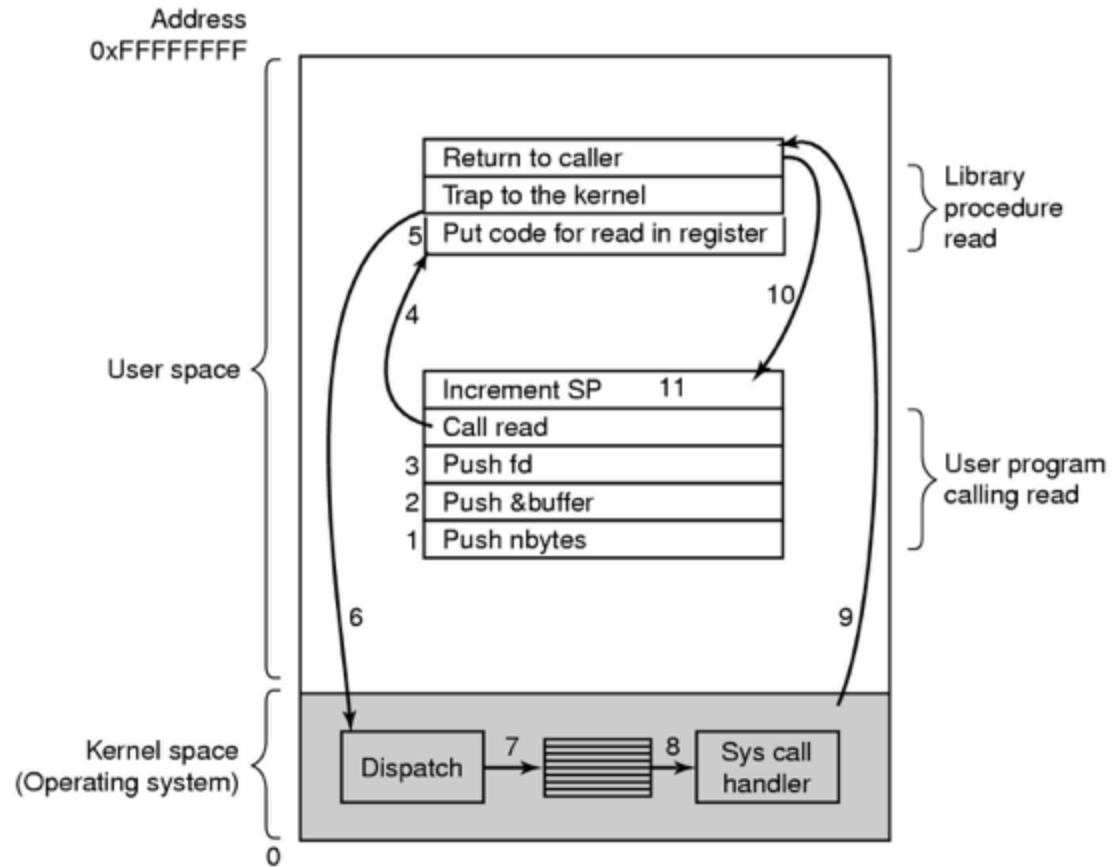
El modo dual de ejecución permite al SO protegerse así mismo y otros componentes del sistema

- 2 modos: usuario y supervisor (kernel)
- El bit de modo permite:
 - Distinguir cuando el sistema está ejecutando código del sistema o usuario
 - Algunas instrucciones designadas como privilegiadas, solo ejecutar en modo kernel
 - Las llamadas al sistema cambia a modo kernel, cuando retornan vuelven al modo usuario

Ejecución en Modo Protegido



Llamadas al Sistema



Aministración de Procesos

Llamada	Descripción
<code>pid = fork()</code>	crea un nuevo proceso idéntico al hijo
<code>pid = waitpid(pid, ...)</code>	espera que un proceso termine
<code>s = execv(name, argv, env)</code>	reemplaza la imagen del proceso
<code>exit(status)</code>	termina la ejecución de un proceso

Manejo de Archivos

Llamada	Descripción
<code>fd = open(file, ...)</code>	abre un archivo
<code>s = close(fd, ...)</code>	cierra un archivo abierto
<code>n = read(fd, buffer, nbytes)</code>	lee de un archivo a un buffer
<code>n = write(fd, buffer, nbytes)</code>	escribe de un buffer a un archivo
<code>pos = lseek(fd, offset, whence)</code>	mueve el puntero
<code>s = stat(fd, &buf)</code>	retorna información de estado de un archivo

Manejo de Archivos

Llamada	Descripción
<code>s = mkdir(name, mode)</code>	crea un nuevo directorio
<code>s = rmdir(name)</code>	borra un directorio vacío
<code>s = link(name1, name2)</code>	crea una nueva entrada <i>name2</i> apuntando a <i>name1</i>
<code>s = unlink(name)</code>	remueve una entrada
<code>s = mount(special, name, flag)</code>	monta un sistema de archivos
<code>s = umount(special)</code>	desmonta un sistema de archivos

Un Shell Simplificado

```
while (TRUE) {
    type_prompt( );
    read_command(command, parameters)
    if (fork() != 0) {
        /* Parent code */
        waitpid( -1, &status, 0);
    } else {
        /* Child code */
        execve(command, parameters, 0);
    }
}
```

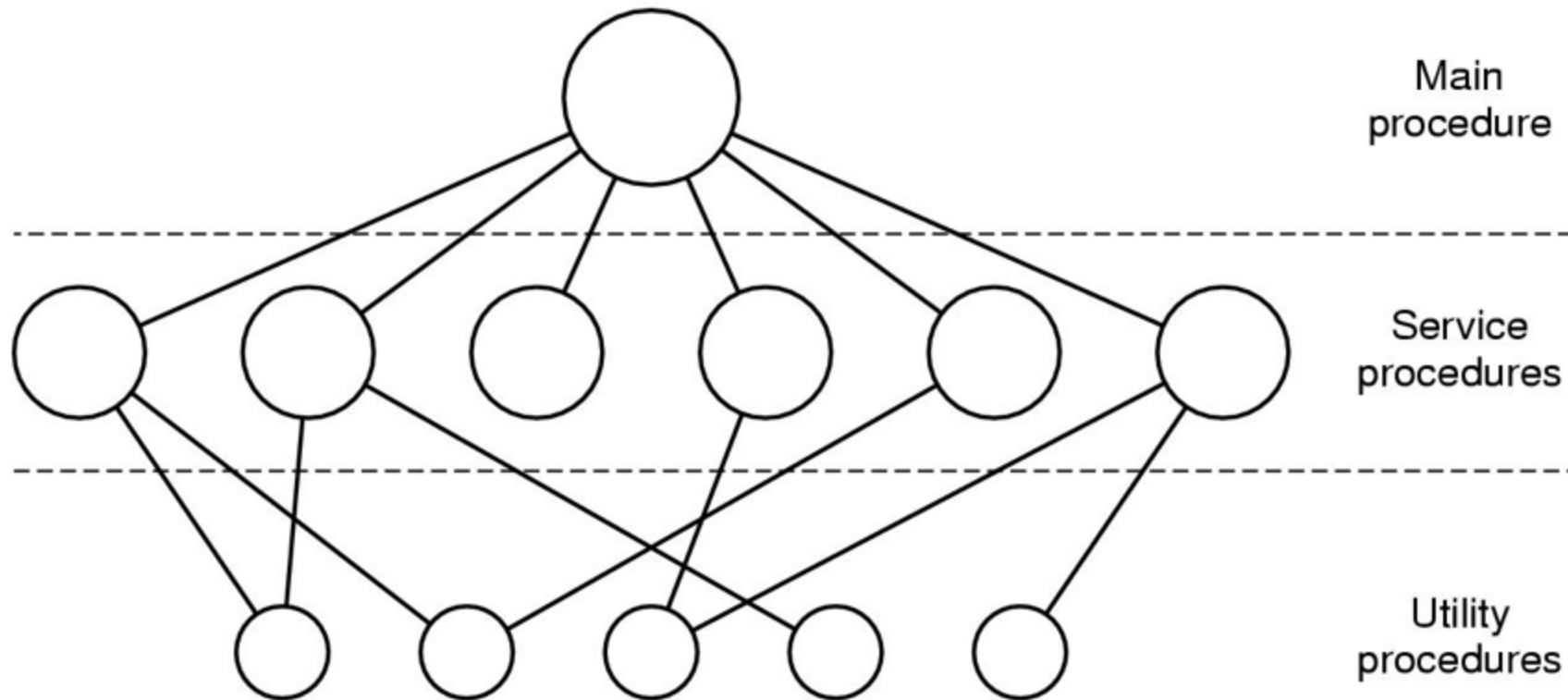
/* repite por siempre */
/* mostrar el prompt */
/* ingreso desde la terminal */
/* fork de un proceso */
/* esperar finalización del hijo */
/* ejecutar comando */

Llamadas al Sistema WIN32

UNIX	WIN32	Descripción
<code>fork()</code>	<code>CreateProcess</code>	crea un nuevo proceso idéntico al hijo
<code>waitpid</code>	<code>WaitForSingleObject</code>	espera que un proceso termine
<code>execv</code>	-	reemplaza la imagen del proceso
<code>exit</code>	<code>ExitProcess</code>	termina la ejecución de un proceso
<code>open</code>	<code>CreateFile</code>	abre un archivo
<code>close</code>	<code>CloseHandle</code>	cierra un archivo abierto
<code>read</code>	<code>ReadFile</code>	lee datos de un archivo
	

Estructura de los Sistemas Operativos

Sistema Monolítico



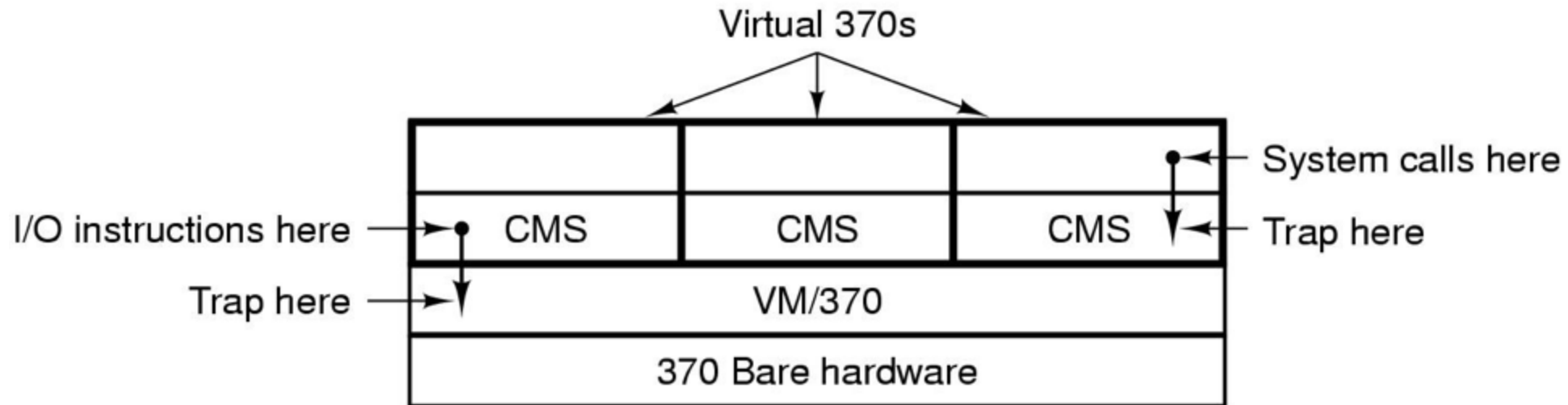
- Los programas de aplicación invocan los servicios del sistema
- Tienen un conjunto de servicios del sistema implementan las llamadas al sistema
- Un conjunto de procedimientos de utilidad que ayudan a los servicios del sistema

Estructura en Capas

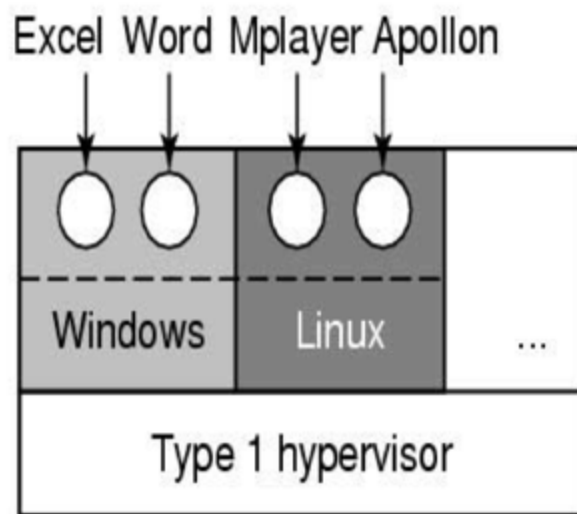
Layer	Función
5	El operador
4	Programas del usuario
3	Manejo de entrada/salida
2	Comunicación entre procesos
1	Manejo de memoria
0	Asignación de procesador y multiprogramación

- El sistema operativo está dividido en un número de capas (niveles), cada una construido sobre las capas - inferiores
- La capa inferior (capa 0) es el hardware; la capa más alta (capa N) es la interfaz del usuario
- En un sistema modular, las capas son elegidas de forma tal que cada capa usa solo las funciones y servicios de las capas inferiores

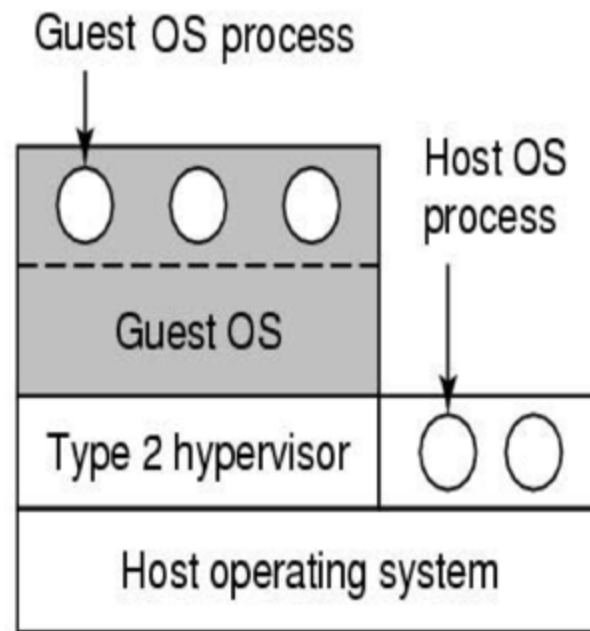
Máquinas Virtuales



- Una máquina virtual tiene una aproximación de un sistema en capas respecto de su siguiente capa lógica. Trata al sistema operativo y al hardware como si fuesen hardware
- Una máquina virtual brinda una interfaz idéntica al hardware subyacente
- El sistema operativo host crea la ilusión que cada proceso tiene su propio procesador y memoria
- A cada huésped se le brinda una copia virtual de la computadora subyacente

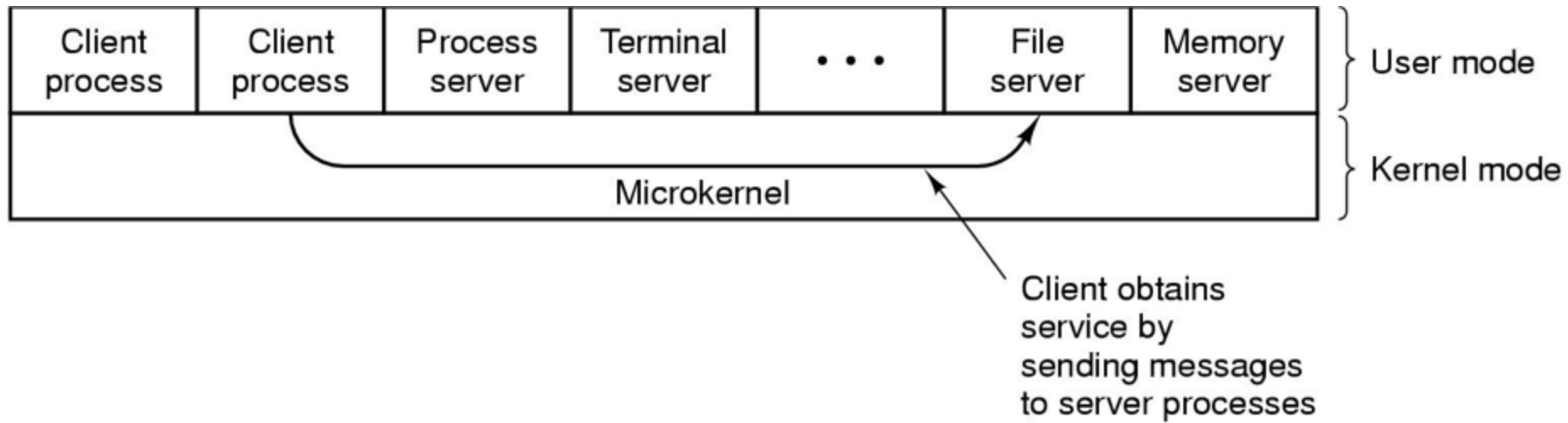


(a)



(b)

Microkernel

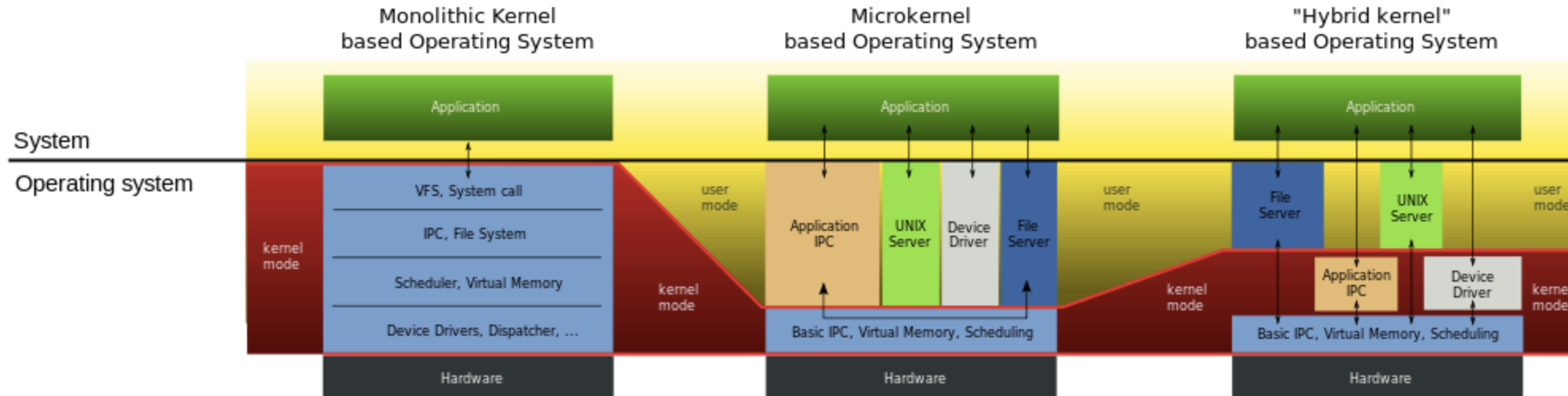


- Mueve la mayor cantidad de funcionalidad posible del kernel al espacio de "usuario"
- Solo algunas funciones esenciales quedan en el kernel: manejo primitivo de memoria (espacio de direcciones), E/S y manejo de interrupciones, Comunicación entre procesos (IPC), planificación básica
- Otros servicios del SO son provistos por procesos que ejecutan en espacio del usuario:
drivers de dispositivos, sistema de archivos, memoria virtual

La comunicación entre los módulos del usuario se hace mediante pasaje de mensajes

- Beneficios:
 - más fácil de extender
 - más fácil de portar
 - más sólido
 - más seguro
- Desventajas:
 - Hay un overhead de performance por la comunicación entre el espacio de usuario y espacio de kernel

Comparación de Estructuras



Arbol Genealógico

