

SISTEMAS OPERATIVOS I

UNIDAD I

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS



INSTITUTO TECNOLÓGICO
DE MORELIA

Departamento de Sistemas y
Computación

Disponible en: www.benito.org.mx

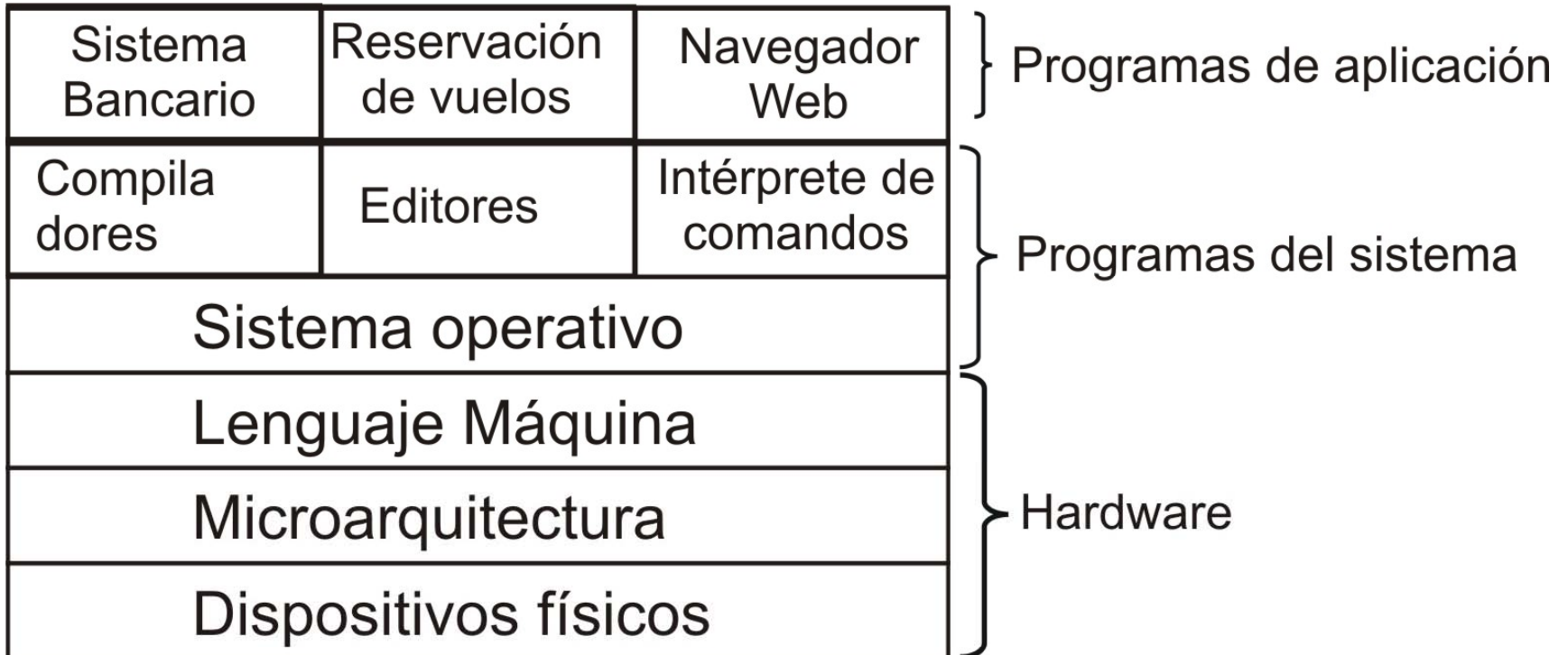
M.C. Benito Sánchez Raya

sanchezraya@hotmail.com

CONTENIDO

1. ¿Qué es un sistema operativo?
2. Historia de los sistemas operativos
3. Tipos de sistemas operativos
4. Hardware
5. Conceptos de los sistemas operativos
6. Llamadas al sistema
7. Estructura del sistema operativo

1. ¿QUÉ ES UN SO?



● NIVELES DEL SISTEMA DE CÓMPUTO:

– Dispositivos físicos:

- Circuitos integrados, cables fuentes de poder, CRT, etc.

– Microarquitectura:

- Unidades funcionales, es decir, un agrupamiento de dispositivos físicos. Como Registros, CPU, ALU, etc.

– Microprograma:

- Código que manipula los datos de los registros y sus operaciones entre ellos.
- En algunas máquinas se manipula vía hardware.

– Lenguaje Máquina

- También se le llama Arquitectura de Conjunto de Instrucciones (*Instruction Set Architecture*). Conjunto de instrucciones para manipulación del hardware, en ensamblador.
- Normalmente incluye de 50 a 300 instrucciones.

– Sistema operativo

- Capa de software que aísla precisamente esos detalles de implementación del hardware.
- Proporciona al programador el conjunto de instrucciones.

- Junto al sistema operativo encontramos, más software de sistema: Shell, compiladores, editores, etc)
- Es importante saber que estos no son parte del SO.
- El SO por lo regular es solo la porción del software que opera en *modo kernel* o *modo supervisor*, y esta protegido del usuario.
- Los compiladores y editores se ejecutan en *modo de usuario*.
- Un software que se ejecuta en modo de usuario, pudiera realizar funciones críticas del kernel.
 - Aplicación para cambio de contraseñas.

- 
- 
- Los programas de aplicación:
 - Puntos de venta, reservaciones, servicio Web, etc.

- Sistema operativo:

- a) Como máquina extendida

- Sus arquitecturas son primitivas, sobre todo en E/S.
- La idea es presentar al usuario una máquina extendida o máquina virtual.
- Presta servicios a las aplicaciones, a través de las llamadas al sistema.
- Separa al programador del hardware: alejándolo de interrupciones, temporizadores, administración de memoria, etc.
- Ejemplo:
 - Floppy PD765: 16 instrucciones, en registros de 1 a 9 bytes. Read y Write requiere 13 parámetros en 9 bytes. Devuelve 23 campos de estado y error en 7 bytes.

- Sistema operativo:

- b) Como administrador de recursos

- Recursos: procesadores, memorias, temporizadores, discos, ratón, tarjeta de red, impresoras, etc.
 - Tarea: Efectuar un reparto ordenado y controlado de los recursos, entre los diversos programas que compiten por ellos.
 - Cuando hay muchos usuarios se debe administrar y proteger el uso del CPU, la memoria y dispositivos de E/S.
 - Se debe compartir hardware e información; para lo cual se debe saber, quienes compiten, que recursos usan, conceder solicitudes, dar cuenta de su uso, mediar entre solicitudes de programas y/o usuarios en conflicto.

- Administrador de recursos por Multiplexión:
 - a) Multiplexión de recursos en el tiempo
 - Diferentes programas y usuarios se turnan para usarlo
 - El SO determinará los criterios para el uso de recursos
 - ¿Quién sigue?, ¿Cuánto tiempo?
 - Ejemplo: Cola de impresión.

b) Multiplexión de recursos en el espacio

- En lugar de que se turnen por el uso del recurso, cada uno recibe una parte de él.
- Ejemplo:
 - La RAM se divide entre los programas en ejecución, siendo todos residentes al mismo tiempo.
 - Suponiendo que haya suficiente memoria para contener varios programas, sería mas eficiente tener varios programas en la memoria, que darle toda la memoria a uno de ellos, sobre todo que sólo la usará una fracción del total.
- Requiere manejo de problemas de equidad, protección, etc.
- Otro ejemplo:
 - El uso del disco duro.

2. HISTORIA DE LOS SO

- Primera computadora digital:
 - Diseñada por Babbage, la máquina analítica. Nunca se logró que funcionará. Falta de tecnología para producir los engranes.
 - Primera programadora del mundo: Ada Lovelace, contratada para programar por Babbage.
 - El lenguaje de programación toma su nombre en su honor.

● **Primera Generación (1945 – 1955), Tableros y tubos de vacío**

- Las primeras empleaban relevadores mecánicos
 - Lentas, con ciclos medidos en segundos.
- Relevadores → Bulbos
 - Máquinas enormes
 - Miles de bulbos
 - Todo se realizaba en lenguaje máquina
 - Un solo grupo de personas:
 - Diseñaba, construía, programaba, operaba y mantenía cada máquina.

- Tableros alambrados
- No existían lenguajes de programación, ni ensamblador, ni los sistemas operativos.
- Se inicia la introducción de tarjetas perforadas.

- **Segunda Generación (1955 – 1965), Transistores y sistemas por lotes**
 - Bulbos → Transistores
 - Hubo distinción entre diseñadores, constructores, operadores, programadores y personal de mantenimiento.
 - Máquinas llamadas **mainframes** o **macrocomputadoras**.
 - Sólo pocas empresas o gobiernos las tenían.



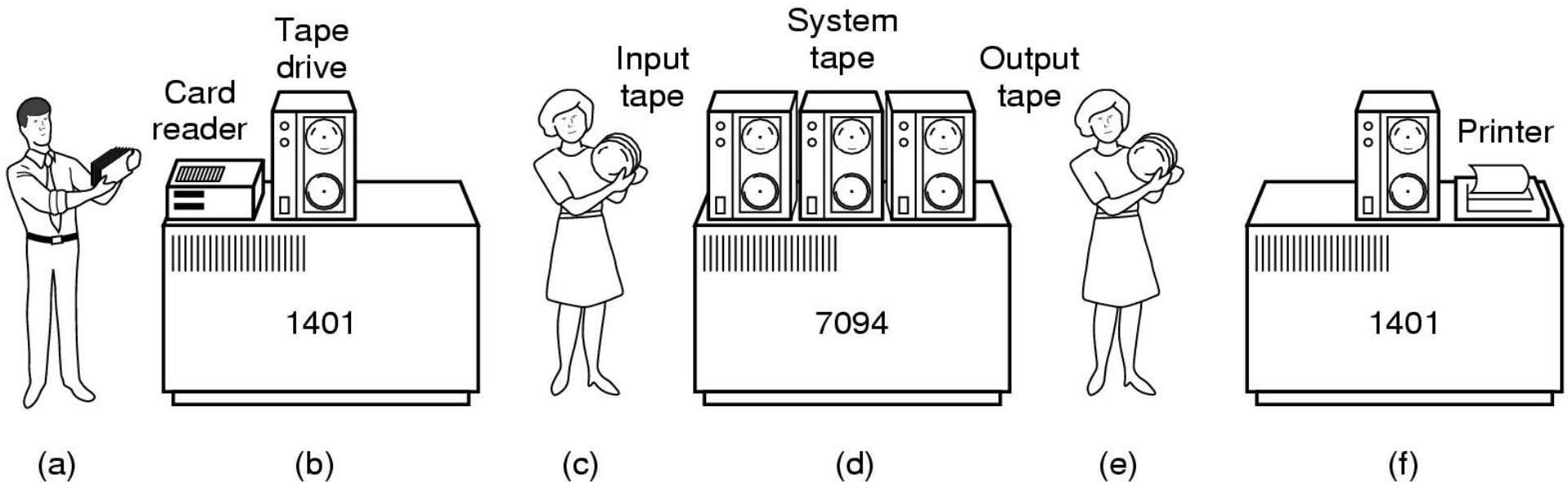
– Operación:

1. Escribían el programa en FORTRAN o ensamblador.
2. Luego lo perforaba en tarjetas.
3. Llevaba el montón de tarjetas al cuarto de entrada.
4. Entregaba a los operadores y esperaba hasta que saliera el resultado.
 - Se desperdiciaba mucho tiempo de computadora.
 - El sistema por lotes vino a mejorar estos tiempos:

– Sistema por lotes

- En la 7094 se ejecutaba el antecesor de los sistemas operativos.

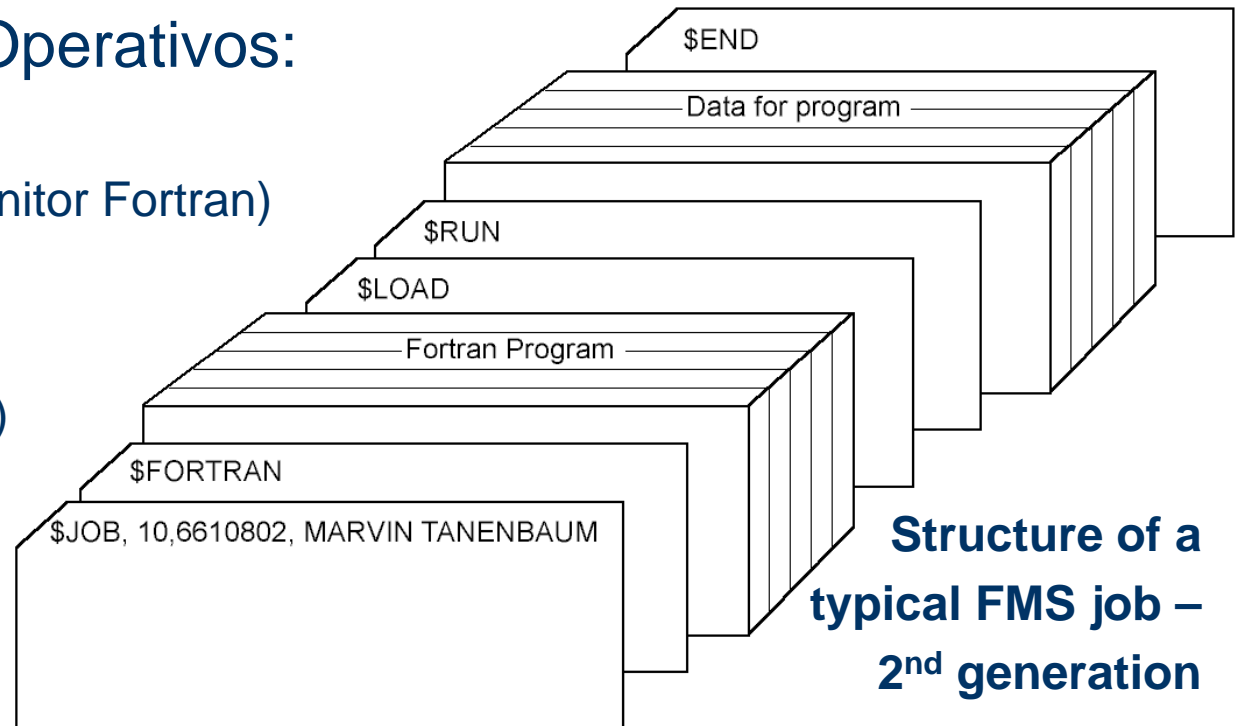
- Encargado de leer los trabajos de la cinta y pasar el resultado a la cinta de salida.



– Proceso:

1. Una tarjeta \$JOB, que especificaba el tiempo de ejecución máximo en minutos
2. Luego una tarjeta \$FORTRAN, especificando al SO que debía cargar el compilador de FORTRAN de la cinta del sistema.
3. Después el programa a compilar.
4. Luego una tarjeta \$LOAD para indicar al SO que cargase al programa objeto recién compilado.
5. A continuación una tarjeta \$RUN, que pedía al SO que ejecutase el programa con los datos contenidos en dicha tarjeta.
6. Al final la tarjeta \$END, indicando el fin del trabajo.

- Estas tarjetas de control fueron las precursoras de los lenguajes de control de trabajos e interpretes.
- Sistemas Operativos:
 - FMS
(Sistema Monitor Fortran)
 - IBSYS
(SO de IBM para la 7094)



● **Tercera Generación (1965 – 1980), Circuitos integrados y multiprogramación**

- Dos líneas de computadoras diferentes e incompatibles entre si:
 - Computadoras científicas a gran escala (7094)
 - Para cálculos numéricos en ciencias e ingeniería
 - Computadoras comerciales (1401)
 - Uso más común, manejo de cintas e imprimir
- IBM trató de solucionar esas incompatibilidades con las computadoras del sistema 360.

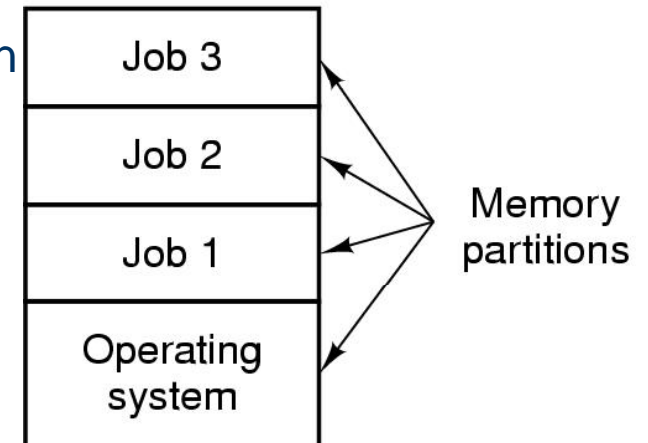
– Sistemas 360

- Equipos de distintos rendimientos y capacidades, de pequeños a grandes, compatibles entre si, con la misma arquitectura y conjunto de instrucciones.
- Siguieron las series: 370, 4300, 3080 y 3090, compatibles con la serie 360.
- La mayor ventaja de la idea de una sola “familia”, se convierte en su punto más débil.
 - Que todo el software incluido en el sistema operativo OS/360 debía funcionar en todos sus modelos.
 - Fue difícil que lo lograría, quedó enorme, complejo y con muchos errores; tres veces más que FMS.

Características de los SO de 3a Generación

– Multiprogramación:

- En procesos que involucraban E/S, se llegaba a desperdiciar hasta el 90% el tiempo de procesador.
- La idea era mantener al 100% ocupado el costoso CPU.
 - Dividir la memoria en varias partes
 - Colocar un trabajo en cada partición
 - Requiere hardware especial que proteja cada trabajo contra espionaje y acciones hostiles.
- Se iban ejecutando uno a uno.



– Spooling

- Spool (*Operación simultanea de periféricos en línea*)
- Consiste en la capacidad de cargar uno a uno los trabajos del disco duro.
- Una vez que se desocupaba una partición, se colocaba el siguiente, y así sucesivamente en cada partición.
- Esto permitió desechar las 1401.
 - Ya no se requerían.
- Más enfocado a E/S

– Tiempo compartido

- Los dos conceptos anteriores siguen siendo por lotes, es decir, un usuario no tenía para sí sólo la máquina.
- Cada usuario tiene una terminal en línea.
- El tiempo se divide solamente entre los trabajos que desean ser atendidos, aunque haya muchos más en ejecución.
- El primer sistema de tiempo compartido:
 - CTSS (compatible time sharing system)
 - En el MIT en una computadora 7094
 - Sin embargo se populariza hasta esta generación

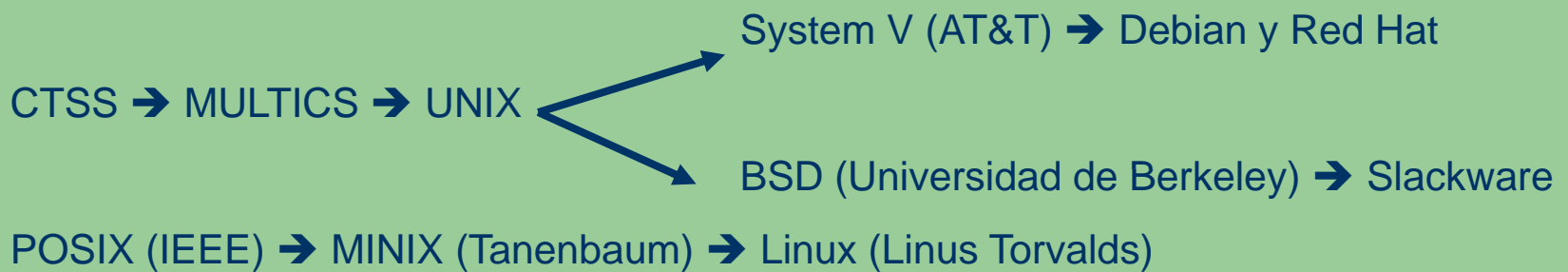
- Ejemplo:

- MULTICS (Servicio de Información y Computación MULTiplexado)
- Creado por MIT, Bell Labs y GE
- Proyecto ambicioso, para cientos de usuarios.
- No funciona del todo, se separan GE y Bell Labs.
- El MIT lo termina solo.
- Escrito en PL/I

– Minicomputadoras

- Serie PDP (PD-1 a PDP11)
- Incompatibles entre si.
- MULTICS → Unix
- Al dejarse el código abierto de Unix, surge el caos:
 - Versiones distintas e incompatibles.
 - Se crean dos vertientes:
 - System V, de AT&T
 - BSD (Berkeley Software Distribución), de la Universidad de Berkeley.
- IEEE crea un estándar para compatibilidad de los Unix:
 - POSIX

- En 1987 Tanenbaum desarrolla Minix:
 - Es un POSIX educativo
 - Disponible en la Web
 - Compatible con POSIX
- Linus Torvalds desarrolla Linux, a partir de Minix
 - Estudiante Finlandés
 - Creando una versión libre, mas allá de solo educativa.





- **Cuarta Generación (1980 a la fecha),
computadoras personales**

- Era de las computadoras personales
- En 1974 Intel presenta el 8080:
 - Primero de 8 bits de propósito general (74 instrucciones)
 - Intel pide a uno de sus consultores (Digital Research) que escriba un SO para este procesador.
 - Gary Kildall lo desarrolla, CP/M (Programa de Control para Microcomputadoras)
 - Fue la primer microcomputadora con disco.

– CP/M

- En 1977 Digital Research reescribe el CP/M para otras computadoras personales distintas al 8080, (Zilog Z80).
- Dominó por 5 años las microcomputadoras

– IBM PC

- A inicios de los 80s se desarrolla.
- IBM buscaba software para su PC
- Contacta a Bill Gates para usar bajo licencia su interprete de BASIC.
- IBM le pregunta a Gates que le recomiende un SO para su computadora.

- Gates le aconseja a IBM ir con Kildall.
- Kildall no firma con IBM:
 - Envía un subordinado
 - Su abogado no firma por no querer mantener la confidencialidad del equipo (IBM-PC) que todavía no salía
- IBM regresa con Gates
 - Para ver si le puede ofrecer un SO.
 - Gates compra (50000 USD) el DOS a Seattle Computer Products
 - Y se lo vende a IBM junto con el BASIC
 - Gates contrata a quien se lo vendió para hacerle cambios que IBM pidió.

- Se renombra a MS-DOS
- Gates lo vende a compañías, Kildall sólo lo vendía a usuarios.
- A partir de ahí se da el boom del MSDOS
 - En 286, 386 y 486.
- Microsoft toma partes de Unix
 - Desarrolla un Unix → Xenix

– GUI (Interfaz Grafica de Usuario)

- En los 60's por Doug Engelbart
- Ventanas, iconos, menús, ratón.
- Lo adopta Xerox en sus equipos
- Lo adopta Steve Job en la Apple (Lisa) y Apple Macintosh.
- De 1985 a 1995 Microsoft lo incorpora pero más que SO fue sólo un Shell gráfico.
- Windows 95, 98, Me, XP, Vista, 7.
- Windows NT, 2000, 2003, 2008

- En el mundo Unix
 - Sistema X Windows
 - Unix/Linux esta reemplazando a Microsoft en áreas especificas.
- Sistemas operativos
 - De red
 - Pocos cambios (módulos para manejo de sesión, tarjeta de red, etc)
 - Cada computadora tiene su SO local y su usuario local
 - Distribuidos
 - Interfaz como si fuese monoprocesador
 - Ejecución en múltiples procesadores
 - Paralelismo
 - Requiere cambios radicales
 - Ejemplos: Plan 9 e Inferno.



- Se repite la Historia:

- Mainframe, minicomputadoras, microcomputadoras, PDAs, dispositivos móviles, etc)

- Siguen la misma tendencia:

- Iniciaron con lenguaje máquina o ensamblador.

- Hardware y software limitado.

- Capacidad de almacenamiento, de procesamiento, de Ram, etc.

3. TIPOS DE SISTEMAS OPERATIVOS

- Tipos de Sistemas Operativos:
 - a) De mainframe
 - b) De servidor
 - c) Multiprocesador
 - d) De computadora personal
 - e) De tiempo real
 - f) Integrados
 - g) De Tarjeta Inteligente



- Tipos de Sistemas Operativos:

- a) De mainframe

- Manejo de enormes capacidades de E/S.
- Servicios:
 - a) Lotes (No hay usuario interactivo)
 - b) Procesamiento de transacciones (en línea, pequeñas pero muchas)
 - c) Tiempo compartido (Consulta de usuarios remotos)
- Ejemplos: OS/360 y OS/390. Y los del top500.

b) De servidor

- Se ejecutan en servidores
 - Web, FTP, Mail, impresión etc.
- Ejemplos: Unix, Linux, Windows 2000/2003/2008

c) Multiprocesador

- Para computadoras paralelas (multicomputadoras o mutiprocesadores)
- Clúster
- Ejemplos: Son variantes de los de servidor

d) De computadora personal

- Presentar una interfaz amigable a un solo usuario
- Para aplicaciones de ofimática e internet
- Ejemplos: Windows 9x-Xp-Vista, Linux, MacOS

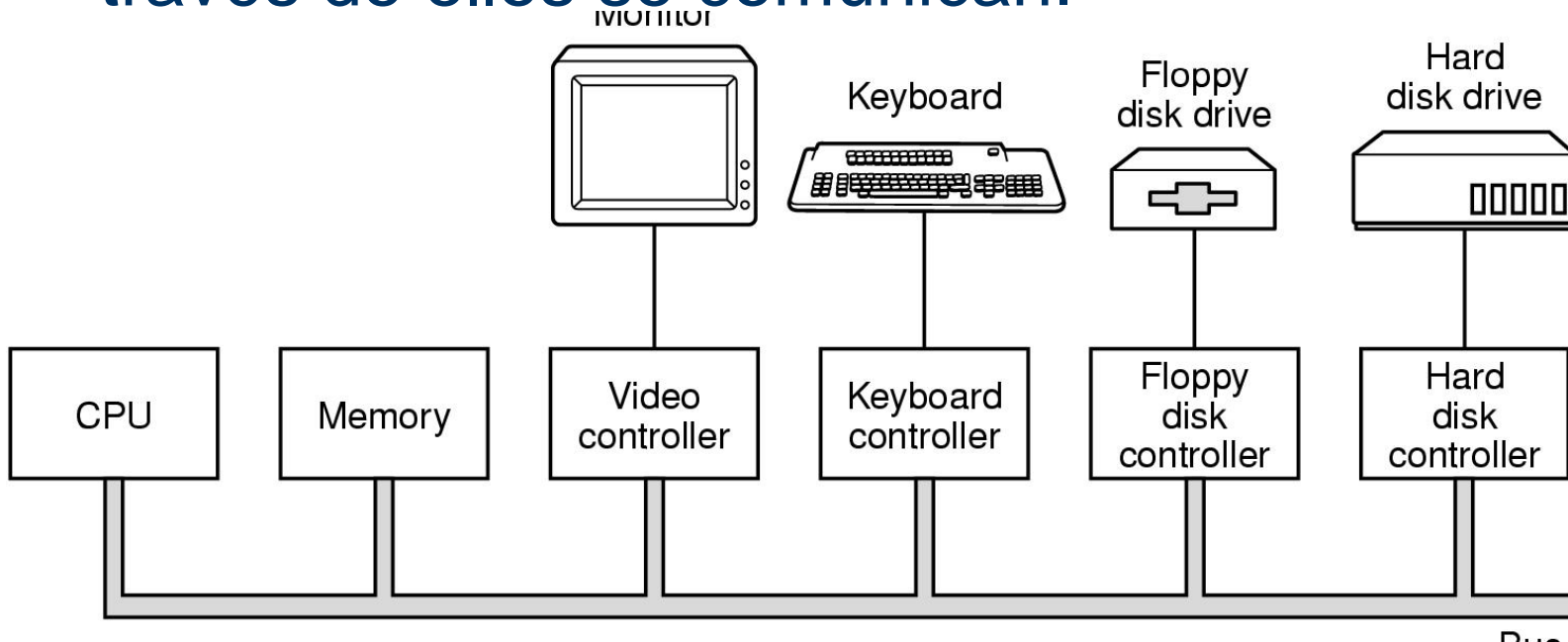
e) De tiempo real

- Parámetro clave el tiempo, en sistemas industriales y multimedia.
- Tipos:
 - Sistemas de tiempo real riguroso
 - Indispensable que la acción se ejecute en cierto momento.
 - Sistemas de tiempo real no riguroso
 - Aceptable que de vez en cuando no se cumpla en tiempo
- Ejemplos: VxWorks(Honda ASIMO) y QNX

- f) Integrados/empotrados
 - Usados en PDAs, dispositivos móviles pequeños, electrodomésticos, etc.
 - Emplean algunas características de los de tiempo real
 - Limitantes de hardware, software y energía.
 - Ejemplos: Web OS (antes Palm OS), Windows Mobile, Symbian, etc
- g) De Tarjeta Inteligente
 - Se ejecutan en microchips o tarjetas inteligentes
 - Limitantes de hardware y software.
 - Usos: pagos electrónicos, tarjetas de ocasión, juguetes, etc.
 - Usan Java (interprete JVM)
 - Se descargan los applets y el interprete de JVM los ejecuta.
 - Pueden manejar múltiples applets (multiprogramación, calendarización, protección de recursos, etc)

4. HARDWARE

- El CPU, la memoria y los dispositivos están conectados por varios buses de sistema y a través de ellos se comunican.



- Procesadores:

- Pasos:

- Tomar la 1ª instrucción de la memoria
 - La decodifica (chechar tipo y operandos)
 - La ejecuta
 - Tomar decisiones
 - Toma la 2ª instrucción . .
 - . .

- Conjunto de instrucciones:

- Cada CPU ejecuta un conocido y único.
 - 4004 → 46 instrucciones
 - 8080 → 74 instrucciones

- Registros

- 8080 → 14 registros de 16 bits
 - Itanium → 384 registros



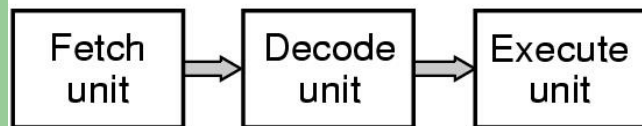
– Registros:

- Contador de programa
 - Dirección de memoria donde esta la siguiente instrucción a ejecutar.
- Apuntador de pila
 - Apunta a la parte superior de la pila actual en memoria.
 - La pila contiene un marco con: parámetros de entrada, variables locales, y variables temporales
- Palabra de estado del programa (PSW)
 - Contiene el código de condición (para instrucciones de comparación)
 - Verificando: Prioridad de CPU, modo (kernel o usuario), etc
 - Los programas de usuario leen toda la PSW, pero solo escriben en una parte.

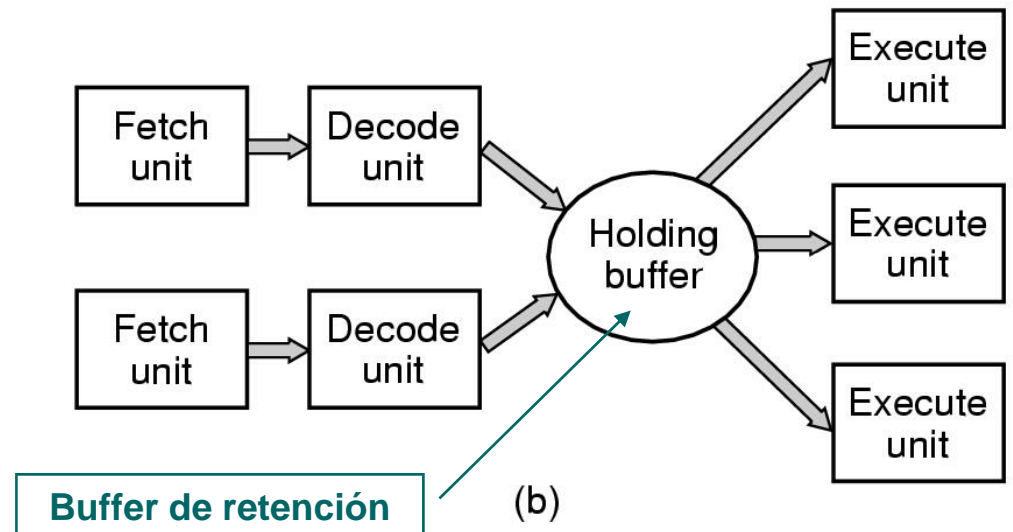
- Cada vez que se detiene el programa en ejecución para iniciar o reiniciar otro:
 - Se deben guardar todos los registros.
- Canalización (Pipeline)
 - Inicia una instrucción antes de terminar la anterior.
 - Ejecuta varias instrucciones simultáneamente.
 - El Intel Pentium ejecuta mas de 6 instrucciones (Webopedia)

Core 2 Duo: 4 instrucciones por ciclo
291 millones de transistores

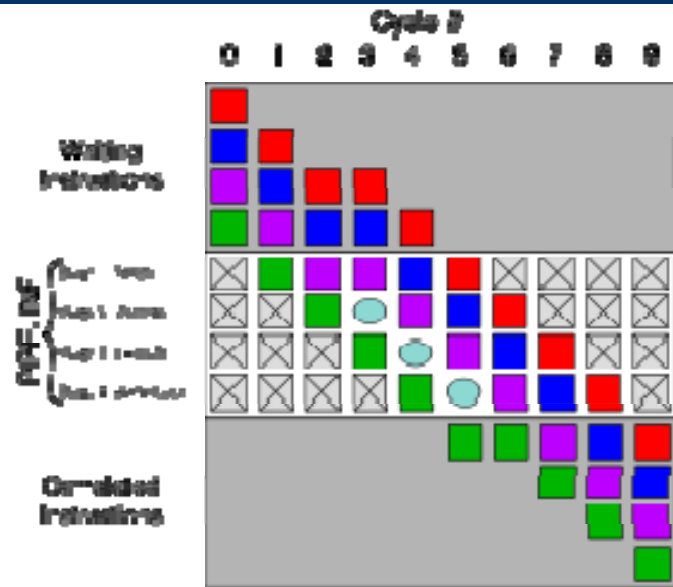
- (a) Pipeline de 3 etapas
- (b) CPU superescalar



(a)



(b)



Pipeline de 4 etapas

Programa de instrucciones en el Intel Pentium 4 (20 etapas).

	TC	NI	TR	F	D	AR	AR	AR	Q	S	S	S	D	D	R	R	E	F	BC	D		
j		TC	NI	TR	F	D	AR	AR	AR	Q	S	S	S	D	D	R	R	E	F	BC	D	
t			TC	NI	TR	F	D	AR	AR	AR	Q	S	S	S	D	D	R	R	E	F	BC	D

– Modos de ejecución:

- Kernel

- El CPU puede ejecutar todas las instrucciones de su conjunto de instrucciones y todas las funciones de hardware.
- El SO se ejecuta en este modo (acceso a todo el hardware)

- Usuario

- Programas de usuario.
- Solo ejecutan un subconjunto de las instrucciones y de las funciones.
- Inhabilitadas las instrucciones de E/S y de protección de memoria

Ambos modos son controlados por PSW

- Un bit indica el modo, solamente el SO lo puede cambiar

- Llamadas al sistema

- Cuando un usuario desea obtener servicios del SO
- A través de una interrupción del sistema* (trap) al kernel invocando al SO, cambia de modo: usuario → kernel.
- Una vez terminado el trabajo regresa el control al programa del usuario (modo) en la instrucción inmediatamente después de la llamada.

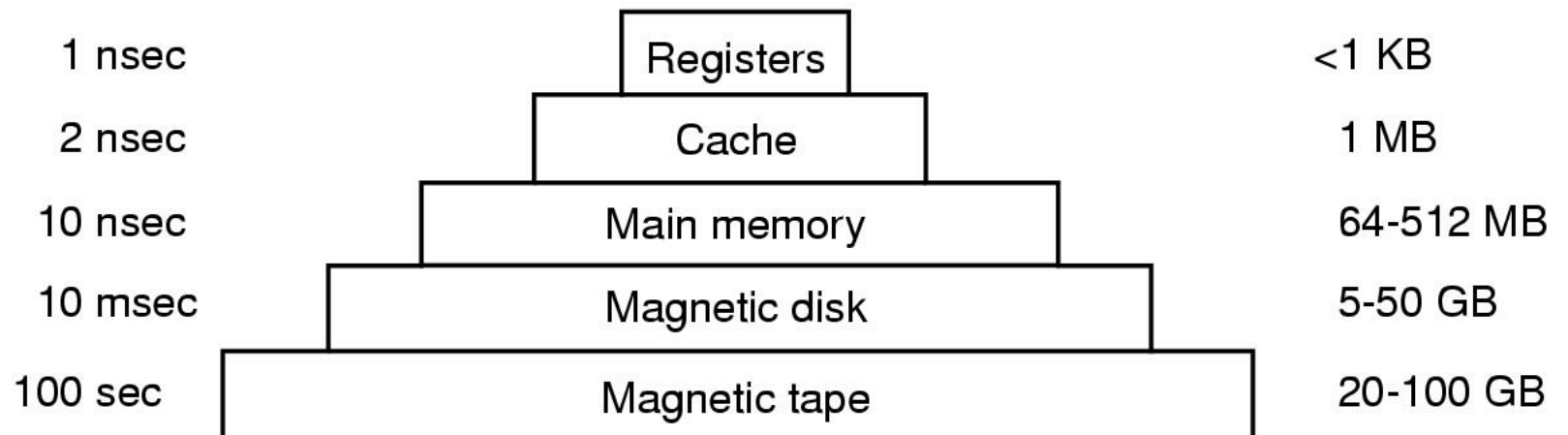
*Otro tipo de instrucciones son generadas por el hardware

- Memoria

- Se construye en una jerarquía de capas:

Typical access time

Typical capacity





– Registros:

- Capacidad 32x32 bits en CPU de 32 bits (64x64 en CPU de 64 bits[512 bytes]), menos de 1 KB en ambos casos.

– Caché

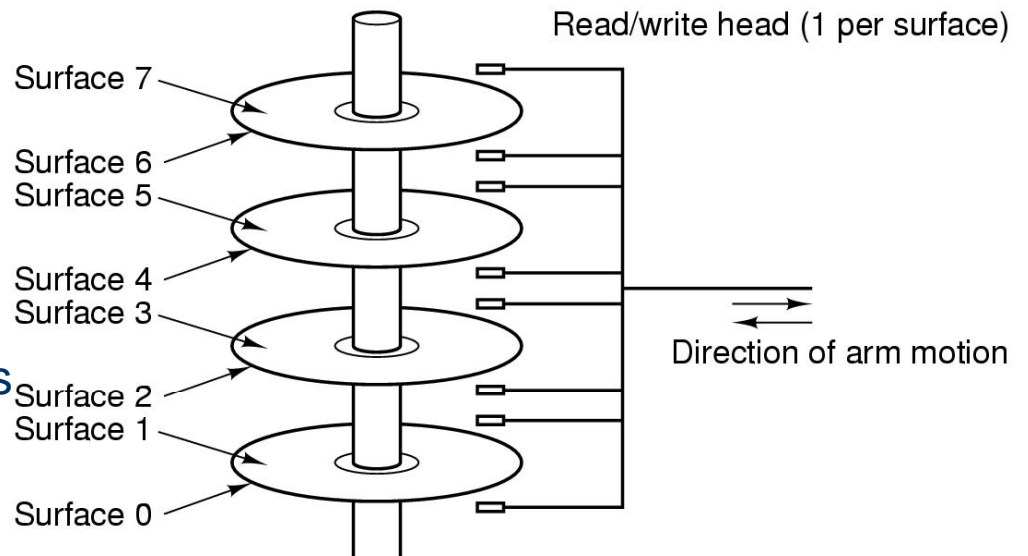
- Se divide en líneas de caché. Por lo regular de 64 Bytes
- Normalmente manejan hasta tres niveles de caché.

– Memoria Principal

- Llamada también memoria RAM.
- Es el “caballo de batalla”

- Disco Magnético

- Más capacidad y más barato
- Más lento en acceso
- Uno o más platos que giran 5400, 7200, 10800 rpm (10K)
- Pistas
- Cilindros
 - Pistas concéntricas.
- Sectores
- Tiempo acceso
- Sistemas de archivos
- Tipos de discos duros



- Cinta magnética
 - Removible
 - Acceso secuencial
 - La más barata
- Otras memorias:
 1. Discos ópticos
 2. SSD (unidades de estado sólido)
 3. Flash (USB o Tarjetas de memoria)
 4. ROM
 1. PROM, EPROM, EEPROM (electronic)
 5. CMOS (complementary metal oxide semiconductor)
 - Chips con bajo consumo de energía. Ejemplo: BIOS.

- Unidad de Administración de Memoria (MMU):
 - Esta en el CPU o cerca de él.
 - Cuando hay dos o más programas en ejecución, se tienen estos problemas:
 1. Cómo proteger los programas entre si y al kernel de todos ellos.
 2. Cómo manejar la relocalización.

- 
- Se apoya de 2 registros:
 - Registro Base
 - Registro Limite

0xFFFFFFFF

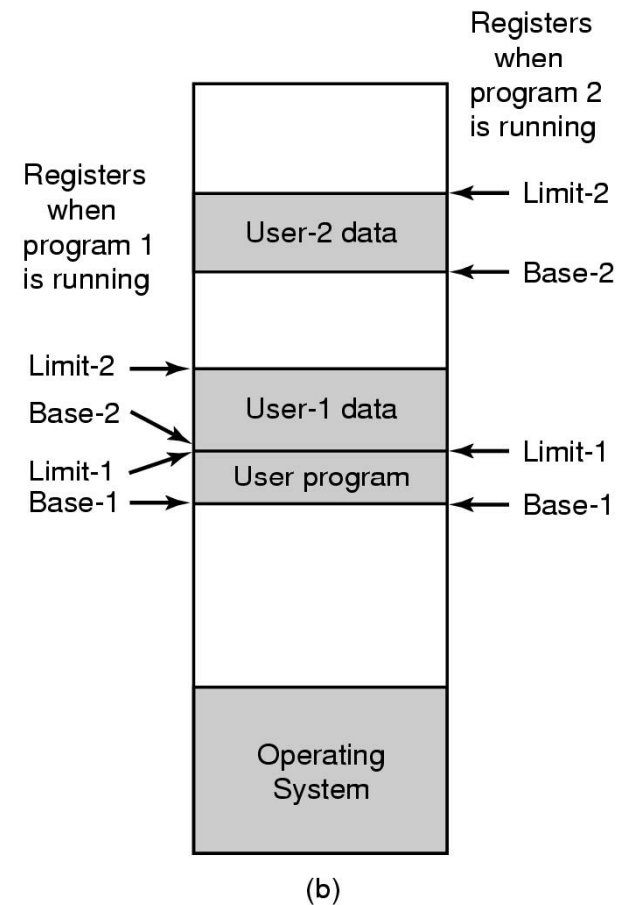
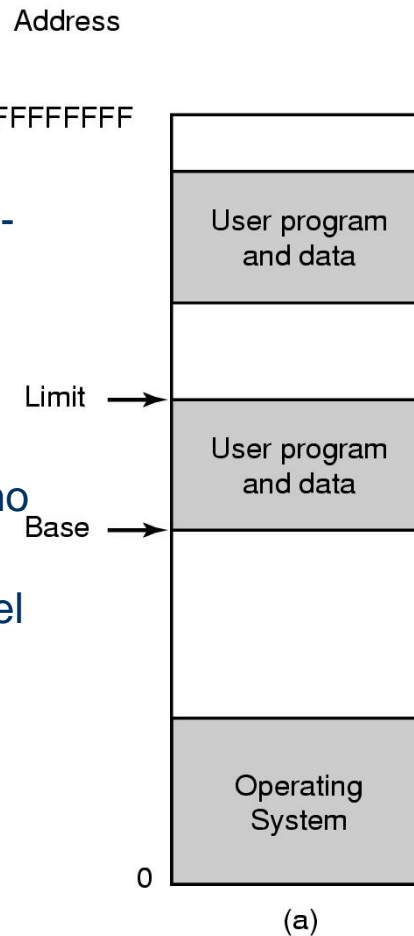
a) Uso de un par Base-Limite

b) Uso de dos pares Base-Limite.

* Varios usuarios comparten el mismo programa.

* Una sola copia del programa en memoria.

* Menos "volcado" de registros.



- Cambiar de un programa a otro resulta costoso (cambios de registros)
- La memoria caché juega un papel importante.

● Dispositivos de E/S

- Un recurso más, administrado por el SO
- Constan de dos partes:
 1. La controladora
 - Chip o conjunto de ellos empotrados en la tarjeta insertable.
 - Recibe comandos del SO y los ejecuta.
 - Normalmente presenta una interfaz sencilla al SO, aunque internamente su procesamiento sea complejo.
 2. El dispositivo en si.
 - Dispositivos con distintas interfaces (IDE, SATA, PCI, AGP, LPT, COM, USB, Firewire, AGP, etc).

- Lo único que ve el SO es la interfaz de la controladora, no interactúa con la interfaz del dispositivo.

- Formas de colocar el controlador del dispositivo en el kernel:
 1. Compilar y reenlazar el kernel, con el nuevo controlador, reiniciando el SO. Ejemplo: Unix y Linux.
 2. Incluir una entrada en un archivo del SO, para indicar la necesidad del dispositivo, y reiniciar cargando los dispositivos. Ejemplo: Windows.
 3. Instalación y ejecución en “caliente”, instala y configura sobre la marcha del SO. Sin reiniciar. Ejemplo: USB y Firewire (IEEE 1394). Los controladores se cargan en forma dinámica.


– Manejo de los registros de los dispositivos:

1. Correspondencia con el espacio de direcciones del SO

- De modo que se manejan como si fueran palabras de memoria ordinarias.
- Los programas de usuario pueden aislarse del hardware con solo dejar esa región de memoria fuera de su alcance.
- No requiere instrucciones especiales, pero ocupa una porción del espacio de direcciones.

2. Se colocan en un espacio especial de puertos de E/S

- Cada registro tiene una dirección de puerto
- Requiere instrucciones especiales (IN y OUT para manejo de los registros, en modo kernel)
- No ocupa direcciones de memoria

- 
- Formas de ejecutar las operaciones de E/S
 1. Espera activa
 2. Interrupciones
 3. DMA: Acceso Directo a Memoria

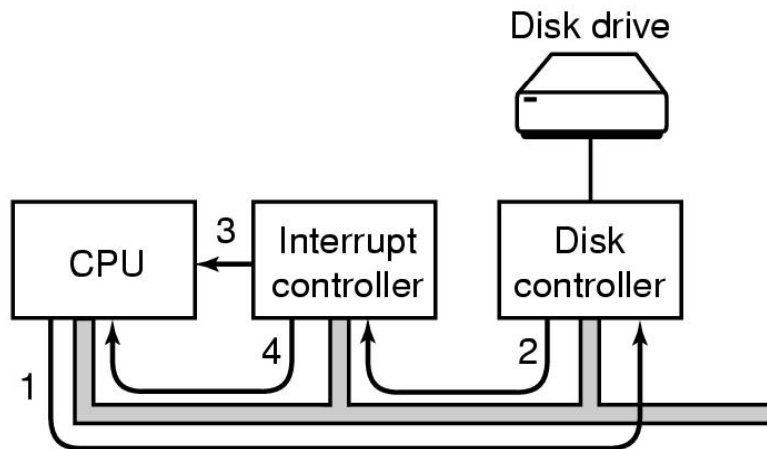
1. Espera activa

- El programa usuario emite una llamada al sistema, el kernel lo traduce a llamada de procedimiento al controlador del dispositivo adecuado.
- El controlador de dispositivo inicia la E/S en un ciclo, preguntando constantemente al dispositivo si ya termino (un bit indica el estatus)
- Una vez terminado, coloca los datos en donde se requieren y regresa.
- El SO devuelve el control al invocador
- Desventaja: Acapara el CPU.

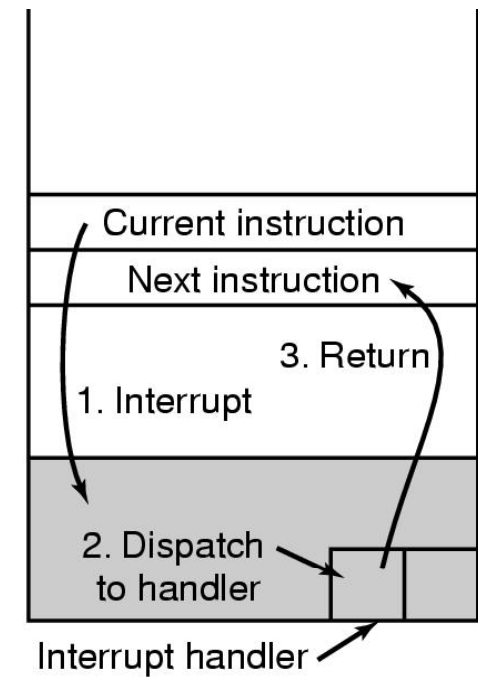
2. Interrupciones

- El controlador pone en marcha al dispositivo y le pide que genere una interrupción cuando haya terminado.
- El CPU se mantiene ocupado mientras el dispositivo se comunica.
- Al terminar el dispositivo la controladora genera la interrupción para avisar.
- El CPU podrá habilitar/inhabilitar la recepción de interrupciones.
- Las peticiones de interrupciones se harán mediante prioridades.

- a) Pasos para iniciar un dispositivo E/S y obtener una interrupción
- b) Cuando el CPU es interrumpido.



(a)



(b)

- 3. DMA: Acceso Directo a Memoria

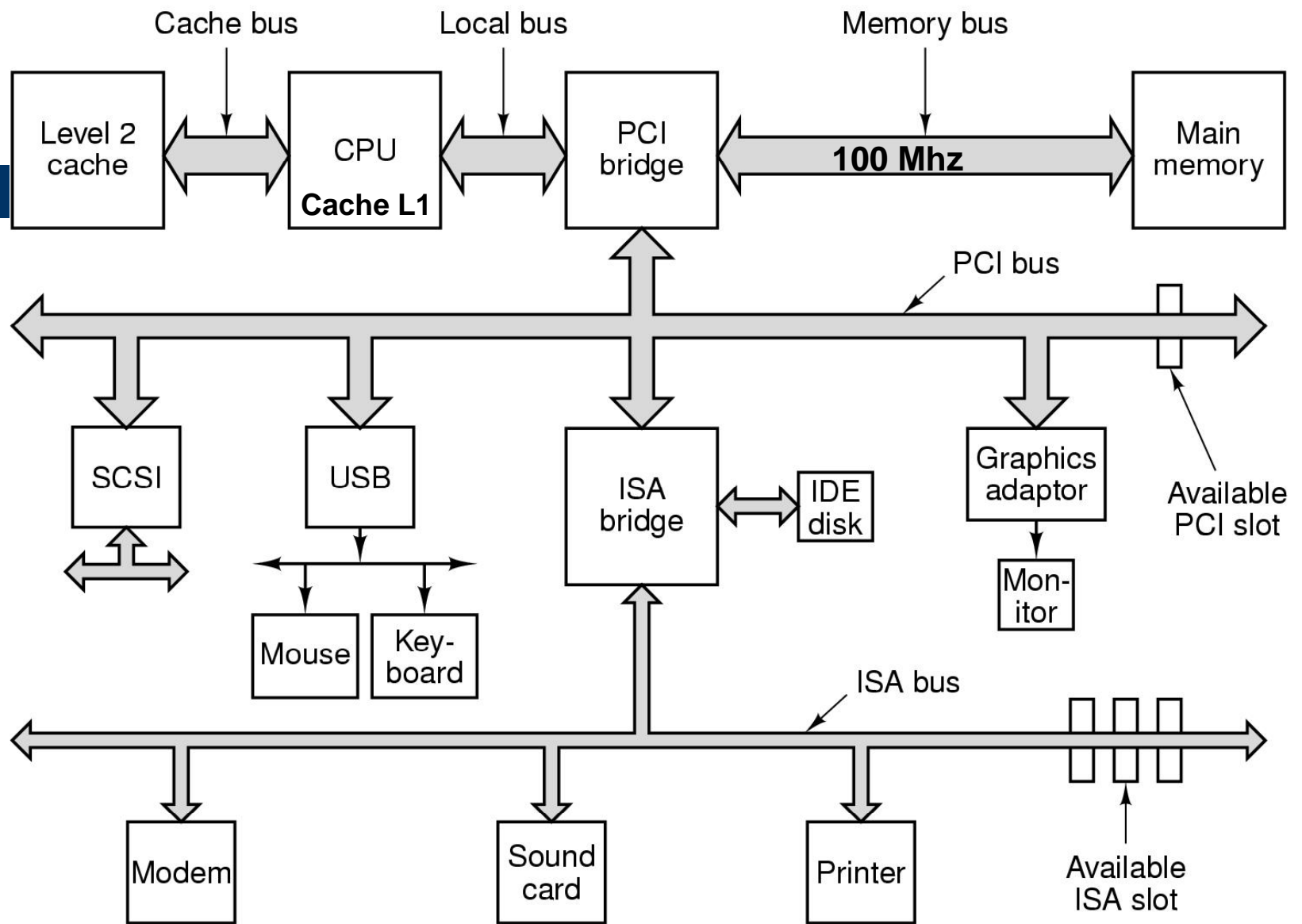
- Controla el flujo de bits entre la memoria y la controladora, con la mínima intervención del CPU.
- El CPU prepara el chip DMA indicándole:
 - Bytes a transferir
 - El dispositivo
 - Las direcciones de memoria en cuestión
- Y se desentiende de él.
- Al terminar, el DMA produce una interrupción indicando “ya acabe”.



- Buses

- Un sistema Pentium tiene 8 buses (caché, local, memoria, PCI, SCSI, USB, IDE e ISA) cada uno con diferente función y tasa de transferencia.
- El SO debe conocerlos, configurarlos y manejarlos.

Estructura de un sistema Intel Pentium



– Principales Buses del Pentium:

- ISA (Arquitectura estándar de la industria)
 - El original de la PC/AT de IBM
 - Opera a 8.33 Mhz
 - Transfiere dos bytes a la vez (Vel máxima de 16.67 MB/s)
 - Se incluyó por compatibilidad
- PCI (Interconexión de componentes periféricos))
 - Sucesor del ISA
 - Inventado por Intel.
 - Opera a 66 Mhz.
 - Transfiere 8 bytes a la vez. (Vel máxima de 528 MB/s)
 - Muy estándar, incluso en otros tipos de computadoras

- IDE (Controlador electrónico integrado)
 - Normalmente para discos duros, CD ROM, DVD ROM, etc.
 - Nombre oficial ATA (Conexión AT), IDE son los dispositivos
 - Transferencias: ATA-1 de 8.3 MB/s, ATA-7 133 MB/s
 - Descontinuado
- S-ATA (Serial ATA)
 - 150 a 600 MB/S
 - SATA I:150 MB/s SATA II: 300 MB/s, SATA III: 600 Mbps

- SCSI (Interfaz estándar de computadoras pequeñas)
 - Bus de alto desempeño, 160 MB/s
 - En MAC, Intel, Unix, etc.
 - Descontinuado
- SAS (Serial Attached SCSI)
 - Serial Attached SCSI
 - Hasta 128 discos
 - Soporta dispositivos
 - Se espera alcancen los 1.5 GBps

- USB (Bus serie universal)
 - Para dispositivos lentos (teclado y ratón).
 - Usa 4 hilos (2 llevan corriente)
 - Sistema centralizado (La raíz cada ciertos ms consulta a los dispositivos para ver si tienen tráfico)
 - Transmite hasta 60 MB/s (Estándar 2.0)
 - Estándares: 1.1. a 12 Mbs y 2.0. a 480 Mbps
 - Todos comparten un solo controlador, por eso funcionan en caliente.
- IEEE 1394 (Firewire)
 - Trabaja en serie
 - Velocidades hasta 100 MB/s
 - Normalmente en equipos multimedia
 - No tiene una controladora central
 - Estándares: 1394a a 400 Mbps y 1394b a 800 Mbps

- AGP (Puertos Acelerador de Gráficos)
 - Versión AGP 8x, de 2 GB/s
 - Descontinuado
- PCI Express
 - Velocidades:
 - X1 = 500 MB/s
 - X2 = 1 GB/s
 - X4 = 2 GB/s
 - X8 = 4 GB/s
 - X12 = 6 GB/s
 - X16 = 8 GB/s
 - X32 = 16 GB/s

– Plug and Play

- Se usa en Apple y en Intel-Microsoft
- Su función, saber que dispositivos existen, y su configuración.
- Recaba información de manera automática, para asignar las interrupciones y direcciones de memoria de E/S en forma centralizada.
- En Pentium, el BIOS contiene software de E/S de bajo nivel de la tarjeta madre:
 - Interrupciones, direcciones de memoria de E/S predeterminados, y procedimientos para interactuar con los dispositivos, buses, memoria, teclado, ratón etc.

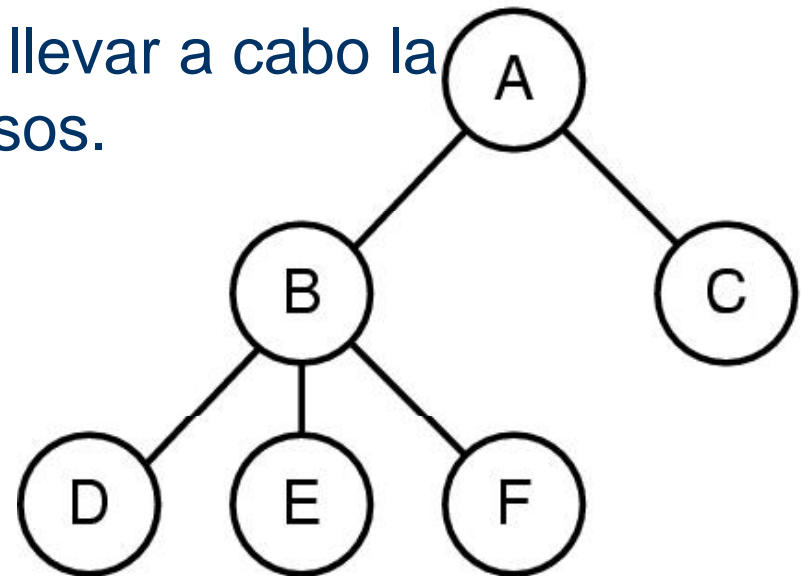
5. CONCEPTOS DE LOS SO

● PROCESOS


- Es un programa en ejecución.
- Cada proceso tiene un espacio de direcciones (rango de memoria)
- El espacio de direcciones contiene:
 - El programa ejecutable
 - Sus datos
 - Su pila
 - Conjunto de registros asociados a el proceso

- Cuando un proceso se suspende:
 - Debe almacenar los apuntadores de los archivos abiertos que estaba usando.
- Tabla de procesos:
 - Arreglo o lista enlazada de estructuras, una para cada uno de los procesos existentes.
 - Aquí se guarda la configuración de los procesos suspendidos, sus registros usados.
- Normalmente las llamadas al sistema son las que se encargan de crear o terminar los procesos.

- Es común que un proceso cree uno o más procesos (procesos hijos) y estos a su vez puedan crear más procesos, originando una estructura de árbol, de 3 o 4 niveles.
- Existen mecanismos para llevar a cabo la comunicación entre procesos.



- Envío de información (que no espera) a un proceso en ejecución:
 - Se envía el mensaje.
 - Pide al SO que se le notifique en n segundos, para ver si llego respuesta (confirmación) de lo contrario retransmitirá.
 - El proceso emisor seguirá trabajando normalmente
 - Agotado el temporizador el SO envía una señal de alarma a proceso emisor.
 - El proceso interrumpe su ejecución para atender la señal (suspende lo que estaba haciendo), ya sea retransmitir o enviar el siguiente mensaje.
 - Las señales son el equivalente en software a las interrupciones en hardware.

- 
- Cada proceso tiene un UID (identificador del usuario) del usuario que lo lanzó.
 - Los procesos hijos tienen el identificador del padre.
 - Hay usuarios que tienen un identificador de grupo (GID)

● BLOQUEOS IRREVERSIBLES

- Es cuando dos o más procesos se colocan en una situación de estancamiento.
- Ejemplos del mundo real:
 - Trafico
 - Juego de cartas (poker)
- Otro ejemplo:
 - Cuando un proceso A solicita el recurso CD y el proceso B solicita el recurso floppy; a ambos se le otorga el recurso. Ahora el proceso A solicita el floppy, pero se queda suspendido en espera de que B lo desocupe. El proceso B solicita el recurso CD el cual A no ha liberado, quedándose suspendido en espera de ello.

● ADMINISTRACIÓN DE MEMORIA

- Varios programas pueden estar en memoria al mismo tiempo.
- Se requieren mecanismos de protección, para protegerse entre sí y proteger al SO de ellos.
- Estos mecanismos están en hardware, pero el SO los controla.
- Memoria virtual:
 - Cuando un proceso requiere más memoria de la disponible.

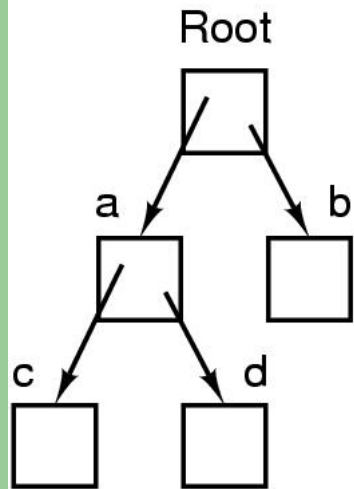
● **ENTRADAS Y SALIDAS**

- Todo sistema cuenta con un subsistema para administración de dispositivos de E/S.

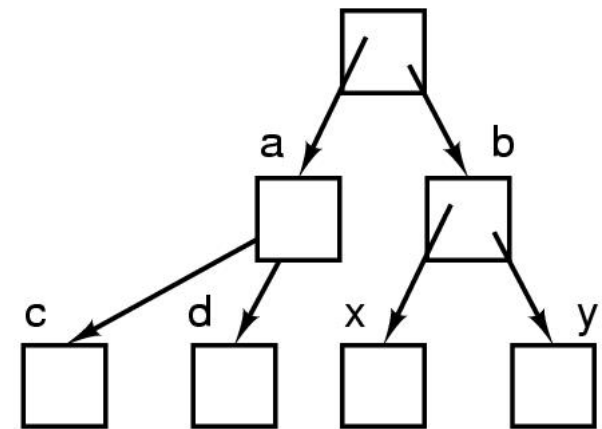
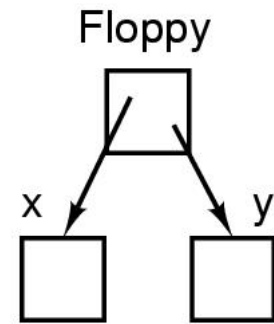
● **ARCHIVOS**

- Directorio, para agrupar archivos.
- Llamadas al sistema para:
 - Eliminar, crear, leer, escribir, copiar, renombrar, cambiar y permisos a archivos y carpetas.
- Jerarquía en forma de árbol, de varios niveles.

- Nombre de ruta
 - \ Windows o / Unix
 - Relativas y absolutas
- Directorio raíz
- Directorio de trabajo
- Descriptor de archivos
 - Código que regresa el SO después de haber revisado los permisos al tratar de leer o escribir un archivo.
- Montado de unidades
 - Normalmente se montan en carpetas vacías (para evitar ocultar la información en ellas)



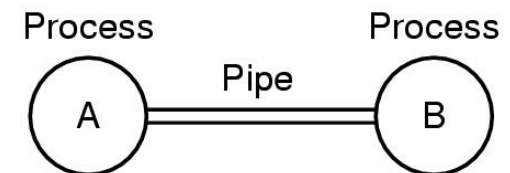
(a)



(b)

- a) Antes de montar el floppy, es inaccesible
- b) Después de montar el floppy

- Archivos especiales:
 - Son para que los dispositivos de E/S se manejen como archivos
 1. Archivos especiales de bloques:
 - Donde se lee o escribe por bloques, (discos duros)
 2. Archivos especiales de caracteres:
 - Dispositivos que manejan flujo de caracteres (módems, impresoras, etc)
- Canalización
 - Es un pseudo archivo que conecta dos procesos para poder comunicarse.
 - El proceso B puede leer los datos de A de la canalización como si fuera un archivo de entrada.



● SEGURIDAD

- En Unix los archivos tienen un código de 9 bits, dividido en tres bloques de tres, para su protección.
 - Propietario rwx
 - Grupo rwx
 - Usuarios rwx
- En un directorio x significa permisos de búsqueda
- Un – significa que el permiso no se a otorgado.
- drwxrwxrwx
 - Primero directorio o no, propietario, grupo, público
 - Con chmod

● EL SHELL

- Es el interprete de comandos.
- Es un ejemplo típico de cómo se hacen las llamadas al sistema.
- No es parte del SO, pero hace uso intensivo de sus funciones.
- Hay muchos shell en Unix: sh, csh, ksh y **bash**. Este último es el más común.

– Al ejecutarlo:

- Se crea un indicador de comandos (prompt)

- Al ejecutar el comando DATE:

- El proceso shell crea un proceso hijo y ejecuta el comando DATE como hijo.
- Al terminar “mata” al proceso hijo y retorna la ejecución al proceso padre.

– Ejemplo:

- `cat arch1 arch2 arch3 | sort > /dev/lp`

- Se usa la salida de un programa como entrada a otro, a través de la canalización.
- Concatena a los tres archivos, ordena el concatenado y lo envía ya ordenado a la impresora.

6. LLAMADAS AL SISTEMA

- Las llamadas al SO es la interfaz entre el SO y los programas de usuario.
- Ejecutar una llamada al sistema es como ejecutar cualquier otro procedimiento, excepto que las llamadas al sistema entran al kernel.

- Ejemplo:

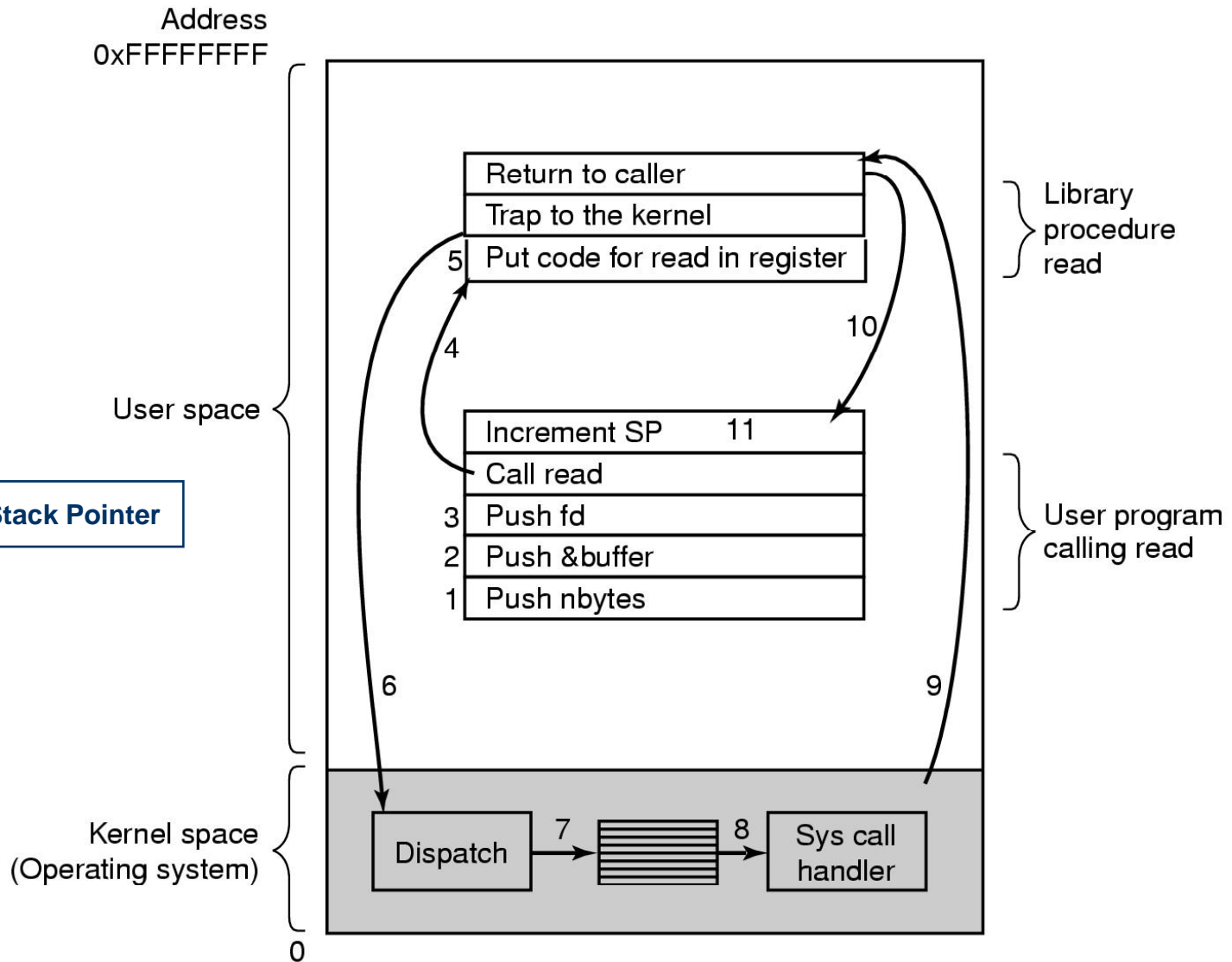
- La llamada a READ tiene tres parámetros:

- El archivo,
- el apuntador al área de memoria (buffer) donde colocará los datos y
- La cantidad de bytes a leer.

Cuenta = read(fd, buffer, nbytes);

- Cuenta: Devolverá la cantidad de bytes leídos o -1
- Errno se activará con el número de error.

LLAMADAS AL SISTEMA



Cuenta=read(fd, buffer, nbytes)

- Llamadas al sistema:
 - a) Para administración de procesos
 - b) Para administración de archivos
 - c) Para administración de directorios
 - d) Llamadas diversas

a) Llamadas para administración de procesos:

LLAMADA	DESCRIPCIÓN
pid=fork()	Crea un proceso hijo idéntico al padre
pid=waitpid(pid, &statloc, options)	Espera a que un hijo termine
s=execve(name, argv, environp)	Sustituye la imagen del núcleo de un proceso. Para ejecutar comando de usuario
exit()	Termina la ejecución de un proceso y devuelve el estado.

s=-1 hubo un error, pid=Identificador de proceso, fd=Descriptor de archivo, n=número de bytes, position=desplazamiento de archivo, second=tiempo transcurrido

b) Llamadas para administración de archivos:

LLAMADA	DESCRIPCIÓN
<code>fd=open(file, how, ...)</code>	Abre un archivo para leer, escribir o ambas cosas
<code>s=close(fd)</code>	Cierra un archivo abierto
<code>n=read(fd, buffer, nbytes)</code>	Lee datos de un archivo a un buffer
<code>n=write(fd, buffer, nbytes)</code>	Escribe datos de un buffer a un archivo
<code>position=lseek(fd, offset, whence)</code>	Mueve el apuntador de archivo
<code>s=stat(name, &buf)</code>	Obtiene información de estado de un archivo

c) Llamadas para administración de directorios y sistema de archivos:

LLAMADA	DESCRIPCIÓN
fd=mkdir(name, mode)	Crea un directorio nuevo
s=rmdir(name)	Elimina un directorio vacío
s=link(name1, name2)	Crea un enlace a un archivo existente, varios nombres en lugares distintos apuntando a un solo archivo
s=unlink(name)	Desenlaza un archivo
s=mount(special, name, flag)	Monta un sistema de archivos
s=umount(especial)	Desmonta un sistema de archivos

d) Llamadas diversas:

LLAMADA	DESCRIPCIÓN
<code>s=chdir(dirname)</code>	Cambia el directorio de trabajo
<code>s=chmod(name, mode)</code>	Cambia los bits de protección de un archivo
<code>s=kill(pid, signal)</code>	Envía una señal a un proceso
<code>seconds=time(&seconds)</code>	Obtiene el tiempo transcurrido desde el 1 de enero de 1970.

LA API WIN32 DE WINDOWS

- API
 - Interfaz de Programación de Aplicaciones
- UNIX
 - Llamadas al sistema
- WINDOWS
 - Es controlado por eventos
 - También tiene llamadas al sistema
 - Se usa la API Win32 para solicitar servicios al sistema.



- API Win32

- Conjunto de procedimientos
- Conserva la compatibilidad desde Windows 95.
- Es muy grande (incremental), varios miles de procedimientos.
- Nunca se sabe cuando es una llamada al sistema o una llamada de biblioteca en espacio de usuario.

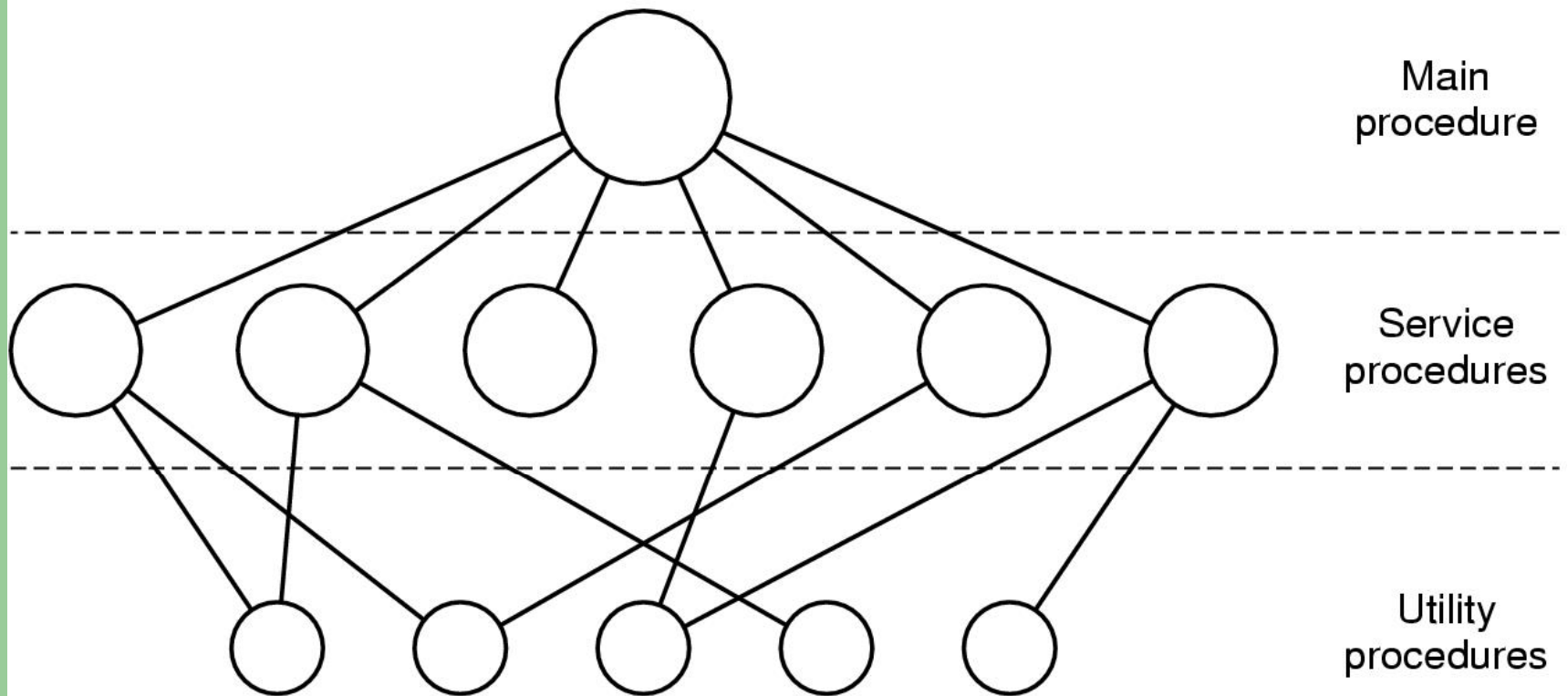
UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

7. ESTRUCTURA DEL SO

1. Sistemas monolíticos
2. Sistemas en capas
3. Máquinas virtuales
4. Exokernels
5. Modelo Cliente-servidor

7.1. Sistemas monolíticos

- Es una colección de procedimientos
- Un procedimiento puede invocar a cualquier otro
- No hay ocultamiento de información
- Estructura básica
 - **Programa principal** que invoca los procedimientos
 - **Conjunto de procedimientos de servicios**, que ejecutan las llamadas al sistema
 - **Conjunto de procedimientos utilitarios** de apoyo a los procedimientos
- Las llamadas a sistema, es a través de los procedimientos de servicios.



Main
procedure

Service
procedures

Utility
procedures

7.2. Sistemas en capas

- Esta cimentado en capas, similar al anterior.
- THE:
 - Primer sistema en capas.
 - Creado en Technische Hogeschool Eindhoven, por E.W. Dijkstra y sus estudiantes.
 - Era un sencillo sistema por lotes para la computadora Electrologica X8
 - Tenia 6 capas

- Capas del THE:

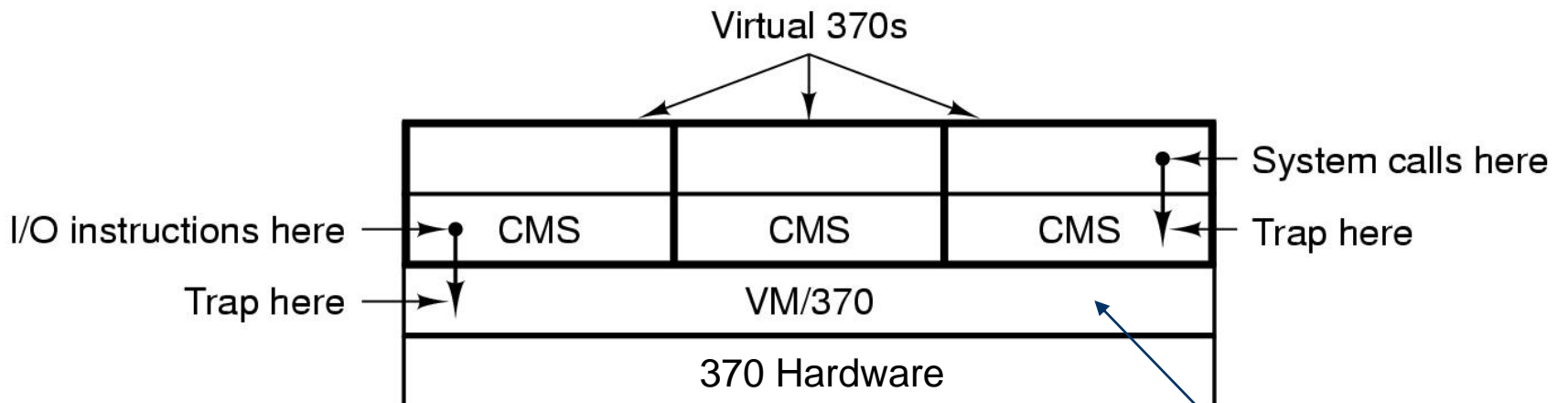
Capa	Función
5	El operador (procesos del operador del sistema)
4	Programas de usuario
3	Administración de E/S
2	Comunicación operador-proceso (consola del operador)
1	Administración de memoria y tambor de palabras (tipo memoria virtual)
0	Asignación de procesador y multiprogramación (multiprogramación)

- MULTICS es similar, pero las capas están en forma de anillo.

7.3. Máquinas virtuales

- Un ejemplo es el VM/370 de IBM (antes CP/CMS).
 - Se usa en las mainframes.
 - Separa por completo la *multiprogramación* de la *máquina extendida* (interfaz cómoda para no interactuar directamente con el hardware desnudo).
 - Monitor de máquina virtual (VM)
 - Es el “corazón” del sistema
 - Se ejecuta en hardware desnudo.
 - Realiza la multiprogramación
 - Proporciona varias máquinas virtuales a la capa inmediata superior

- Cada VM es una copia exacta del hardware desnudo, por lo que puede ejecutar en cada VM, incluso una variante distinta del SO.



CMS = Sistema Monitor de Conversaciones

Monitor de máquina virtual

- La ejecución de los SO se llevan a cabo como si fueran máquinas reales.
- Otro ejemplo: El MSDOS en Windows 2K/XP
 - Es un tipo de máquina virtual
 - Actúa como una 8086 a 16 bits.
 - Mientras la ejecución no afecte áreas reservadas se ejecuta en hardware desnudo.
 - Si afecta a áreas reservadas se hace un TRAP a la VM para la llamada al sistema.

- **MSDOS y VM/370**

- **MSDOS**: Es virtual, cada usuario tiene una copia exacta de una computadora distinta (8086/8088).
- **VM/370**: Cada usuario obtiene una copia exacta de la computadora real.

- **Un tercer ejemplo: JVM**

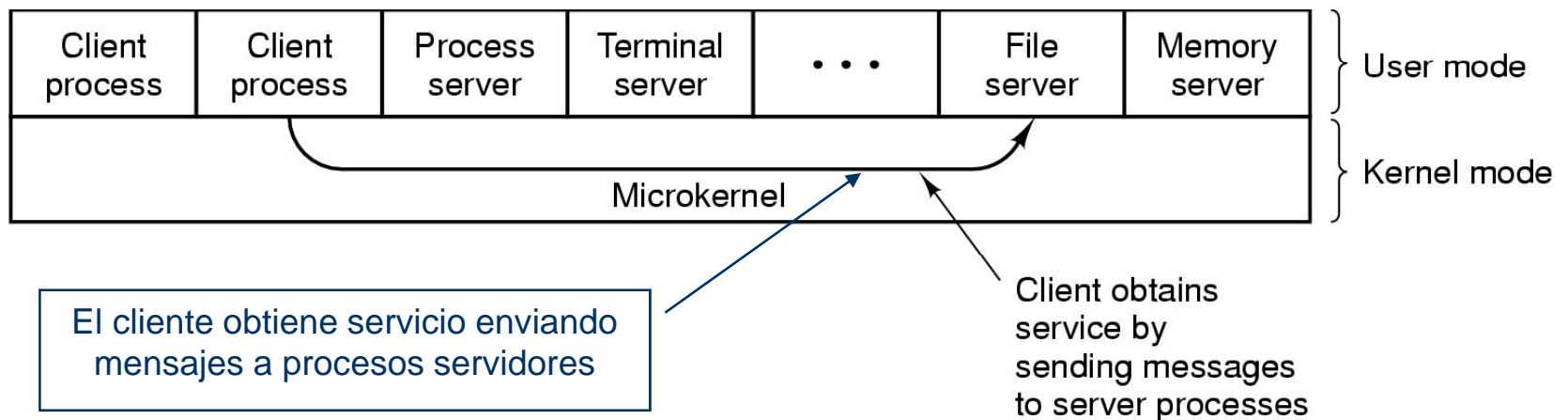
- El compilador Java produce código para JVM que es ejecutado por un interprete en software.
- El código puede compartirse en distintas plataformas.

7.4. Exokernels

- Similar a la estructura de VM.
- Se proporciona una copia real del hardware, pero con un subconjunto de los recursos.
- En la capa más baja se ejecuta un kernel llamado exokernel.
 - Su función es asignar y administrar los recursos.
 - Evita que las VM se interfieran mutuamente con sus recursos.
- Puede ejecutar S.O. distintos.

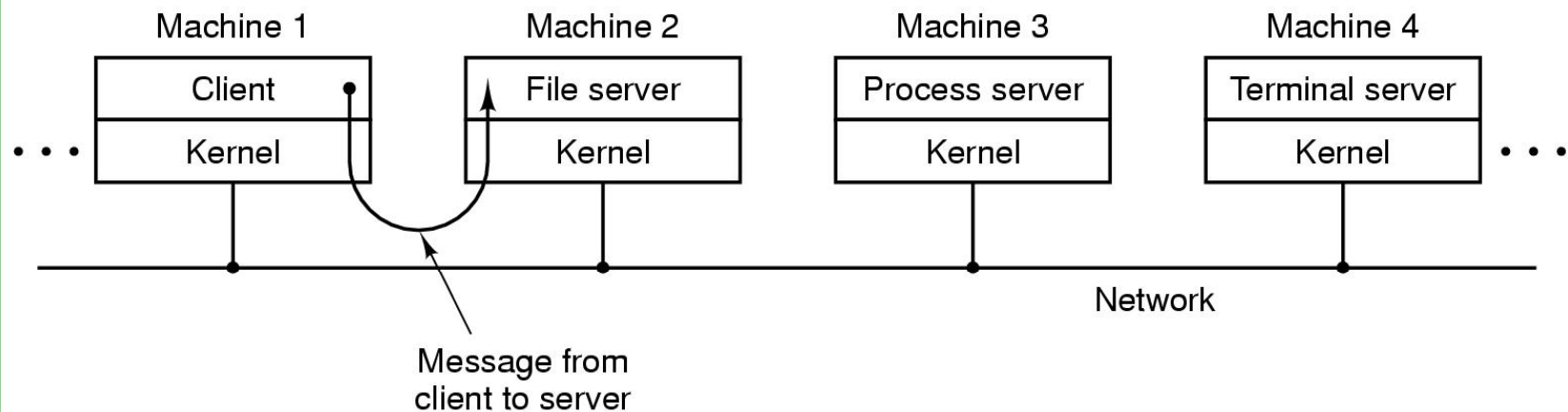
7.5. Modelo cliente-servidor

- La tendencia es llevar la mayoría del código a las capas superiores, y dejar un kernel lo más pequeño posible.
- Mantener un microkernel



- La función principal del Microkernel:
 - Manejar la comunicación entre procesos clientes y servidores.
- Los procesos servidores realmente se ejecutan en modo usuario (no se tiene acceso directo al hardware)
 - Si hay un error en un proceso servidor no se “cae” toda la máquina.
 - Hay procesos críticos que se ejecutan en modo kernel, como carga de comandos en registros de dispositivos de E/S.

- Otra ventaja es que se puede usar en un sistema distribuido.
 - No requiere saber si el mensaje es local o remoto.



REFERENCIA:

- Sistemas Operativos Modernos, Segunda Edición
TANENBAUM
Prentice Hall