

SISTEMAS OPERATIVOS

Unidad 1

Definiciones Básicas

Definición

- El sistema operativo es el software que coordina y dirige todos los servicios y aplicaciones que utiliza el usuario, por eso es el más importante y fundamental en una computadora. Se trata de programas que permiten y regulan los aspectos más básicos del sistema. Los sistemas operativos más utilizados son Windows, Linux, OS/2 y DOS



Definición

- Los sistemas operativos consisten en interfaces gráficas, entornos de escritorio o gestores de ventanas, que brindan al usuario una representación gráfica de los procesos en marcha. También puede ser una línea de comandos, es decir, un conjunto de instrucciones ordenado en base a su prioridad y que funciona en base a comandos y órdenes introducidos por el usuario.
- Permiten que otros programas puedan utilizarlo de apoyo para poder funcionar. Es por ello que a partir del sistema utilizado podrán ser instalados ciertos programas y otros no

Componentes

- **Sistema de archivos:** Es el registro de archivos, donde estos adquieren una estructura arbórea.
- **Interpretación de comandos:** Aquellos componentes que permiten la interpretación de los comandos. Estos tienen como función comunicar las órdenes dadas por el usuario en un lenguaje que el hardware pueda interpretar, sin que aquel que de las órdenes conozca dicho lenguaje.
- **Núcleo:** El mismo permite el funcionamiento en cuestiones básicas como la comunicación, entrada y salida de datos, gestión de procesos y de la memoria entre otros.

Funciones

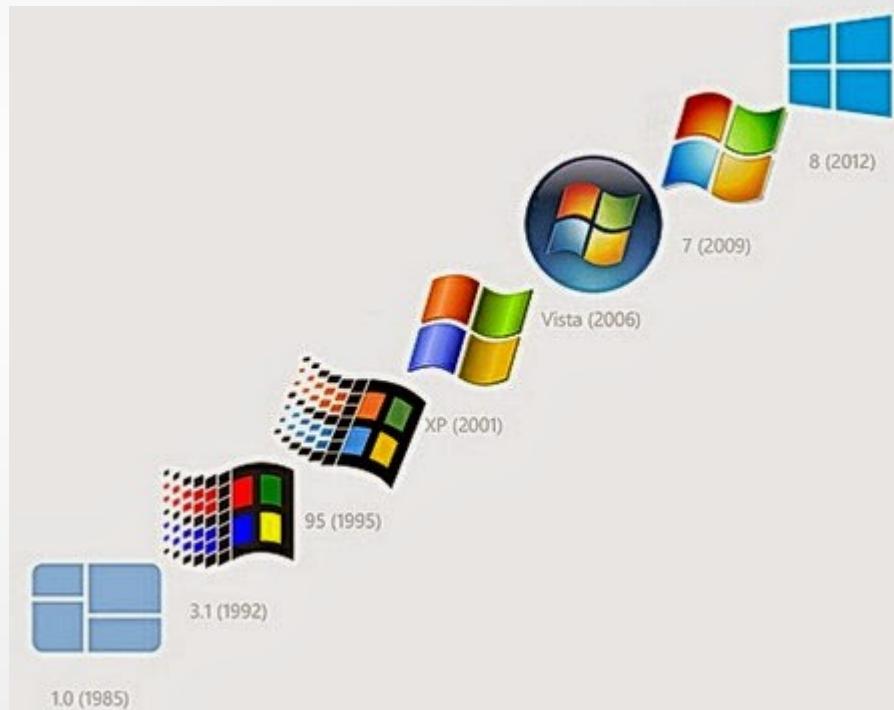
- Gestionar la memoria de acceso aleatorio y ejecutar las aplicaciones, designando los recursos necesarios.
- Administrar al CPU gracias a un algoritmo de programación.
- Direcciona las entradas y salidas de datos (a través de drivers) por medio de los periféricos de entrada o salida.
- Administra la información para el buen funcionamiento de la PC.
- Se encarga de dirigir las autorizaciones de uso para los usuarios.
- Administra los archivos.

Sistemas Operativos

- **Microsoft Windows:** De los más populares que existen, inicialmente se trató de un conjunto de distribuciones o entornos operativos gráficos, cuyo rol era brindar a otros sistemas operativos más antiguos como el MS-DOS, de una representación visual de soporte y de otras herramientas de software. Se publicó por primera vez en 1985 y desde entonces se ha actualizado a nuevas versiones.

Evolución - Microsoft

- Versiones de Microsoft

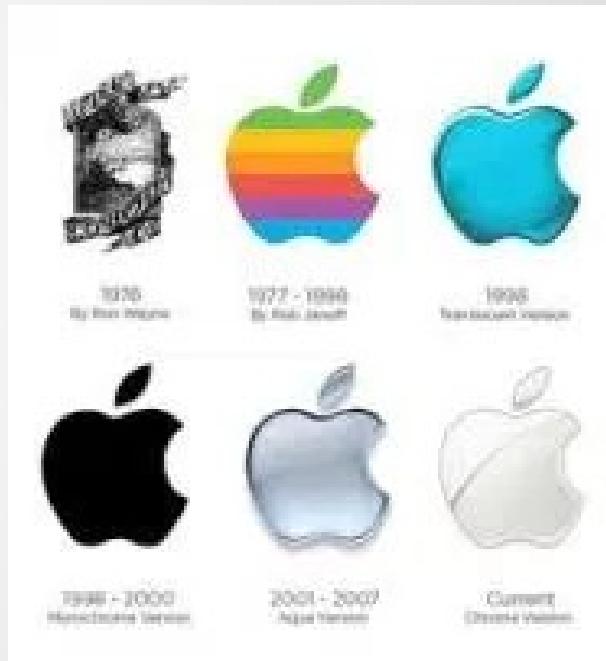


Sistemas Operativos

- **MacOS:** Se llama así al sistema operativo de los computadores Macintosh de Apple, y se le conoce también como OSX o Mac OSX. Basado en Unix y desarrollado y vendido en computadores Apple desde 2002, se trata de la competencia más acérrima del popular Windows.

Evolución - Apple

- Versiones de Apple



Sistemas Operativos

- **UNIX:** Nació a principios de la década de los 70 por los desarrolladores Ken Thompson y Dennis Ritchie. Fue creado en los Laboratorios Bell, que pertenecen a la famosa compañía AT&T. Fue creado como un sistema operativo para manejar servidores, siendo un sistema operativo donde los comandos tienen casi todo el protagonismo.
- **LINUX:** El Kernel Linux fue creado por Linus Torvalds a principios de los 90. El Kernel se creó basándose en Unix y está bajo licencia GNU, por lo tanto es completamente libre y gratuito y cualquier persona puede modificar el código fuente, el cual está disponible para todo el mundo.

Sistemas Operativos - Linux

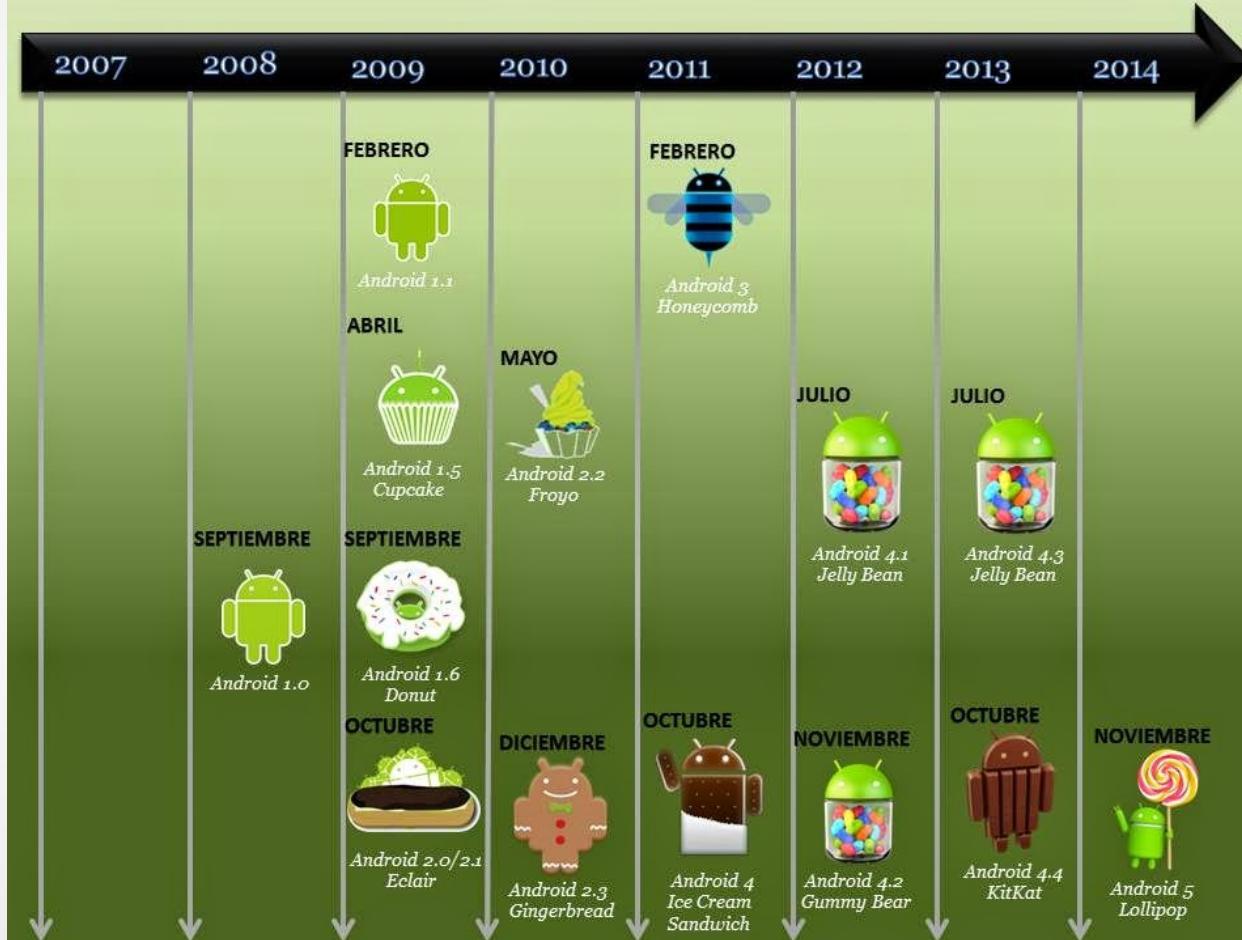
- Linux, sistema operativo de código abierto, esto quiere decir que cada uno puede aportar al desarrollo del mismo, por esa razón hay muchas versiones/distribuciones



Sistemas Operativos

- **Android:** Este sistema operativo basado en el núcleo Linux, opera en teléfonos celulares y tablets y otros artefactos dotados de pantalla táctil. Fue desarrollado por Android Inc. y comprado posteriormente por Google, gracias a lo cual es tan popular que las ventas de sistemas informáticos Android superan a las de IOS (para teléfonos celulares Macintosh) y a las de Windows Phone (para teléfonos celulares MicroSoft).

Evolución - Android



SISTEMAS OPERATIVOS

Unidad 1

Tipos de Sistemas Operativos

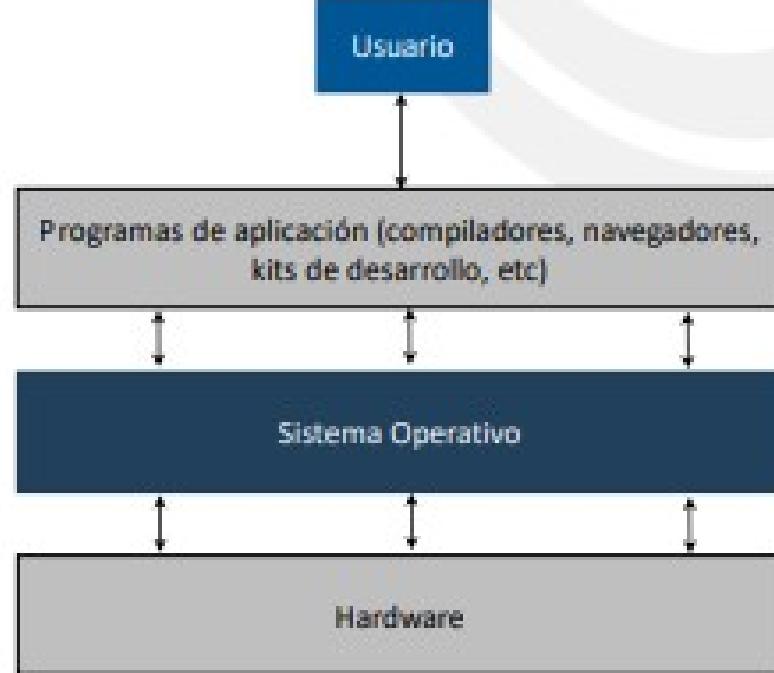
Definición

1.- Hardware

2.- Sistema Operativo

3.- Programas de Aplicación

4.- Usuarios



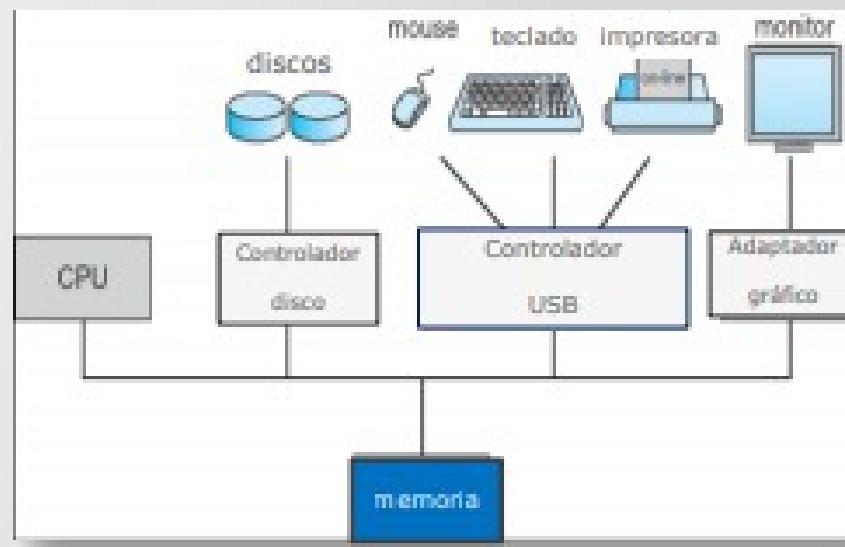
Funciones Principales

- Alocador de recursos
 - administra todos los recursos.
 - decide sobre requerimientos conflictivos para asegurar eficiencia y uso imparcial de recursos
- Programa de Control
 - controla la ejecución de los programas para prevenir errores y el uso impropio de la computadora.

“El programa que ejecuta todo el tiempo la computadora” es el **kernel o núcleo**. Todo lo demás es un programa de sistema o un programa de aplicación.

Funciones

- Operación del Sistema de Cómputo
 - Una o varias CPUs.
 - Ejecución concurrente de CPUs y dispositivos compiten por ciclos de memoria.
 - La CPU mueve datos desde/hacia la memoria principal a/desde los buffers locales.
 - El controlador de dispositivo informa a la CPU que ha finalizado su operación por medio de una interrupción.



Inicio de la PC

- El programa de **bootstrap** es cargado en el encendido o reinicio
 - Típicamente almacenado en ROM o EEPROM, generalmente conocido como firmware
 - Inicializa todos los aspectos del sistema
 - Carga el kernel del sistema operativo y comienza la ejecución

Tipos de Sistemas Operativos

- Distribuidos
- De Tiempo Real
- Monousuario
- Multiusuario
- Propósito General

Distribuidos

- Es una colección de procesadores conectados en red donde cada uno tiene su propia memoria
- Si un host de un sistema distribuido falla el resto de los host pueden continuar funcionando.
- La falla de un host debe detectarse por el sistema, el cual tendrá que tomar una acción adecuada para recuperarse de ella, entre las acciones está ver si existe la posibilidad de que otro host se encargue de la tarea de aquel que falló haciendo la oportuna transferencia de funciones

De Tiempo Real

- Un **sistema de tiempo real** es un sistema informático que interacciona con su entorno físico y responde a los estímulos del entorno dentro de un plazo de tiempo determinado. No basta con que las acciones del sistema sean correctas, sino que, además, tienen que ejecutarse dentro de un intervalo de tiempo determinado
 - Tiempo Real Duro: plazos de respuesta deben respetarse siempre estrictamente, una respuesta tardía puede tener consecuencias graves
 - Tiempo Real Suave: se pueden tolerar retrasos ocasionales en la respuesta a un suceso

Ejemplo: Un robot que debe tomar una pieza de una cinta transportadora, si llega antes la pieza no esta y puede bloquear el resto y si llega tarde la pieza ya paso y el proceso se debe interrumpir

Monousuario

- Sólo puede ser ocupado por un único usuario en un determinado tiempo.
- Es un sistema en el cual el tipo de usuario no está definido y, por lo tanto, los datos que tiene el sistema son accesibles para cualquiera que pueda conectarse.
- Puede ser:
 - Monotarea: Este tipo de software se limita a la ejecución de **solo una tarea** por vez.
 - Multitarea: permite realizar **varias tareas a la misma vez**, se pueden ejecutar diferentes programas y aplicaciones al mismo tiempo.

Ejemplos SO Monousuario: MS-DOS, todos los windows hasta el Windows 98 y ME.

Multiusuario

- Permite que varios usuarios ejecuten simultáneamente sus programas, accediendo a la vez a los recursos de la computadora.
- Cualquier sistema multiusuario es multitarea también.

Ejemplos: Windows XP en adelante, MacOS, Android, Linux, etc.

Propósito General

- Los sistemas de propósito general se caracterizan por tener un gran número de usuarios trabajando sobre un amplio abanico de aplicaciones, son los Sistemas Operativos que mas conocemos.
- Su contraparte son los de propósito específico utilizados en control industrial, simulaciones o controles de vuelo.

SISTEMAS OPERATIVOS

Unidad 2

Componentes del Sistema Operativo

Componentes



Componentes

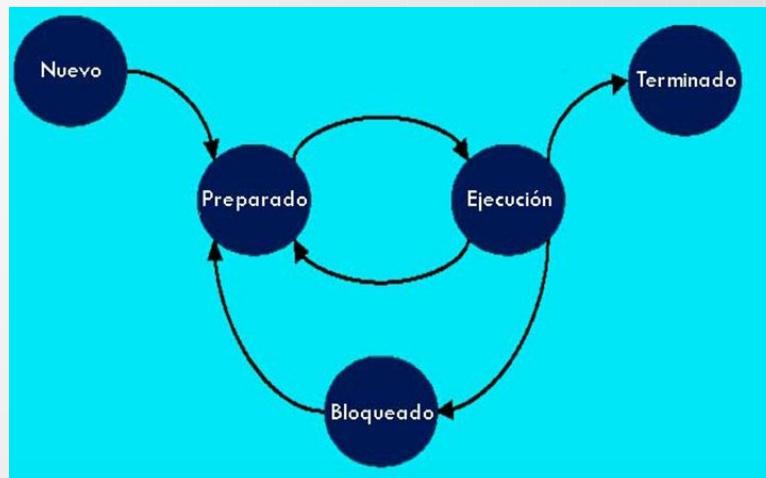
- Gestión de procesos
- Gestión de la memoria principal
- Gestión del almacenamiento secundario
- Sistema de entrada/salida
- Sistema de archivos
- Sistemas de protección
- Sistema de comunicaciones
- Programas del sistema
- Gestión de recursos

Gestión de procesos

- Cuando se habla de un proceso se hace referencia a un **programa en ejecución**. Un proceso es un conjunto de instrucciones que corresponden a un programa y que son ejecutadas por la CPU. En un programa se pueden ejecutar uno o varios procesos diferentes. La ejecución de un programa necesita recursos del sistema como tiempo de CPU, memoria, archivos y dispositivos de E/S.
- Un proceso puede pasar por los estados nuevo, preparado, ejecución, bloqueado y terminado cuando su ciclo de vida es de 5 estados. Si el proceso solo se ejecuta y termina el ciclo de vida es de 2 estados. Los procesos para su ejecución se planifican siguiendo algoritmos. Dos de los algoritmos de planificación más comunes son el algoritmo Round Robin y el algoritmo FIFO.

Gestión de procesos

- El sistema operativo es el responsable de asignar recursos a los procesos, crear y destruir procesos, parar y reanudar procesos y proporcionar que los procesos se comuniquen y sincronicen.



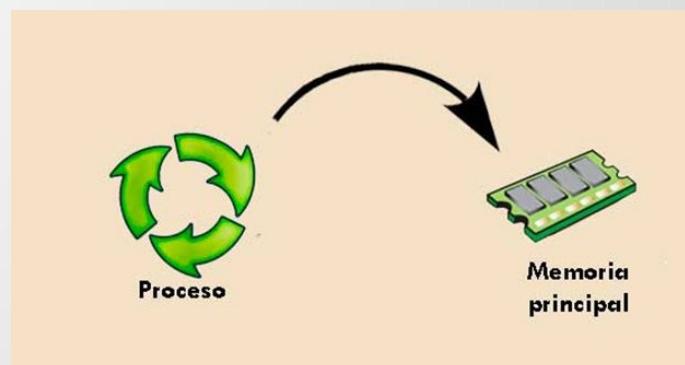
Gestión de procesos

- En Windows esto se puede ver en justamente Administración de tareas

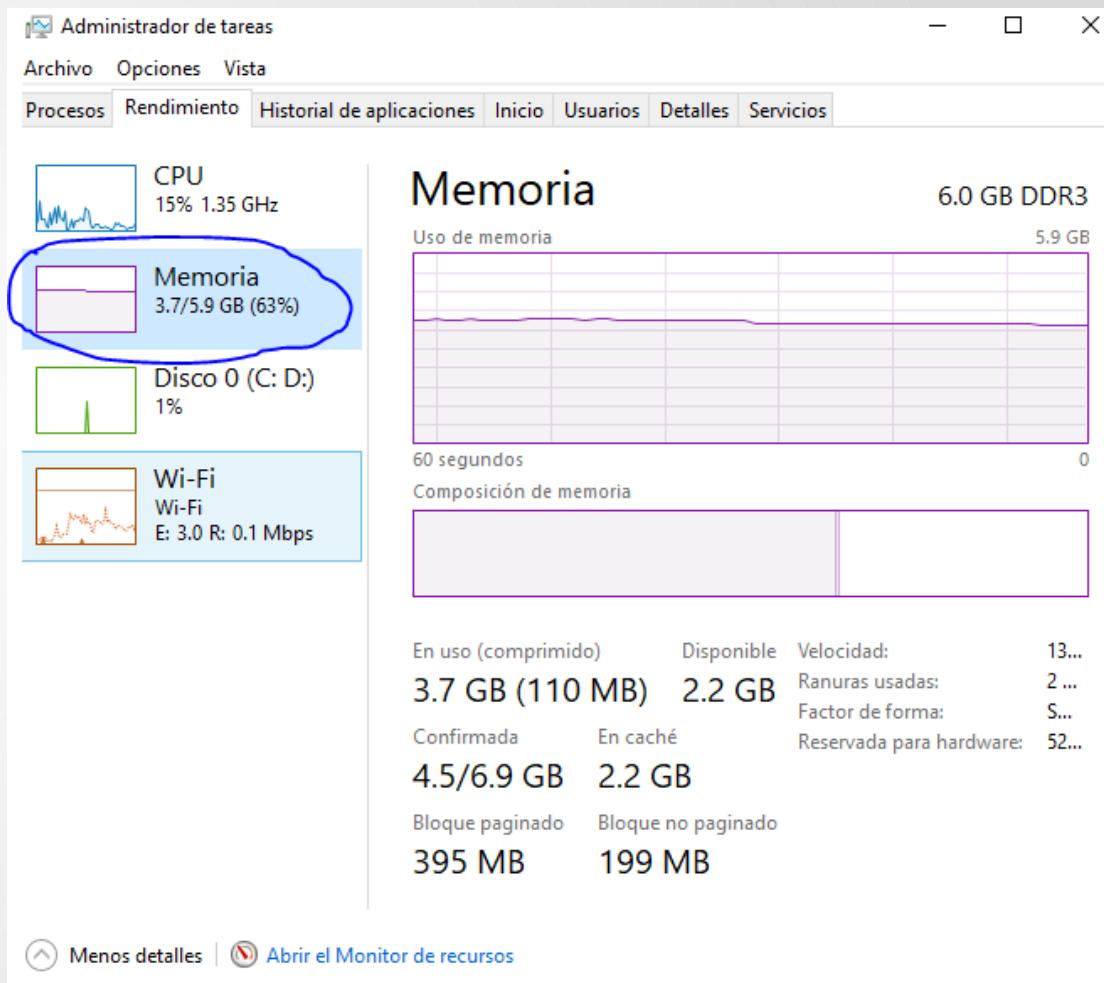
Nombre	Estado	16% CPU	60% Memoria	0% Disco	1% Red	C
Aplicaciones (6)						
> Administrador de tareas		0.9%	21.7 MB	0 MB/s	0 Mbps	
> Explorador de Windows		1.7%	31.6 MB	0 MB/s	0 Mbps	
> Google Chrome (25)		5.8%	720.8 MB	0.1 MB/s	0.1 Mbps	
> Herramienta Recortes		0.6%	3.1 MB	0 MB/s	0 Mbps	
> Microsoft PowerPoint		0.2%	22.9 MB	0 MB/s	0 Mbps	
> Microsoft Word		0%	23.6 MB	0 MB/s	0 Mbps	
Procesos en segundo plano (54)						
> µTorrent Helper (32 bits)		0%	1.8 MB	0 MB/s	0 Mbps	
> 64-bit Synaptics Pointing Enhanc...		0%	0.5 MB	0 MB/s	0 Mbps	
> Antivirus engine server		0%	19.0 MB	0 MB/s	0 Mbps	
> AnyDesk (32 bits)		0%	1.0 MB	0 MB/s	0 Mbps	
> AnyDesk (32 bits)		0%	3.3 MB	0 MB/s	0 Mbps	

Gestión de Memoria

El sistema operativo es el responsable de gestionar la memoria principal conociendo qué espacios de la memoria está siendo utilizada y por qué procesos, decidiendo qué procesos se cargarán en memoria cuando haya espacio disponible, asignando y reclamando espacio de memoria cuando sea necesario, administrar el intercambio entre la memoria principal y la memoria virtual (espacio en el disco usado para los procesos cuando la memoria principal no es suficiente).



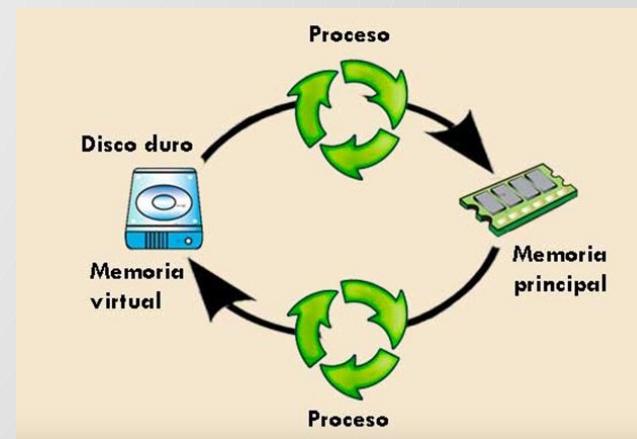
Gestión de Memoria



Gestión del almacenamiento secundario

El sistema operativo se encarga de planificar los discos, gestionar el espacio libre, asignar el almacenamiento y verificar que los datos se guarden en orden.

La memoria principal no es suficiente para almacenar los programas y los datos, además de ser volátil porque los datos se pierden ante un fallo de la energía eléctrica. Por tal motivo es necesario un sistema de almacenamiento secundario también denominado memoria virtual.



Sistema de entrada/salida

- El sistema de entrada/salida representa el intercambio de información entre el procesador y los dispositivos periféricos (teclado, mouse, pantalla, impresora y otros). Los dispositivos periféricos solicitan recursos del sistema por medio de interrupciones.
- El sistema operativo gestiona el almacenamiento temporal de entrada/salida y las interrupciones de los dispositivos de entrada/salida.



Sistema de archivos

- El sistema de archivos forma parte de los componentes de un sistema operativo y son la forma en que se organiza la información. Los sistemas de archivos más comunes son FAT, FAT32, ext3, NTFS, XFS.
- El SO es responsable de construir y eliminar archivos y directorios, manipular archivos y directorios, establecer la correspondencia entre archivos y unidades de almacenamiento, realizar copias de seguridad de archivos.

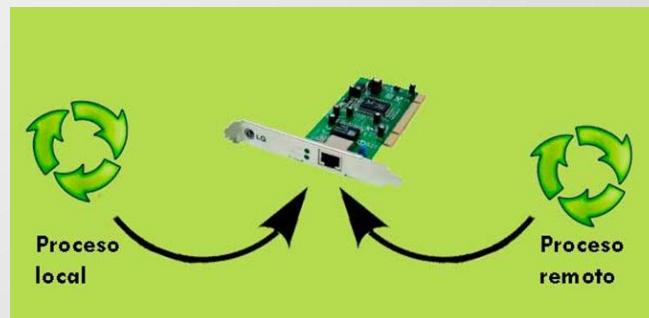
Sistemas de protección

- En un SO varios usuarios pueden ejecutar simultáneamente sus programas, varios procesos se pueden ejecutar simultáneamente, varios programas se pueden ejecutar al mismo tiempo, varios procesos se pueden intercalar para su ejecución simulando una ejecución simultánea.
- El sistema operativo se encarga de distinguir entre uso autorizado y no autorizado, especificar los controles de seguridad a realizar y forzar el uso de los mecanismos de protección **controla el acceso de los programas o los usuarios a los recursos del sistema.**



Sistema de comunicaciones

- Permite el **intercambio de información entre procesos y programas** que se ejecutan localmente con procesos y programas que se ejecutan de forma remota.
- Las tareas de envío y recepción de información las ejecuta el sistema de comunicaciones a través de las interfaces de red.
- El sistema operativo es el responsable de controlar el envío y recepción de la información, crear y mantener la comunicación para que las aplicaciones envíen y reciban información, y crear y mantener conexiones virtuales entre aplicaciones locales y aplicaciones remotas.



Programas del sistema

- Son **aplicaciones que se instalan** con el sistema operativo pero que no forman parte de él. Los programas del sistema son útiles para el desarrollo y ejecución de los programas de usuario.
- Las tareas que realizan los programas del sistema son: manipulación y modificación de archivos, información del estado del sistema, soporte a lenguajes de programación y comunicaciones.
- El SO es el encargado de gestionar las tareas que realizan los diferentes programas del sistema



Gestión de recursos

- Para que una PC pueda realizar las tareas solicitadas por el usuario requiere de la asignación de recursos para cada una de esas tareas. El sistema operativo administra los recursos que se deben asignar a los programas en ejecución.
- El sistema operativo administra la unidad central de procesamiento, los dispositivos de entrada y salida, la memoria principal o memoria RAM, los discos o memoria virtual, los procesos o programas en ejecución y en general todos los recursos del sistema.

SISTEMAS OPERATIVOS

Unidad 2

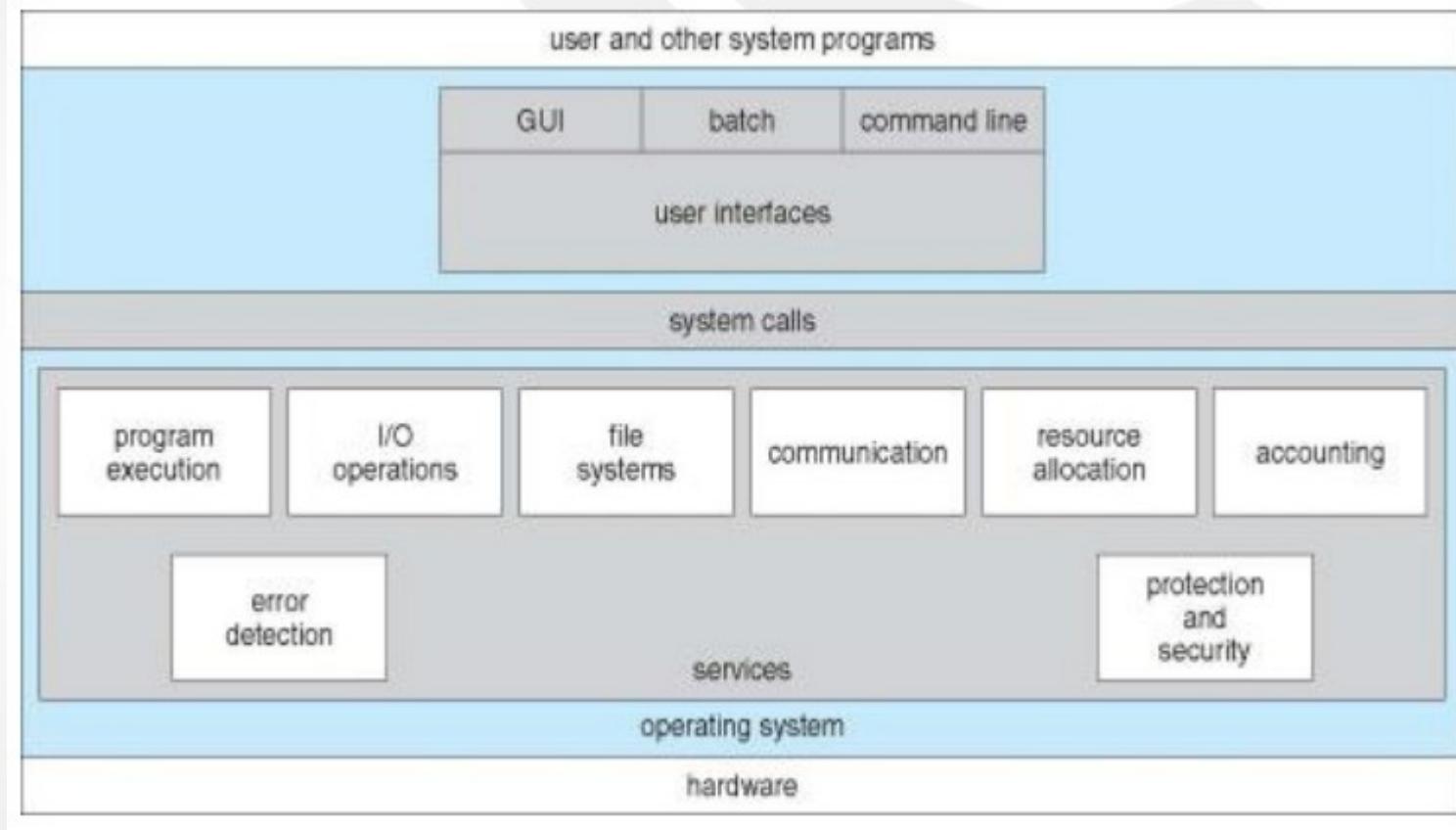
Servicios, Llamadas al Sistema y Estructuras de SO

Servicios

Un conjunto de servicios del SO proveen funciones que son útiles al usuario:

- Interfaz de Usuario
- Ejecución de Programas
- Operaciones de E/S
- Manipulación del Sistema de Archivos
- Comunicaciones
- Detección de errores
- Y otros: alocación de recursos, contabilidad, protección ..

Servicios

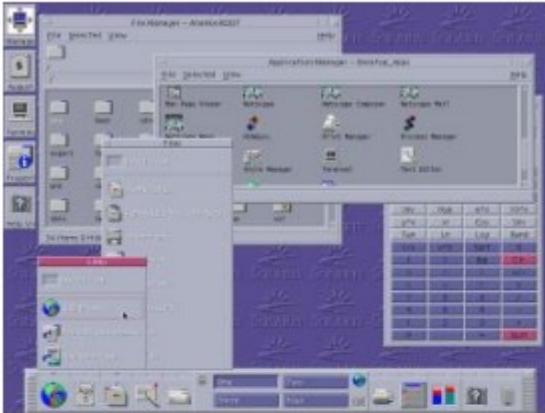


Interfaz de Usuario

- Interfaz de líneas de comando (Command Line Interface - CLI) o intérprete de comando permite entrar comandos en forma directa, pueden ser por línea de comandos o gráficas:
 - Algunas veces implementadas en el kernel, otras como programas de sistema
 - La implementación a veces está embebida, y en otras es invocación a programas.
- Interfaz Gráfica (GUI)
- Interfaz Touch (especialmente en móviles)

Interfaz de Usuario

Solaris – CDE (Common Desktop Environment)



Mac OS GUI



Android



ios



Estructuras de SO

- Sistemas Monolítico
- Sistemas en capas
- Sistemas con micronúcleo o microkernel
- Sistemas Modulados
- Sistemas Híbridos

Sistemas Monolítico

- Estos sistemas no tienen una estructura definida, sino que son escritos como una colección de procedimientos donde cualquier procedimiento puede invocar a otro. Ejemplos de estos sistemas pueden ser MS-DOS o Linux (aunque incluye algo de capas). Es importante tener en cuenta que ningún sistema es puramente de un tipo.

```
C:\>command

Microsoft(R) MS-DOS(R) Version 4.01
(C)Copyright Microsoft Corp 1981-1988

C:\>ver
MS-DOS Version 4.01

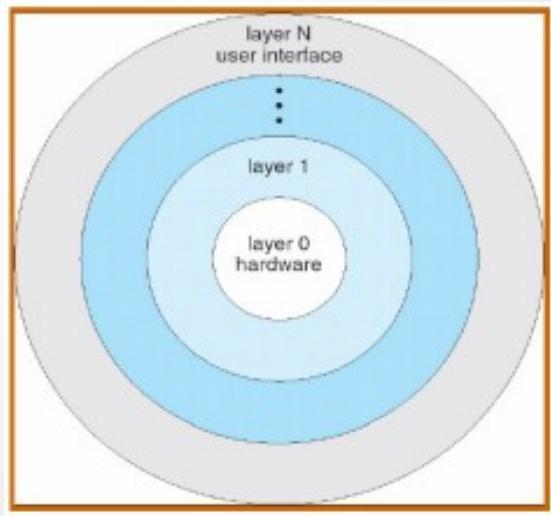
C:\>dir command.com
Volume in drive C is DOS
Volume Serial Number is 2432-07DC
Directory of C:\

COMMAND.COM   37557 12-19-88 12:00a
               1 File(s)  495624192 bytes free

C:\>_
```

Sistema en capas

El diseño se organiza en una jerarquía de capas, donde los servicios que brinda una capa son consumidos solamente por la capa superior. La capa 0 es del Hardware y la N es la de los procesos de Usuario.



Estos sistemas tienen como ventaja que son modulares y la verificación se puede hacer a cada capa por separado (son más mantenibles). Sin embargo el diseño es muy costoso y es menos eficiente que el sistema monolítico ya que pierde tiempo pasando por cada capa.

Sistemas con micronúcleo

La idea consiste en tener un núcleo que brinde los servicios mínimos de manejo de procesos, memoria y que provea la comunicación entre procesos. Todos los restantes servicios se construyen como procesos separados del micronúcleo, que ejecutan en modo usuario.

Estos sistemas tienen como ventaja un diseño simple y funcional, que aumenta la portabilidad y la escalabilidad. Para agregar un nuevo servicio no es necesario modificar el núcleo, y es más seguro ya que los servicios corren en modo usuario.

Sistemas con micronúcleo

Mueve tanto como se pueda al espacio de usuario

Las comunicaciones entre módulos de usuarios se realiza por medio de pasajes de mensajes

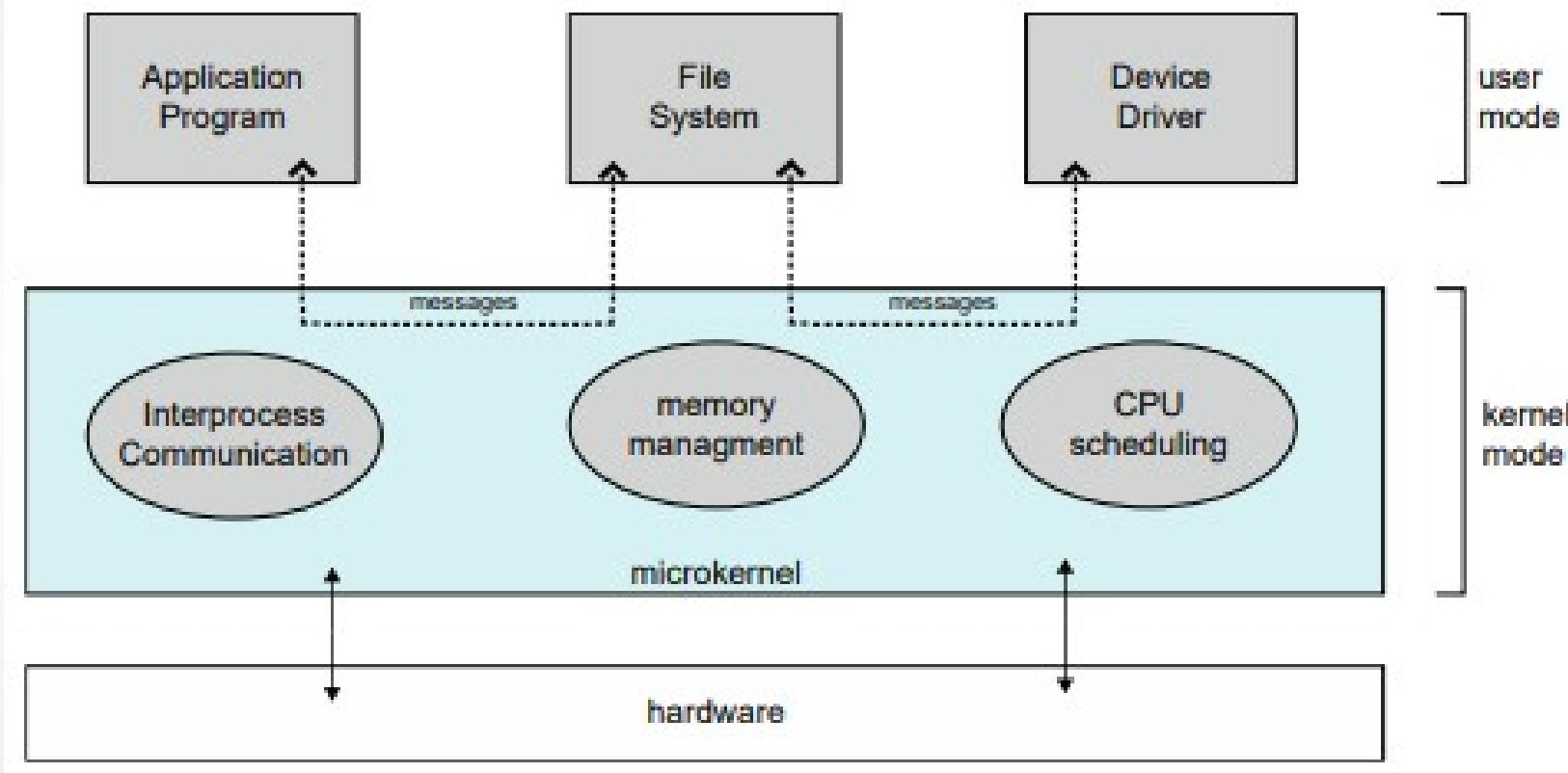
Beneficios:

- Más confiable (menos código corre en el modo kernel)
- Más fácil de portar el SO a nuevas arquitecturas
- Más fácil de extender
- Más seguro

Detrimientos:

- Sobrecarga de rendimiento en la comunicación del espacio de usuario al espacio de kernel

Sistemas con micronúcleo



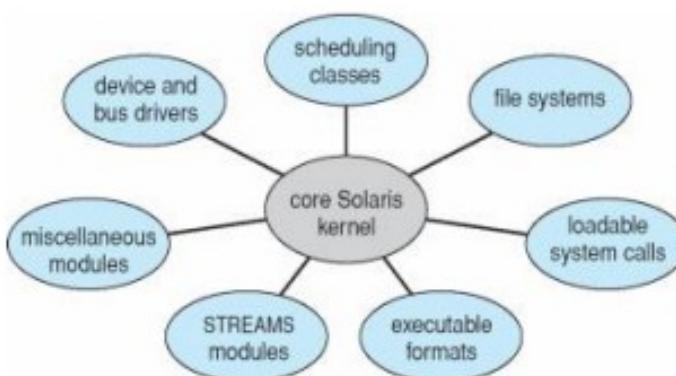
Sistemas Modulados

Los más modernos SOs implementan el kernel en módulos

- Usa un enfoque orientado a objetos
- Cada componente del núcleo está separado
- Los protocolos de comunicación entre ellos son sobre interfaces conocidas
- Cada uno es cargado en la medida que sea necesario dentro del kernel

En resumen, similar a capas pero más flexible

Un ejemplo es Solaris



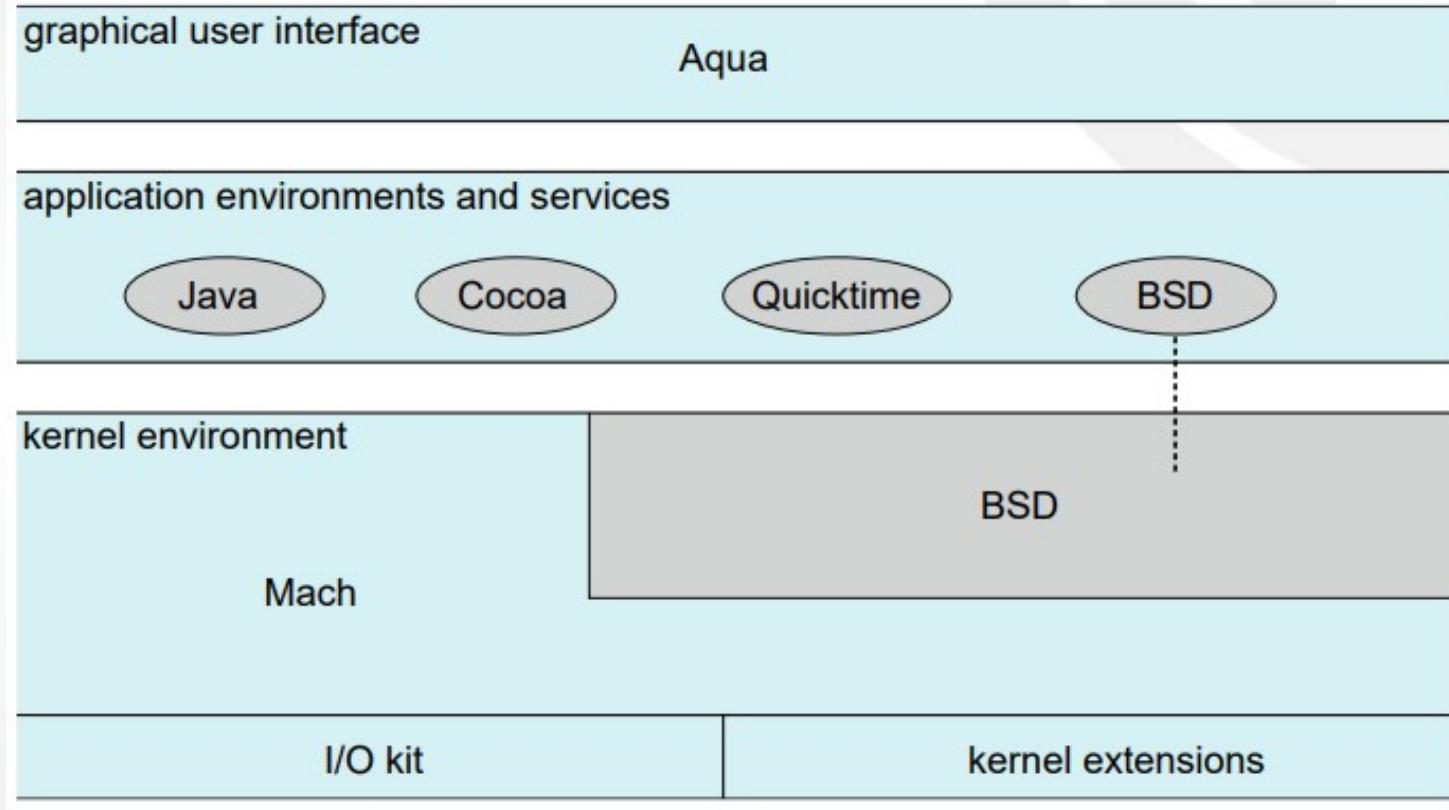
Sistemas Híbridos

Los sistemas operativos modernos no presentan un modelo puro.

Los modelos híbridos combinan multiples aproximaciones para alcanzar rendimiento, seguridad, usabilidad.

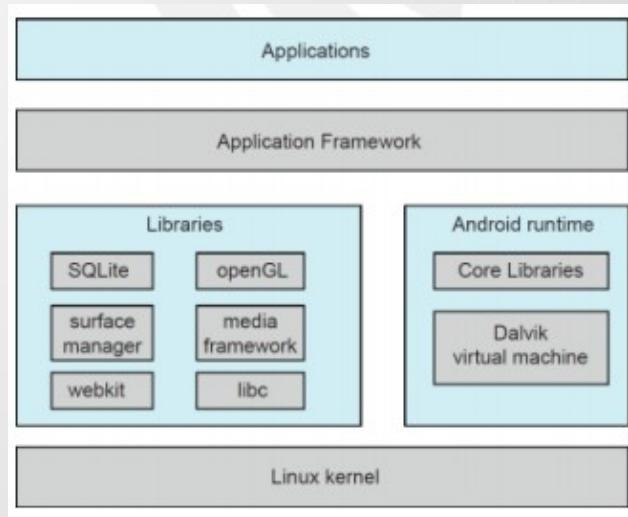
- Kernels de Linux y Solaris: en el espacio de direcciones del kernel presentan características monolíticas, además modulación para la carga dinámica de funcionalidades.
- Windows en su mayoría monolítico, además microkernel para diferentes subsistemas.
- Apple Mac OS X híbrido, por capas, Aqua UI más el ambiente de programación Cocoa.
- Kernel formado por un microkernel Mach y partes de BSD Unix, más un kit de E/S y la carga dinámica de módulos (llamados extensiones del kernel)

Sistemas Híbridos – MAC OS X



Sistemas Híbridos – Android

- Basado sobre un kernel Linux con modificaciones
 - Provee soporte para procesos, memoria, manejadores de dispositivos. Agrega administración de la energía
 - Runtime incluye librería para el conjunto del núcleo y la máquina virtual Dalvik.
 - Librerías incluyen frameworks para web browser (webkit), base de datos (SQLite), multimedia, pequeño libc.



Generación y Boot del sistema

- Los sistemas operativos son diseñados para ejecutar sobre diferentes clases de computadora. El sistema debe configurarse para cada computadora específica.
- Programa SYSGEN obtiene información sobre la especificación de hardware al momento de configurar el sistema.
- El SO debe estar disponible al hardware, entonces el hardware puede iniciar lo
 - Pequeñas piezas de código – bootstrap loader, localiza el kernel, lo carga en memoria, y lo pone en marcha
 - A veces es un proceso en dos pasos donde el boot block en una locación fija carga el bootstrap loader
 - Cuando se le da energía y se inicializa el sistema, comienza la ejecución a partir de una dirección fija de memoria
 - Firmware es usado para contener el código inicial de boot

Gestión de recursos

- Para que una PC pueda realizar las tareas solicitadas por el usuario requiere de la asignación de recursos para cada una de esas tareas. El sistema operativo administra los recursos que se deben asignar a los programas en ejecución.
- El sistema operativo administra la unidad central de procesamiento, los dispositivos de entrada y salida, la memoria principal o memoria RAM, los discos o memoria virtual, los procesos o programas en ejecución y en general todos los recursos del sistema.

SISTEMAS OPERATIVOS

Unidad 4

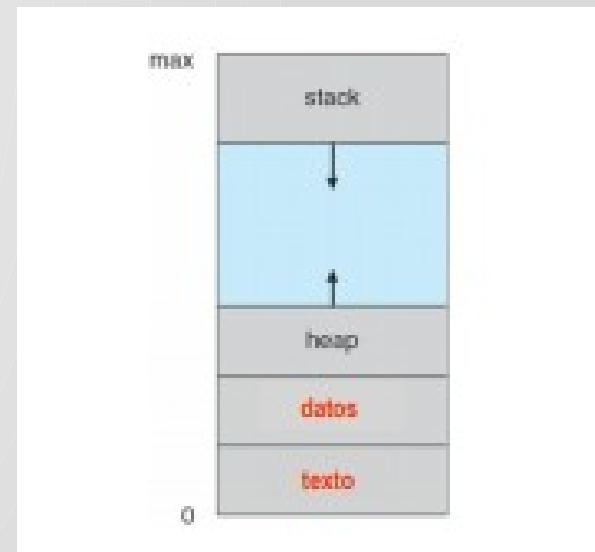
Procesos

Concepto de Proceso

- Un SO ejecuta una variedad de programas:
 - Sistema Batch – jobs
 - Sistemas de Tiempo Compartido – programas de usuario o tareas
- Proceso – un programa en ejecución.

Un proceso incluye:

- contador de programa
- stack
- sección de datos



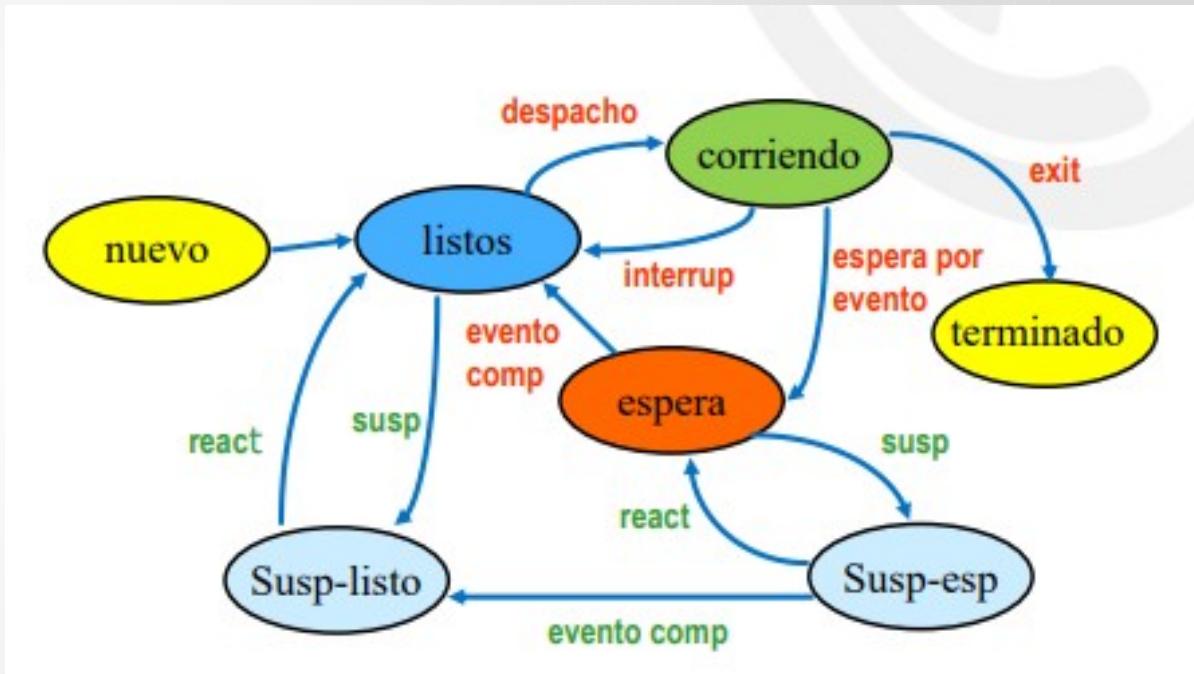
Estado de los procesos

- En tanto que un proceso ejecuta, cambia de estado
 - **NUEVO**: el proceso es creado.
 - **CORRIENDO** (ejecutando): las instrucciones están siendo ejecutadas.
 - **ESPERA**: el proceso está esperando que ocurra algún evento.
 - **LISTO**: el proceso está esperando ser asignado a la CPU.
 - **TERMINADO**: el proceso ha finalizado su ejecución.

Diagrama de Estados de un proceso – 3 estados



Diagrama de Estados de un proceso – 5 estados



Bloque de control de Procesos

Es una estructura de dato que contiene información asociada con cada proceso.

- Estado de Proceso
- Contador de Programa
- Registros de CPU
- Información de planificación de CPU
- Información de administración de memoria
- Información contable
- Información de estado E/S

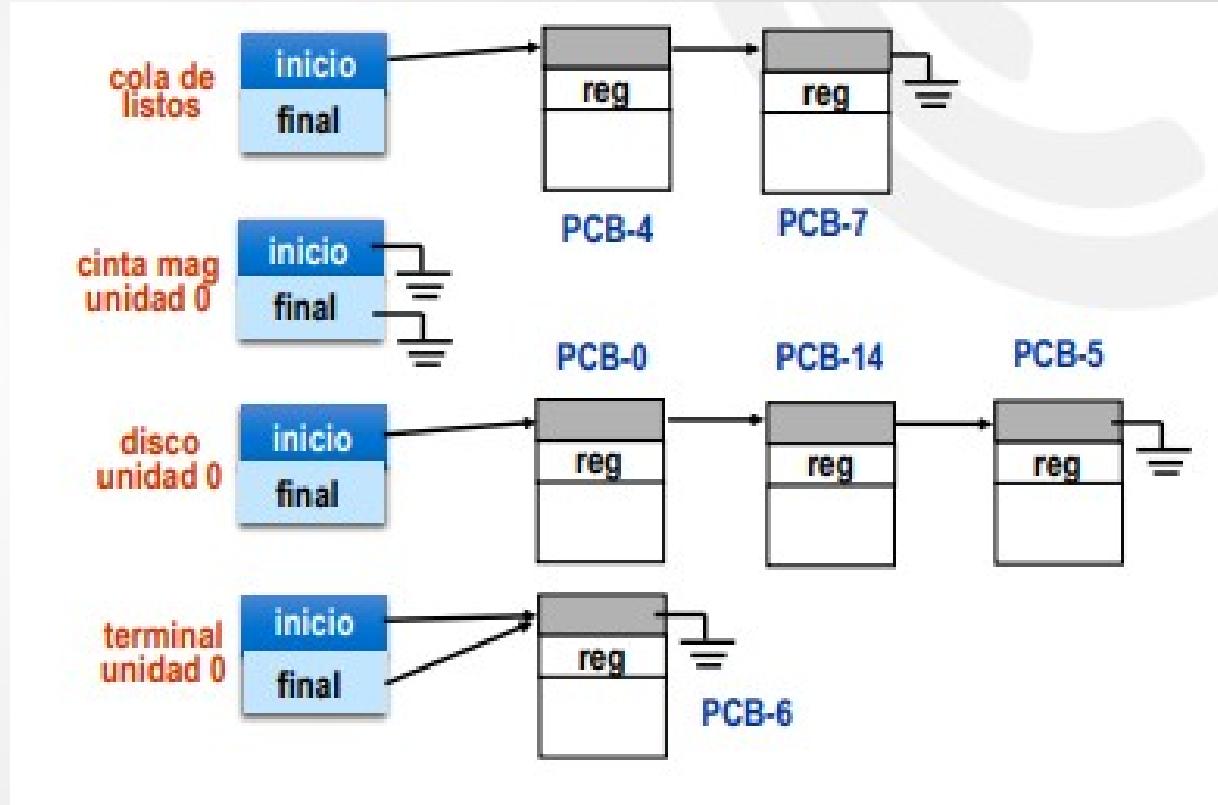
PCB: Process Control Block

estado proceso	prox previo
id proceso	
contador programa	
registros de CPU	
estructura memoria	
tabla de arch abiertos	
etc	

Colas de Planificación de Procesos

- Cola de Job (o tareas) – conjunto de todos los procesos en el sistema.
- Cola de listos – conjunto de todos los procesos residentes en memoria principal, listos y esperando para ejecutar.
- Colas de dispositivos – conjunto de procesos esperando por una E/S en un dispositivo de E/S.
- Migración de procesos entre las colas.

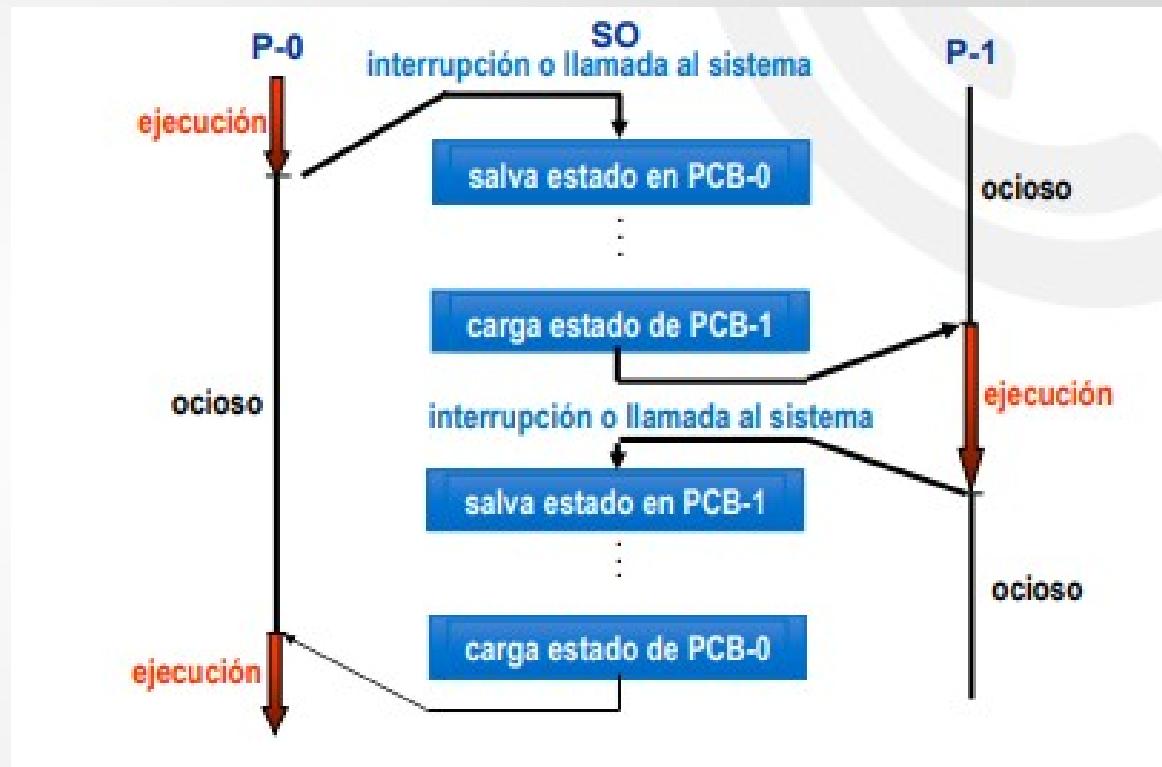
Colas de Listos y Dispositivos de E/S



Cambio de Contexto

- Cuando la CPU conmuta a otro proceso, el sistema debe salvar el estado del viejo proceso y cargar el estado para el nuevo proceso vía un cambio de contexto.
- El contexto de un proceso está representado en el PCB
- El tiempo que lleva el cambio de contexto es sobrecarga; el sistema no hace trabajo útil mientras está conmutando.
- El tiempo depende del soporte de hardware.

Commutacion de CPU de Proceso a Proceso



Creacion de Procesos

- Actividades
 1. Asignar un identificador de proceso único al proceso.
 2. Reservar espacio para proceso.
 3. Inicializar el PCB.
 4. Establecer los enlaces apropiados.
 5. Crear o expandir otras estructuras de datos.

Creación de Procesos - Políticas

- Espacio de direcciones
 - El hijo duplica el del padre.
 - El hijo tiene un programa cargado en él.
- Procesos padres crean procesos hijos, los cuales, a su vez crean otros procesos, formando un árbol de procesos.
- Ejecución
 - Padres e hijos ejecutan concurrentemente.
 - Padres esperan hasta que los hijos terminan.
- Recursos compartidos
 - Padres e hijos comparten todos los recursos.
 - Hijo comparte un subconjunto de los recursos del padre.
 - Padre e hijo no comparten ningún recurso.

Procesos en Windows

- Comandos
 - Crear Proceso: START <nombre del programa>
 - Termina Proceso: TASKKILL /IM <nombre del exe>
TASKKILL /PID <numero proceso>
 - Listar Procesos: TASKLIST | sort

Terminación de Procesos

- El proceso ejecuta la última sentencia y espera que el SO haga algo (exit).
 - Los datos de salida del hijo se pasan al padre (vía wait).
 - Los recursos de los procesos son liberados por el SO.
- El padre puede terminar la ejecución del proceso hijo (abort).
 - El hijo ha excedido los recursos alocados.
 - La tarea asignada al hijo no es mas requerida.
 - El padre está terminando.
 - El SO no permite a los hijos continuar si su padre termina.
 - Terminación en cascada.

Procesos Cooperativos

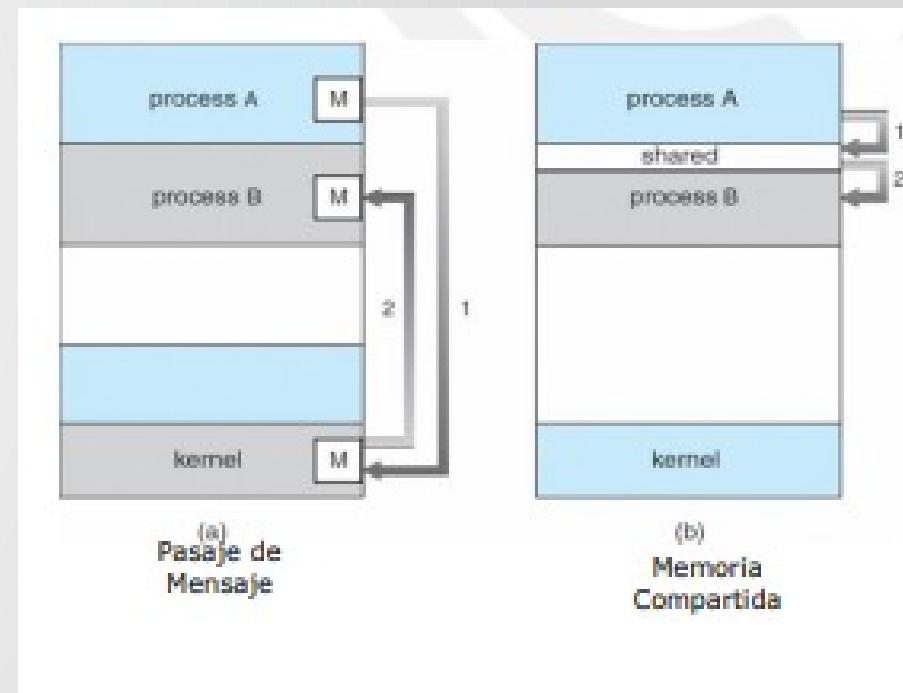
- Un proceso independiente no puede afectar ni ser afectado por la ejecución de otro proceso.
- Un proceso cooperativo puede afectar o ser afectado por la ejecución de otro proceso.
- Ventajas de los procesos cooperativos
 - Información compartida
 - Aceleración de la computación
 - Modularidad

Comunicación Interprocesos

- Los procesos cooperativos necesitan comunicación interprocesos (IPC)

Dos modelos de IPC

- Memoria compartida
- Pasaje de Mensajes



Problema del Productor-Consumidor

- Paradigma procesos cooperativos, el proceso productor produce información que es consumida por un proceso consumidor .
 - buffer ilimitado - no tiene límites prácticos en el tamaño del buffer.
 - buffer limitado supone que hay un tamaño fijo de buffer.

Comunicación Entre Procesos (IPC)

Mecanismo de los procesos para comunicarse y sincronizar sus acciones.

- Sistema de mensajes – los procesos se comunican uno con otro sin necesidad de variables compartidas.
- Las facilidades de IPC provee dos operaciones:
 - **send**(mensaje) – mensaje de tamaño fijo o variable
 - **receive**(mensaje)
- Si P and Q desean comunicarse, necesitan:
 - Establecer un vínculo de comunicación entre ellos
 - Intercambiar mensajes vía send/receive
- Implementación de un vínculo de comunicación
 - lógico (p.e., propiedades lógicas)
 - físico (p.e., memoria compartida, canal hardware)

Comunicación Directa

- Los procesos deben nombrar al otro explícitamente:
 - send (P, mensaje) – envía un mensaje al proceso P
 - receive(Q, mensaje) – recibe un mensaje del proceso Q
- Propiedades del vínculo de comunicación
 - Un vínculo está asociado con exactamente un par de procesos que se comunican.
 - Entre cada par existe exactamente un vínculo.
 - El vínculo puede ser unidireccional, pero es usualmente bidireccional.

Comunicación Indirecta

- Los mensajes son dirigidos y recibidos desde mailboxes
- Vínculo de comunicación
 - Se establece solo si los procesos comparten un mailbox común.
 - Puede ser asociado con muchos procesos.
 - Cada par de procesos puede compartir varios vínculos de comunicación.
 - Puede ser unidireccional o bi-direccional.
- Operaciones
 - crear un nuevo mailbox
 - enviar y recibir mensajes por medio del mailbox
 - destruir un mailbox
- Las primitivas son:
 - `send(A, message)` – enviar un mensaje al mailbox A
 - `receive(A, message)` – recibir un mensaje del mailbox A

Sincronización

- El pasaje de mensajes puede ser bloqueante o no bloqueante.
- Bloqueante es considerado sincrónico
 - Send bloqueante
 - Receive bloqueante
- No bloqueante es considerado asincrónico
 - Send no bloqueante
 - Receive no bloqueante

Buffering

- La cola de mensajes asociada al vínculo se puede implementar de tres maneras.
 - 1. Capacidad – 0 mensajes El enviador debe esperar por el receptor (rendezvous).
 - 2. Capacidad limitada – longitud finita de n mensajes El enviador debe esperar si el vínculo está lleno.
 - 3. Capacidad ilimitada – longitud infinita El enviador nunca espera.

Comunicación Cliente-Servidor

- Sockets
- Llamadas a Procedimientos Remotos (RPC:Remote Procedure Calls)
- Invocación a Métodos Remotos (RMI:Remote Method Invocation (Java))

Comunicación Entre Procesos - PIPE

- Actua como un conducto que permite que dos procesos se comuniquen
- Cuestiones:
 - Comunicación: unidireccional o bidireccional?
 - Comunicación bidireccional, ¿es half o full-duplex?
 - ¿Debe existir una relación (es decir, padre-hijo) entre los procesos de comunicación?
- Pipe ordinario: no se puede acceder desde fuera del proceso que lo creó. Normalmente, un proceso padre crea un pipe y la usa para comunicarse con un proceso hijo que creó.

Planificador de Procesos

- Planificador de largo término (o planificador de jobs) – selecciona que procesos deberían ser puestos en la cola de listos.
- Planificador de corto término (o planificador de CPU) – selecciona que procesos deberían ser próximamente ejecutados y colocados en la CPU.
- Planificador de mediano término, selecciona los procesos que se sacaran/introducirán temporalmente de/en la memoria principal (intercambio, swapping)

Comunicación entre Procesos - PIPE

- Son utilizados para comunicaciones unidireccionales
- Permiten la comunicación en el estilo estándar de productor-consumidor
- El productor escribe en un extremo (el final de la tubería)
- El consumidor lee desde el otro extremo (el extremo de lectura de la tubería)
- Requieren una relación padre-hijo entre los procesos de comunicación



PIPE

- Están implementadas en forma muy eficiente en los sistemas operativos multitarea, iniciando todos los procesos al mismo tiempo, y atendiendo automáticamente los requerimientos de lectura de datos para cada proceso cuando los datos son escritos por el proceso anterior. De esta manera el planificador de corto plazo va a dar el uso de la CPU a cada proceso a medida que pueda ejecutarse minimizando los tiempos muertos.
- Para mejorar el rendimiento, la mayoría de los sistemas operativos implementan los pipes o tunerias usando búferes, lo que permite al proceso proveedor generar más datos que lo que el proceso consumidor puede atender inmediatamente.

Planificador de Procesos

- El planificador de corto término es invocado muy frecuentemente (milisegundos) -> (debe ser rápido).
- El planificador de largo término es invocado poco frecuentemente (segundos, minutos) -> (puede ser muy lento).
- El planificador de medio término controla el grado de multiprogramación.
- Los procesos pueden ser descriptos como:
 - Procesos limitados por E/S – pasa más tiempo haciendo E/S que computaciones, ráfagas (burst) de CPU muy cortas.
 - Procesos limitados por CPU – pasa más tiempo haciendo computaciones que E/S, ráfagas (burst) de CPU muy largas.

Planificador de CPU

- Escoge un proceso de entre los que están en memoria listos para ejecutarse y le asigna la CPU al proceso elegido
- La decisión de planificación puede ocurrir
 - 1. Cuando un proceso pasa de ejecución a espera
 - 2. Cuando un proceso pasa de ejecución a listo
 - 3. Cuando un proceso pasa de espera a listo
 - 4. Cuando un proceso termina
- Un planificador es no expropiativo cuando solo planifica en los casos 1 y 4
- En otro caso decimo que es el planificador es expropiativo

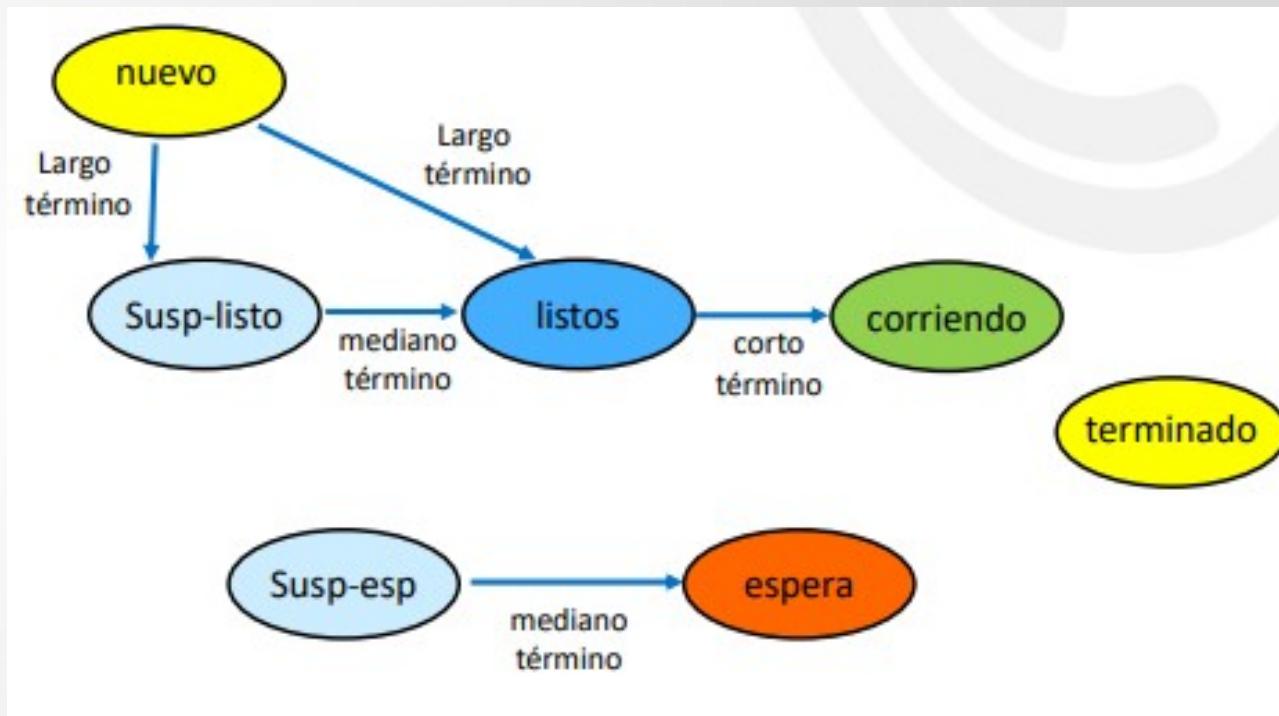
Criterios de Planificación

- Utilizacion de CPU – mantener la CPU tan ocupada como sea posible (maximizar)
- Rendimiento – numero de procesos que se completan por unidad de tiempo (maximizar)
- Tiempos de retorno – tiempo transcurrido desde que se presenta el proceso hasta que se completa (minimizar)
- Tiempo de espera – tiempo que un proceso pasa en la cola de procesos listos esperando la CPU (minimizar)
- Tiempo de respuesta – tiempo que tarda un proceso desde que se le presenta una solicitud hasta que produce la primera respuesta (minimizar)

Políticas de Planificación

- Planificación Round-robin
- Menor tiempo de respuesta primero (EDF (Earliest deadline first scheduling))
- FIFO - También conocido como FCFS "First Come, First Served".
- LIFO «Last In First Out».
- SJF - Shortest Job First.
- CFS - Completely Fair Scheduler (ó Planificador Completamente Justo)
- SRT - Shortest Remaining Time
- SPT - Shortest Process Time
- Planificación mediante colas multinivel.

Planificacion y Transicion de Estados de un Proceso



SISTEMAS OPERATIVOS

Unidad 3

Virtualización

Virtualización

¿Qué es virtual?

- Dícese de lo que tiene virtud para producir un efecto, aunque no lo produce de presente. (diccionario).
- Que no tiene existencia real sino aparente (diccionario).

¿Porqué virtualizar?

- Reduce el costo e incrementa la eficiencia de los existentes recursos de hardware

Virtualización

- Lograr más en menos tiempo
 - Ejecute varios sistemas operativos en una sola computadora
 - Reduzca el número de computadores físicos que se requieren
- Facilitar la migración de aplicaciones
- Agilizar la implementación
 - Pruebe nuevo software y sistemas operativos antes de su implementación

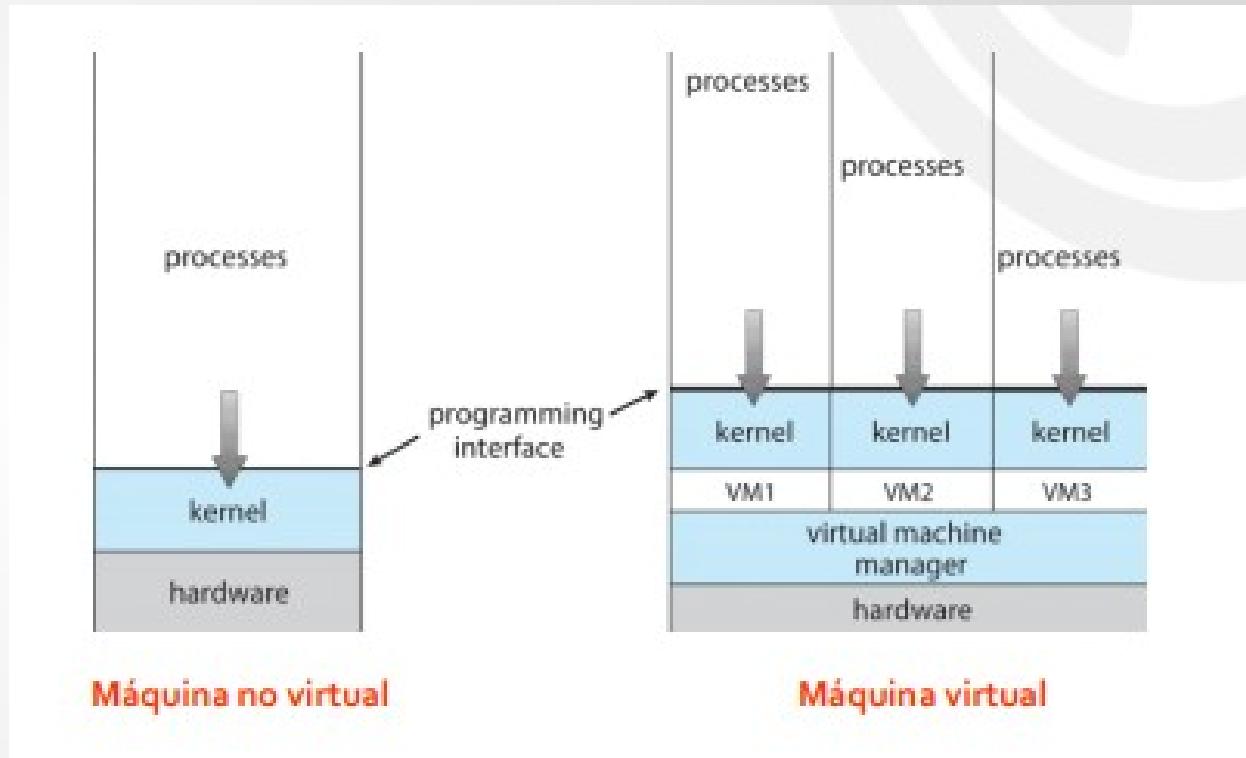
Virtualización

- Acelerar el desarrollo de aplicaciones
 - Incrementa el aseguramiento de calidad al probar en diferentes sistemas operativos utilizando máquinas virtuales
 - Reduzca el tiempo para salir al mercado con menos reconfiguración

Conceptos

- La idea central de una máquina virtual es la abstracción del hardware de una computadora en varios ambientes de ejecución diferentes, creando la ilusión de que cada ambiente de ejecución está corriendo en su propia computadora privada.
- Una máquina virtual provee una interfaz idéntica al hardware primitivo subyacente.
- El sistema operativo crea la ilusión de múltiples procesos, cada uno ejecutando en su propio procesador con su propia memoria (virtual).
- Cada invitado es provisto con una copia (virtual) de la computadora

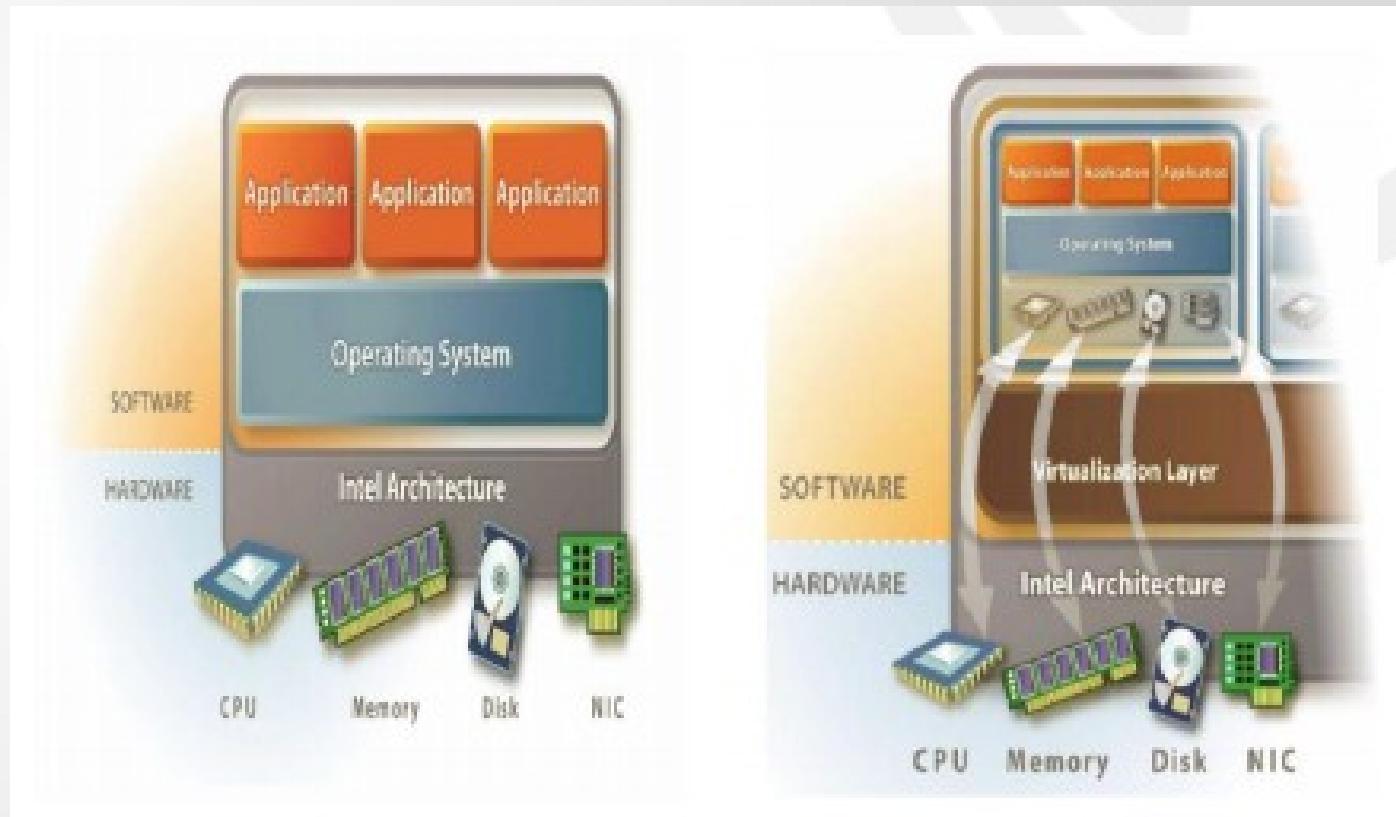
Conceptos



Historia y Beneficio

- Aparecieron comercialmente en las mainframes de IBM en 1972
- Fundamentalmente, múltiples ambientes de ejecución (diferentes SOs) pueden compartir el mismo hardware
- Están protegidos uno de otro
- Puede permitirse, en forma controlada, compartir archivos
- Conmuta uno con otro sistemas físicos vía red
- Útil para desarrollo, testing
- “Open Virtual Machine Format”, un formato standard de máquinas virtuales, permite a una VM correr dentro de diferentes plataformas (host) de máquinas virtuales

Virtualización



Virtual Machine Manager

Crea, administra y ejecuta las maquinas virtuales

Clasificación

- **Tipo 0** – son soluciones basados en hardware, que proveen soporte para la creación y administración via el firmware.
- **Tipo 1** – Hypervisors ejecutan directamente sobre el hardware de la máquina.
- **Tipo 2** – Hypervisors ejecutan sobre el sistema operativo host que provee los servicios de virtualización.

Virtual Machine Manager

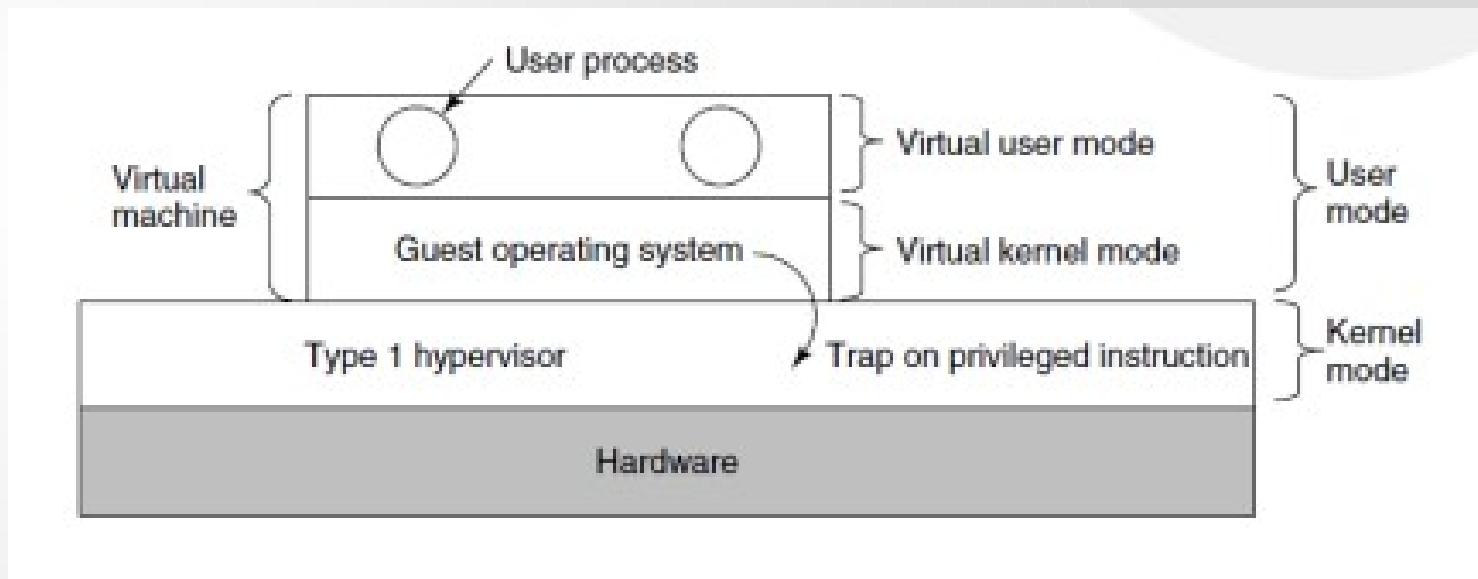
Otras variaciones:

- Paravirtualización
- Ambiente de Programación Virtualizado
 - Utilizado por Oracle Java y Microsoft.Net
- Emuladores
- Contenedor de Aplicación
 - Por ejemplo: Oracle Solaris Zones, BSD Jails, IBM AIX WPARs, Docker

VMM - Implementación

Técnicas para implementar

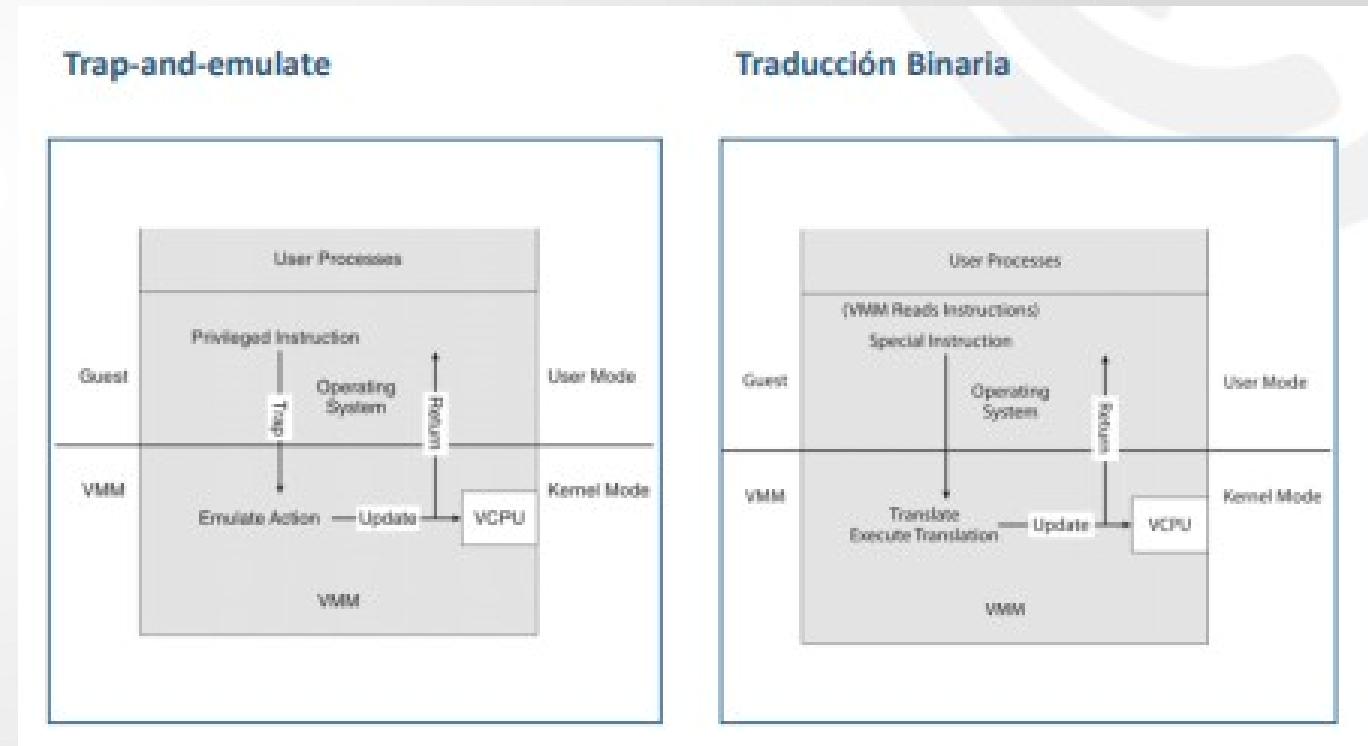
- Trap-and-emulate
- Traducción binaria



VMM - Implementación

Técnicas para implementar

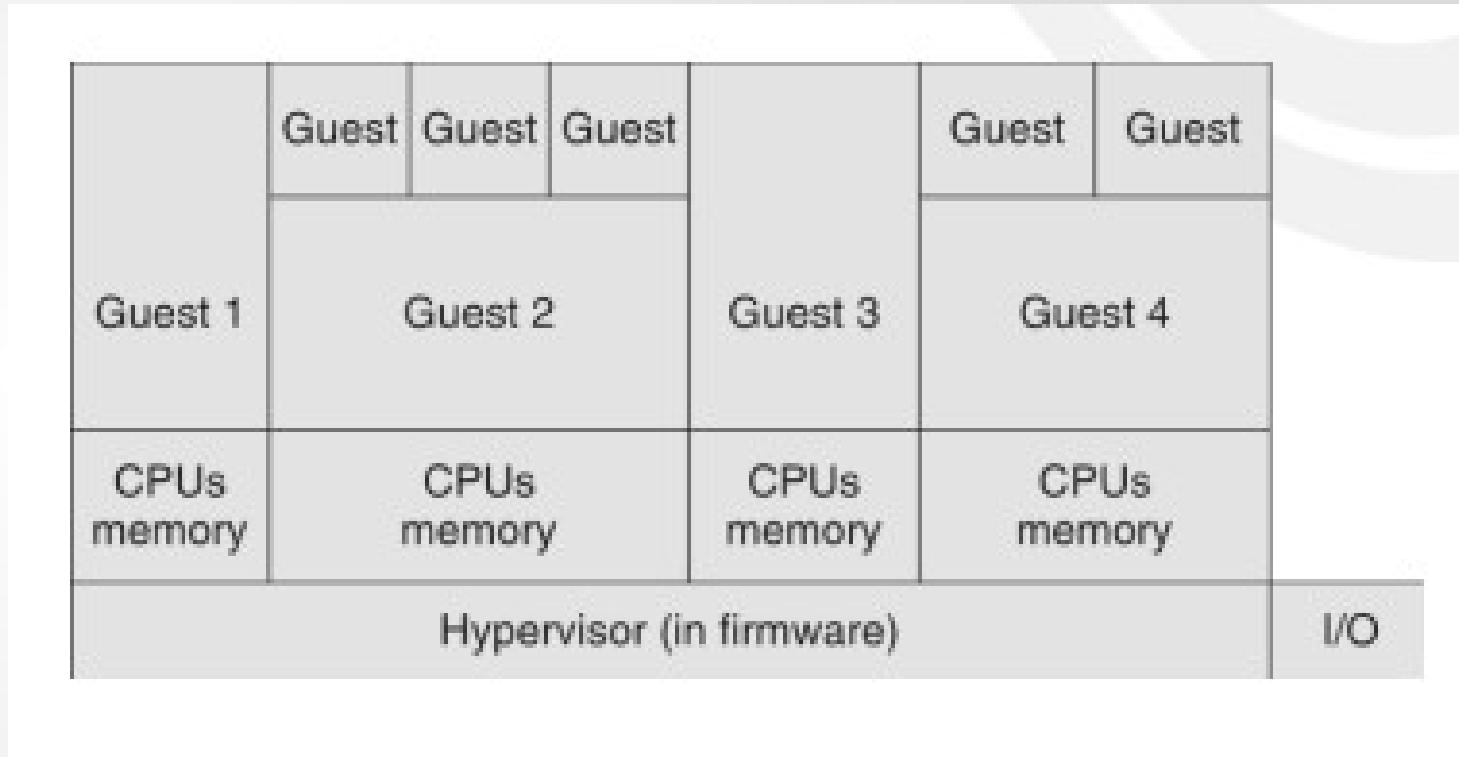
- Trap-and-emulate
- Traducción binaria



Hypervisor

- El **hipervisor** es básicamente una capa de software localizada entre el hardware y el sistema operativo. A través de él, varias máquinas virtuales pueden tener acceso a los recursos de hardware como CPU, memoria, red, almacenamiento, etc. Además, esta característica puede garantizar una mayor seguridad para las máquinas virtuales a partir de mecanismos como aislamiento, encapsulación y particionamiento

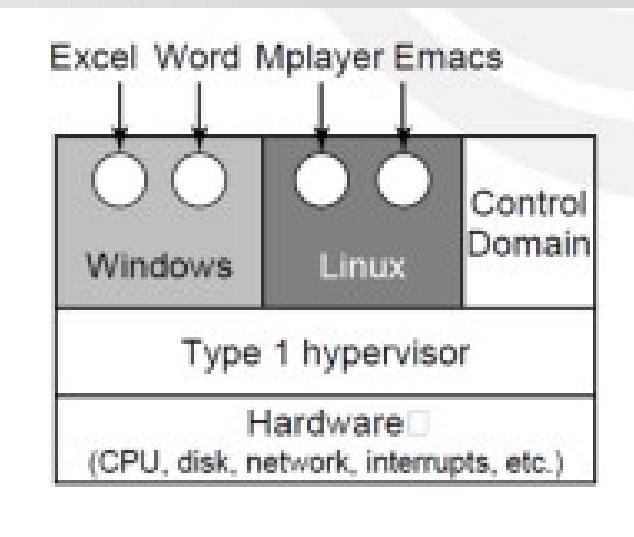
VMM - Tipo 0



- Ejemplos
 - OVMF (Open Virtual Machine Firmware)

VMM - Tipo 1

- Ejecutan en modo kernel
- Proveen
 - planificación de CPU,
 - administración de memoria,
 - administración de E/S,
 - protección,
 - seguridad

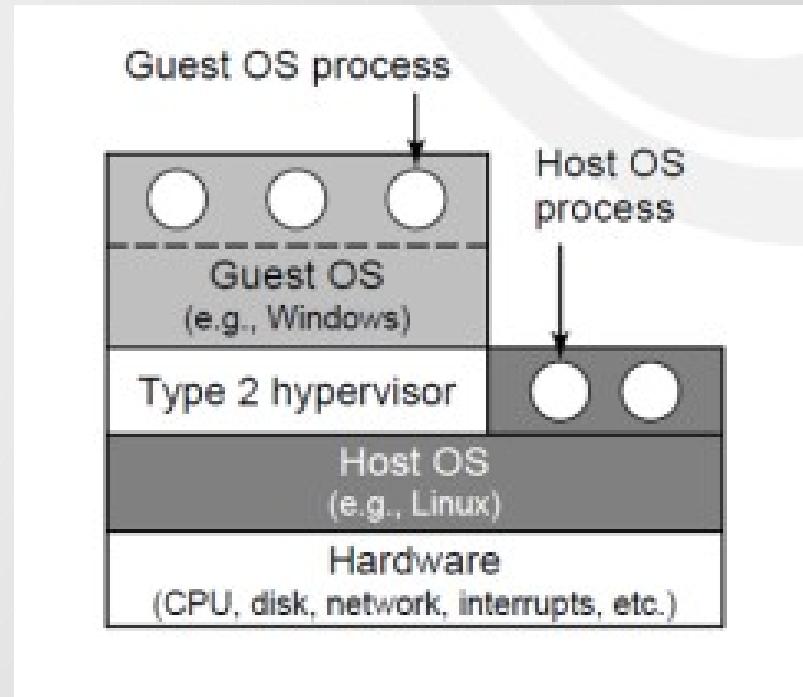


VMM – Tipo 1

- Los hipervisores tipo 1 son un sistema operativo en sí mismo, uno muy básico sobre el que se ejecutan máquinas virtuales. Esto significa que la máquina física en la que se ejecuta el hipervisor sirve solo para propósitos de virtualización. No podrás utilizarlo para nada más.
- Ejemplos
 - VMware ESXi
 - KVM (Máquina Virtual Basada en Kernel)
 - Oracle VM
 - Microsoft Hyper-V

VMM – Tipo 2

- A nivel de aplicación
- Menor rendimiento por las capas que contiene



VMM – Tipo 2

Este tipo de hipervisor se ejecuta dentro de un sistema operativo de una máquina host física.

Es por esto que llamamos hipervisores. A diferencia de los hipervisores de tipo 1 que se ejecutan directamente en el hardware, los hipervisores alojados tienen una capa de software debajo, el Sistema Operativo

Ejemplos

- Oracle VM VirtualBox
- VMware Workstation Pro / VMware Fusion
- Windows Virtual PC
- Parallels Desktop

Paravirtualización

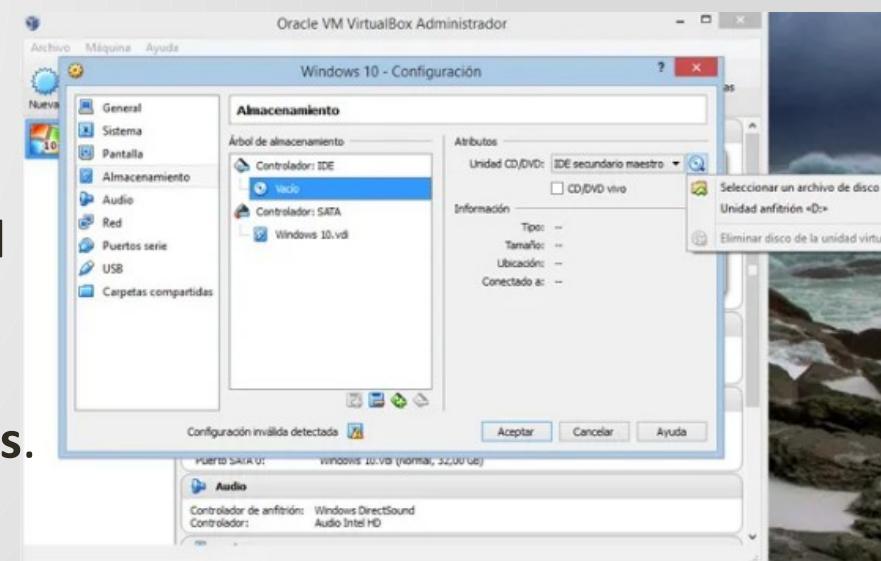
- Una técnica en la cual el Sistema Operativo invitado es modificado para trabajar en cooperación con el VMM (Virtual machine manager) para optimizar el rendimiento.
- Los invitados deben ser modificados para correr en un hardware paravirtualizado.
- Es una técnica moderna ejecución virtual que consiste en permitir algunas llamadas directas al hardware mermando así la penalización en rendimiento que la ejecución 100% virtual implica. Esto es posible gracias a características que los procesadores modernos tienen, ej: Intel tiene Intel VT y AMD tiene AMD-V. Estas APIs ofrecen instrucciones especiales que el software de virtualización puede emplear para permitir una ejecución más eficiente.

Software mas usado

- VirtualBox (Windows/Linux/Mac, gratis)
- VMware Workstation Player (Windows/Linux/Mac, básico: gratis, pro: \$200+)
- Hyper-V
- Parallels Desktop
- KVM
- Qemu
- Xen

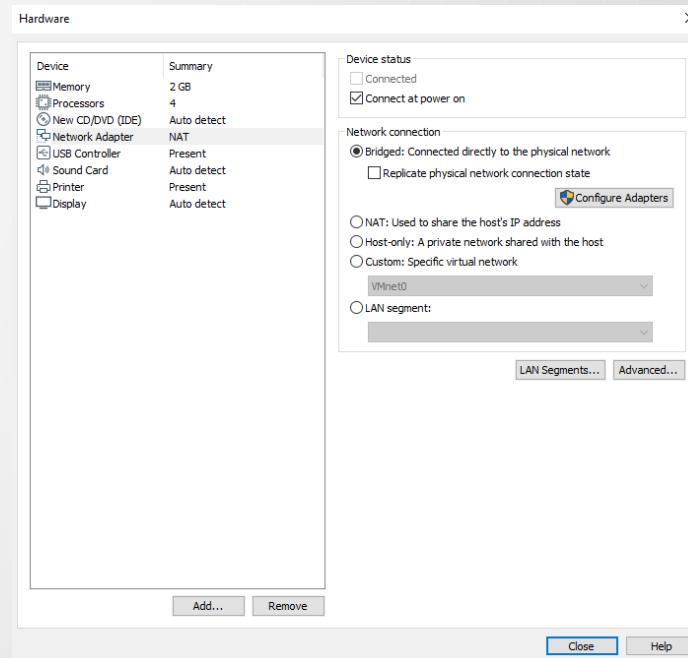
Virtual Box

- Parte de su popularidad viene dada porque es gratuito. Además su funcionamiento es bastante más sencillo que lo ofrecido por otras soluciones de este mismo tipo.
- Cuenta con una buena cantidad de parámetros personalizables que, junto a las descripciones de las máquinas virtuales, se almacenan en archivos de texto en formato **XML**. Esto nos permitirá facilitar su portabilidad a otros equipos, por ejemplo. Además nos permite la instalación de software en la **máquina virtual** con permisos adicionales al original para poder compartir archivos, unidades y **periféricos**.



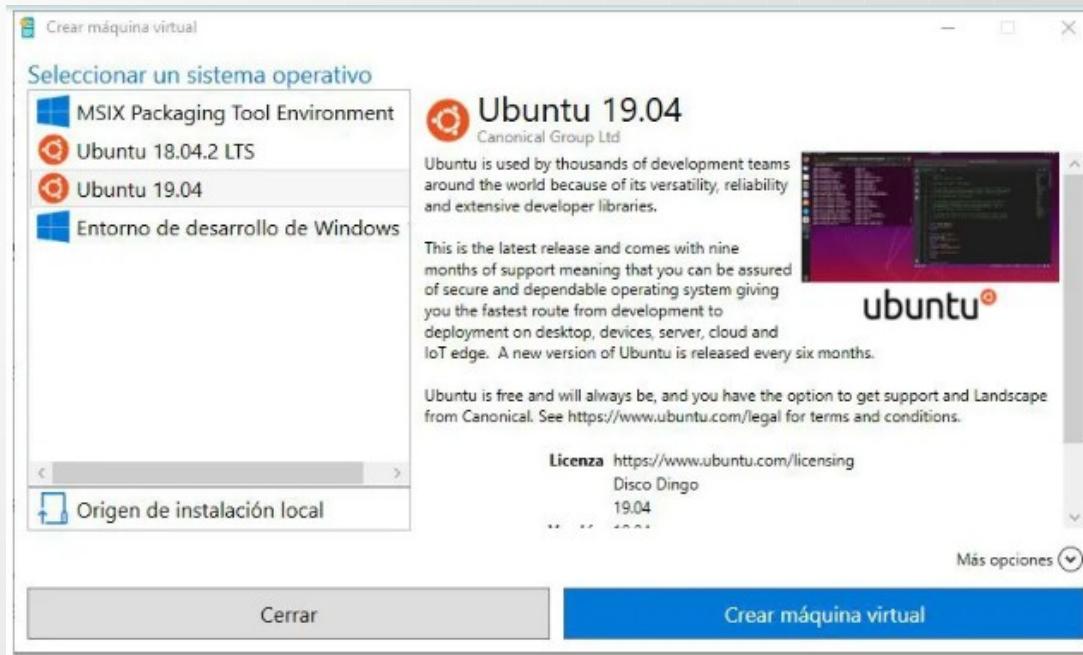
VMWare

- Es una herramienta ideal para ejecutar una única máquina virtual en un equipo con Windows o Linux.
- La version gratuita permite la ejecucion de una unica maquina



Hyper-V

- Esta es la aplicación para crear máquinas virtuales que de **Microsoft**. En un principio se incluyó en Windows Server 2008 R2. Más tarde se incluyó en las versiones Pro y Enterprise de Windows 8, **Windows 8.1** y Windows 10 con arquitectura de **64 bits**.

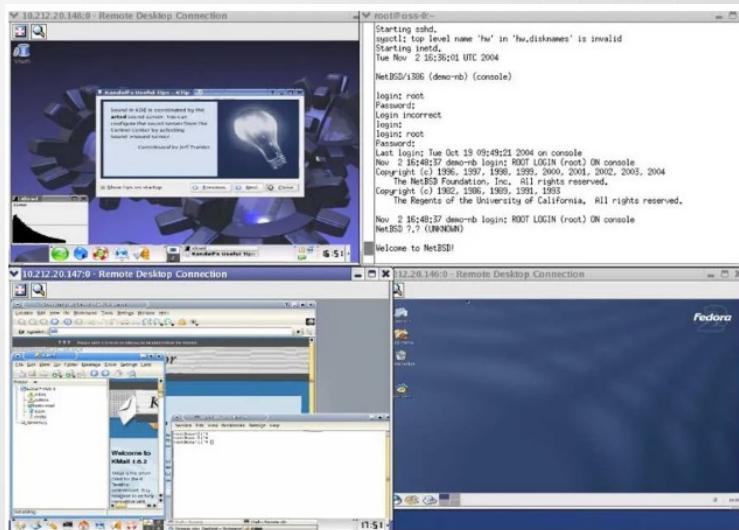


KVM

- De este modo lo que logramos es emular máquinas de distinto **hardware** y además admite el cambio automático del tamaño de los discos virtuales que creamos en el proceso. A todo ello debemos sumarle que **Qemu** puede ejecutarse en hosts sin permisos de administrador, lo que es un punto diferenciador con respecto al resto de propuestas. Con esto logramos que esta solución sea más que adecuada para la creación de máquinas virtuales portables.

XEN

- Este es un proyecto de código abierto y que se ha enfocado especialmente para el uso más profesional y empresarial. Además presenta un **sistema de virtualización** seguro con control de recursos del host. Decir que también se ha diseñado para alcanzar un alto rendimiento en el proceso de virtualización.



SISTEMAS OPERATIVOS

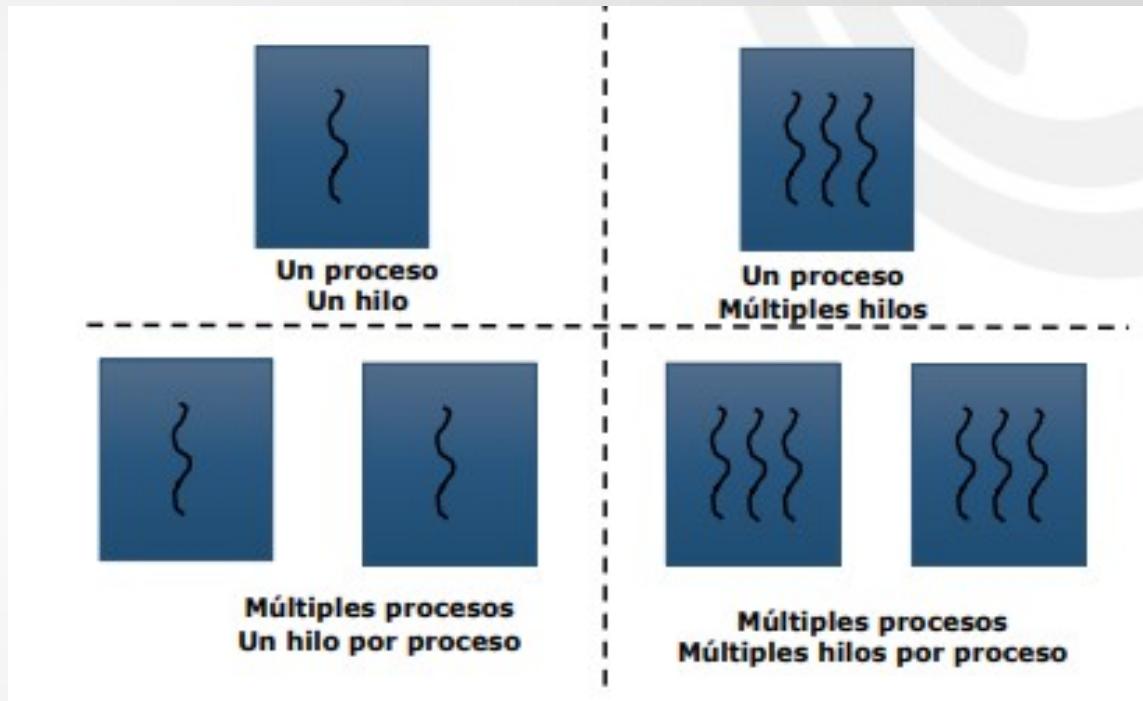
Unidad 5

Hilos

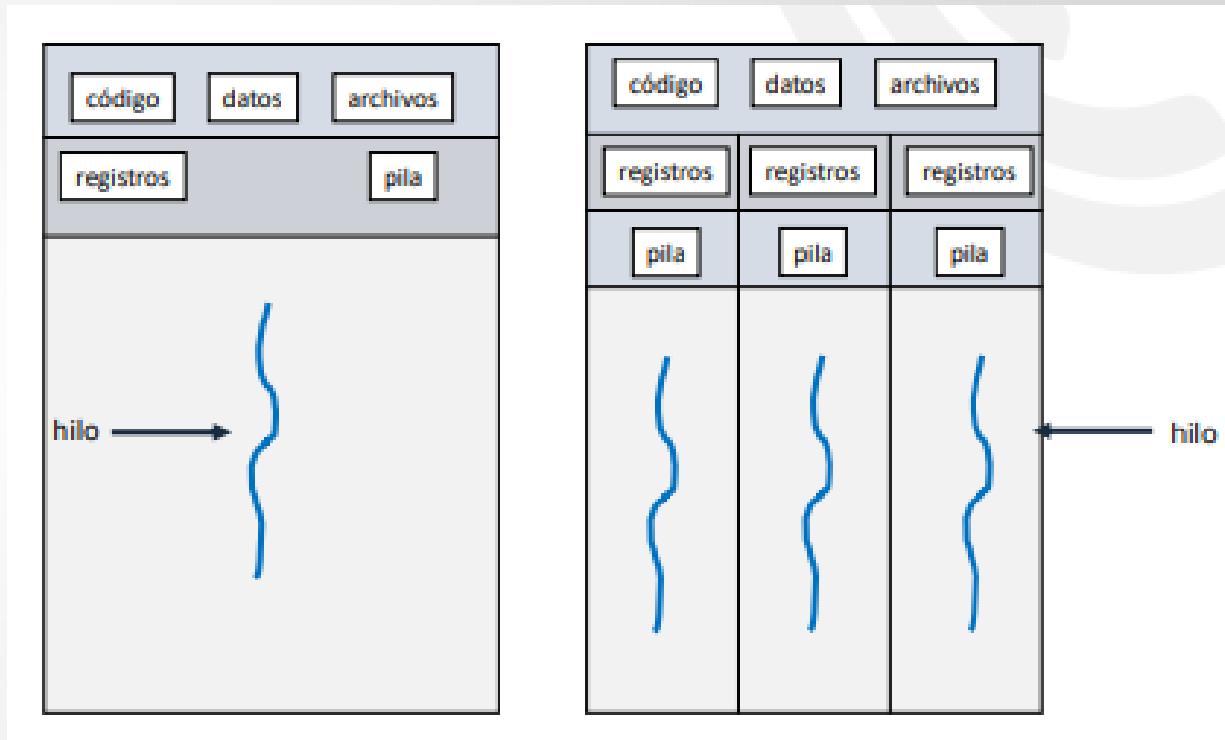
Concepto de Hilo

- Un hilo (thread o proceso de peso liviano) es una unidad básica de utilización de CPU, consiste de:
 - contador de programa
 - conjunto de registros
 - espacio de pila
- Un hilo comparte con sus hilos compañeros (colectivamente conocidos como tarea o task) su:
 - sección de código
 - sección de datos
 - recursos del SO.
- Un proceso tradicional o peso pesado es igual a una tarea con un solo hilo.

Hilos



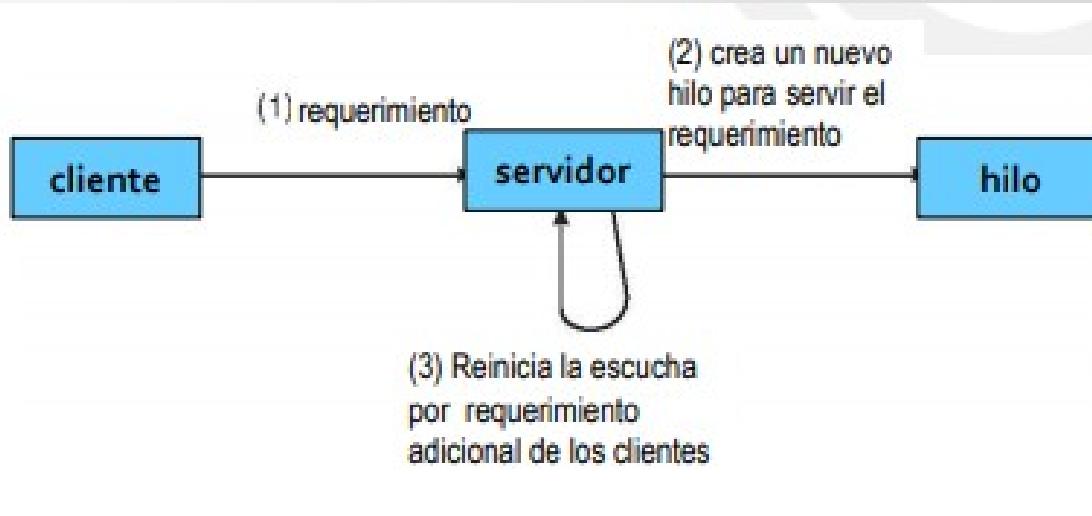
Proceso Mono y Multihilo



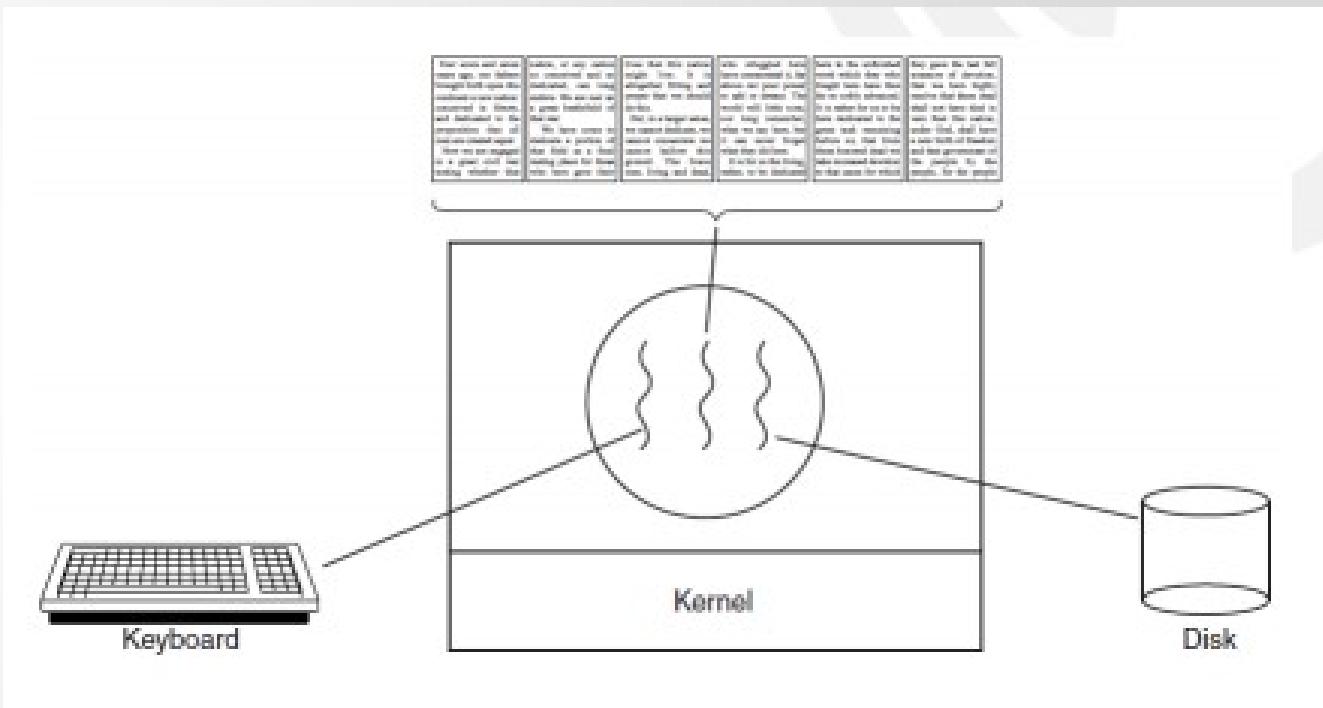
Hilos

- En una tarea con múltiple hilos, mientras un hilo servidor está bloqueado y esperando, un Segundo hilo de la misma tarea puede estar corriendo.
- Cooperación de múltiple hilos en una misma tarea confiere alto procesamiento total y mejora el rendimiento.
 - Aplicaciones que requieren compartir un buffer común (p.e., productor-consumidor) se benefician con la utilización de hilos.
- Los hilos proveen un mecanismo que permite a procesos secuenciales hacer llamadas al sistema bloqueantes mientras que también logra paralelismo.

Ejemplo Arquitectura de Servidor

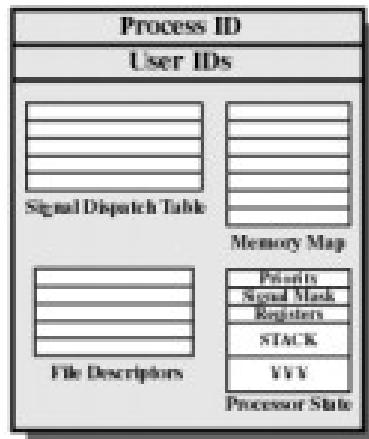


Ejemplo Procesador de Texto

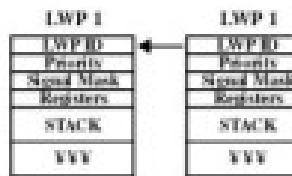
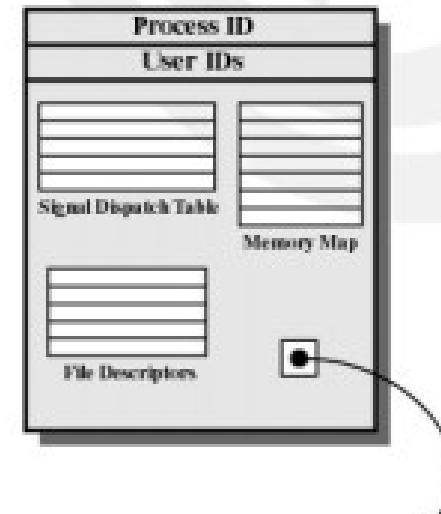


Proceso Mono y Multihilos

UNIX Process Structure



Solaris Process Structure



Beneficios

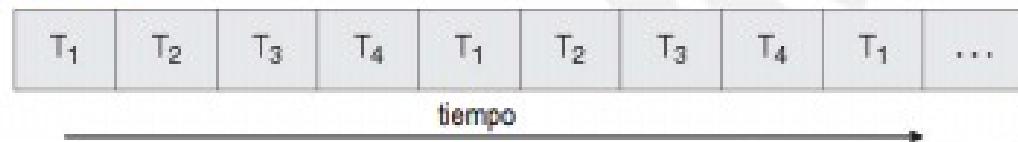
- Capacidad de Respuesta
- Compartir Recursos
 - Dado que los hilos dentro de un mismo proceso comparten memoria y archivos, pueden comunicarse unos con otros sin invocar al kernel
- Economía
- Toma menos tiempo crear un nuevo hilo que un proceso
 - Menos tiempo terminar un hilo que un proceso
 - Menos tiempo en comutar entre dos hilos dentro del mismo proceso
- Utilización de Arquitecturas Multiprocesador
- Escalabilidad

Programación Multicore

- Los sistemas multicore ponen presión sobre los programadores, estos desafíos incluyen:
 - Dividir actividades
 - Balance
 - Partición de datos
 - Dependencia de los datos
 - Verificación y depuración

Ejecución Concurrente y Paralela

Un solo núcleo



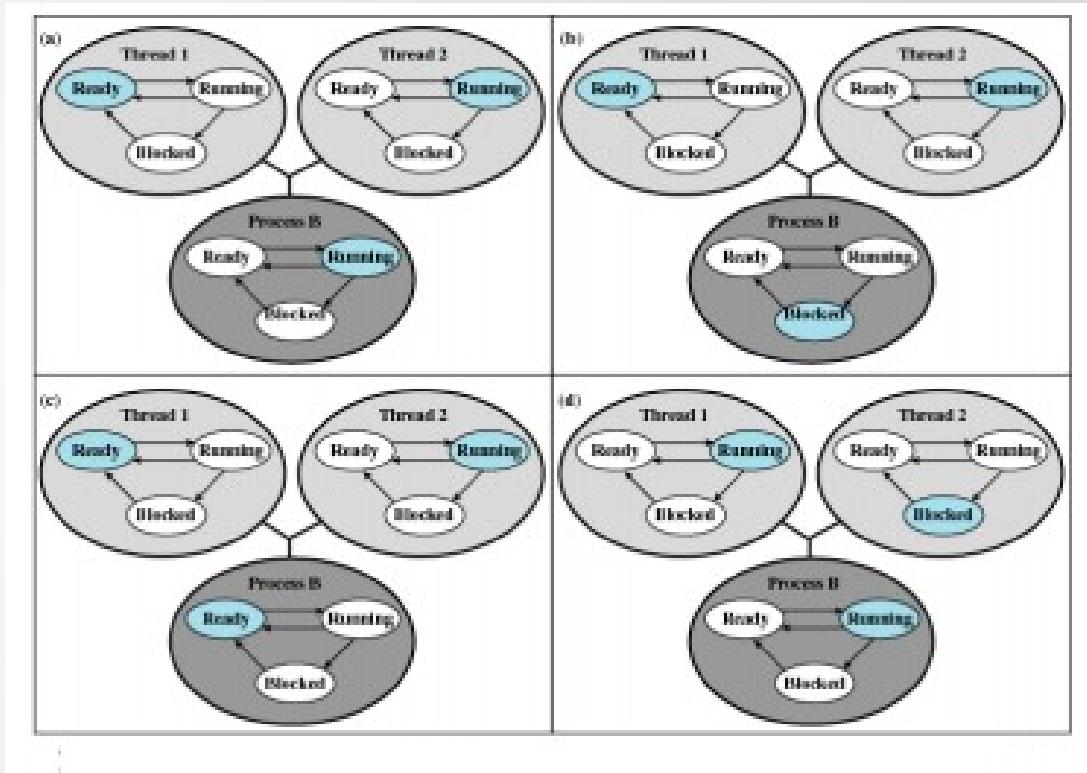
Múltiples núcleos



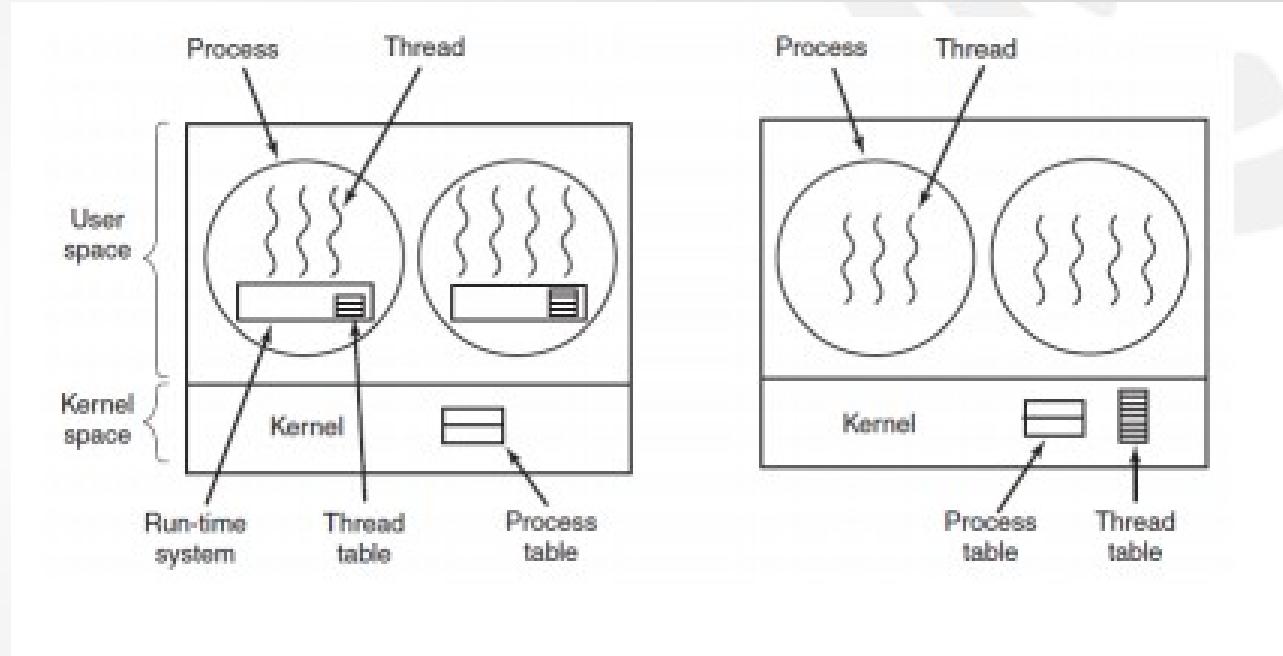
Hilos - Clasificación

- A nivel de USUARIO – la administración es realizada por librerías a nivel de usuario.
 - Tres librerías primarias:
 - POSIX Pthreads
 - Win32 threads
 - Java threads
- A nivel de KERNEL – la administración es realizada por el sistema operativo
 - Ejemplos:
 - Windows XP/2000/Vista/7/8/10
 - Solaris (de Sun, ahora Oracle)
 - Tru64 UNIX (de Digital, luego Compaq, finalmente HP)
 - Mac OS X (Apple)

Hilos a Nivel de Usuario – Relación estado Proceso e Hilo



Hilos a Nivel Usuario

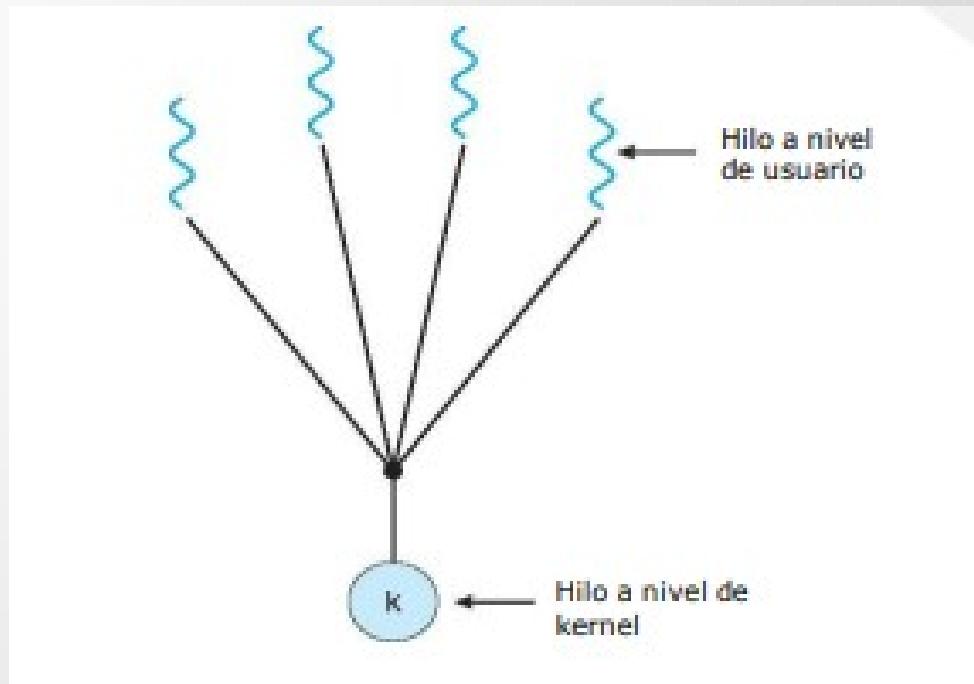


Modelos de Multihilados

- Muchos-a-Uno
- Uno-a-Uno
- Muchos-a-Muchos

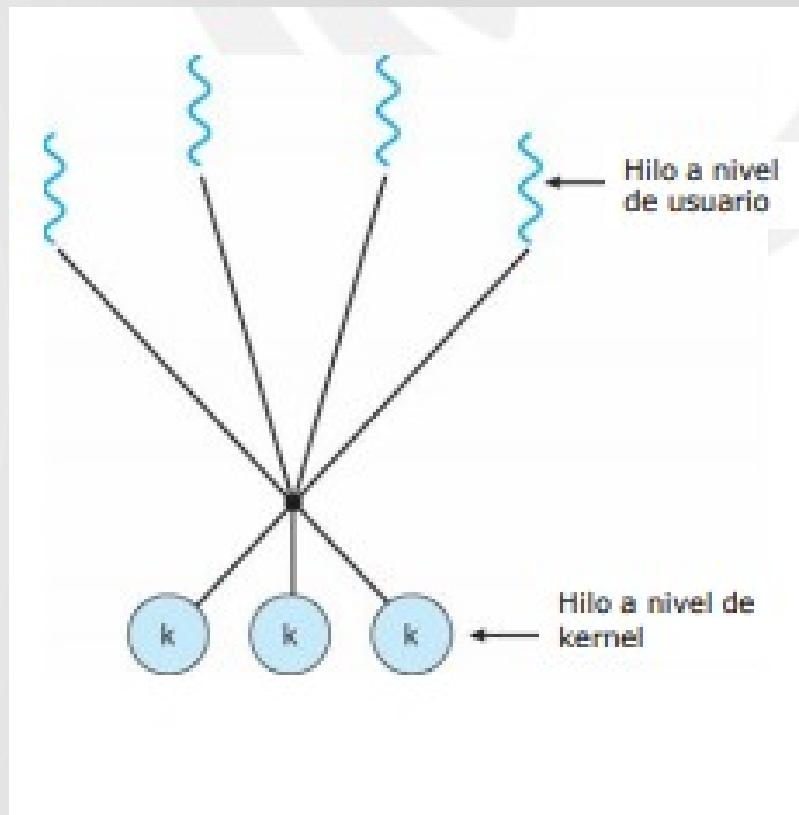
Muchos a Uno

- Muchos hilos a nivel de usuario mapean a un hilo a nivel de kernel.
- Usado en sistemas que no soportan hilos a nivel kernel.



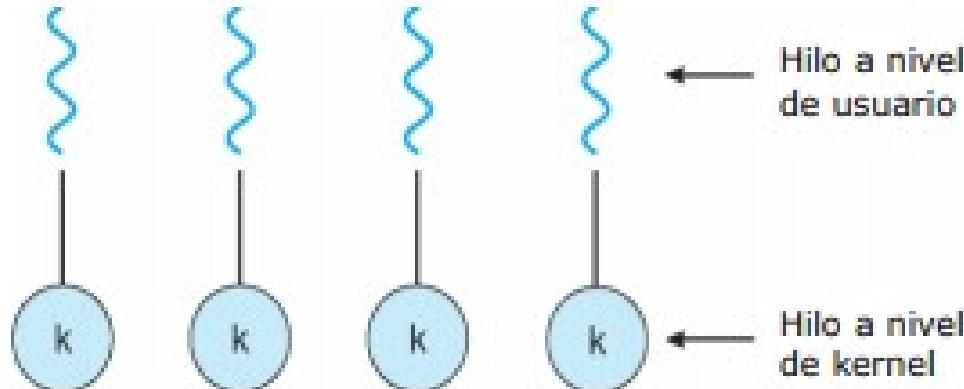
Muchos a Muchos

- Permite que muchos hilos a nivel de usuario mapeen a muchos hilos a nivel de kernel
- Permite al SO crear un número suficiente de hilos a nivel de kernel
- Solaris antes de la versión 9
- Windows NT/2000 en adelante con paquete ThreadFiber



Uno a Uno

- Cada thread nivel usuario mapea a un thread kernel.
- Ejemplos
 - Windows NT/XP/2000 y los que siguen
 - Linux
 - Solaris 9 y los que siguen



Librería de Hilos

- Las librerías de hilos proveen a los programadores con APIs para crear y administrar hilos
- Dos formas primarias de implementarlas
 - Librerías enteramente en espacio de usuario
 - Librería a nivel de Kernel soportada por el SO

Librería PThreads

- Pueden ser provistas sea a nivel de usuario como a nivel de kernel
- Es un standard POSIX (IEEE 1003.1c) API para creación y sincronización de hilos
- Las API especifican el comportamiento de la librería de hilos
- Común en SOs UNIX (Solaris, Linux, Mac OS X)

Manejo de Signal

- Los Signals son usados en UNIX para notificar a un proceso que un particular evento ha ocurrido
- Un signal handler es usado para signals a procesos
 - 1. El Signal es generado por un particular evento
 - 2. El Signal es enviado a un proceso
 - 3. El Signal es manejado
- Opciones:
 - Enviar el signal al hilo sobre el cual el signal se aplica
 - Enviar el signal a cada hilo en el proceso
 - Enviar el signal a ciertos hilos en el proceso
 - Asignar un hilo específico para recibir todos los signals al proceso

Cancelación de Hilos

- Terminar un hilo antes que finalice
- Dos propuestas generales:
 - Cancelación asincrónica termina el hilo señalado inmediatamente
 - Cancelación Diferida permite al hilo señalado verificar periódicamente si debería ser cancelado

Pools de Hilos

- Crea un número de hilos en un pool donde esperan por trabajo
- Ventajas:
 - Usualmente es ligeramente mas rápido servir un requerimiento con un hilo existente que crear uno nuevo
 - Permite que el número de hilos de la aplicación sea limitado al tamaño del pool

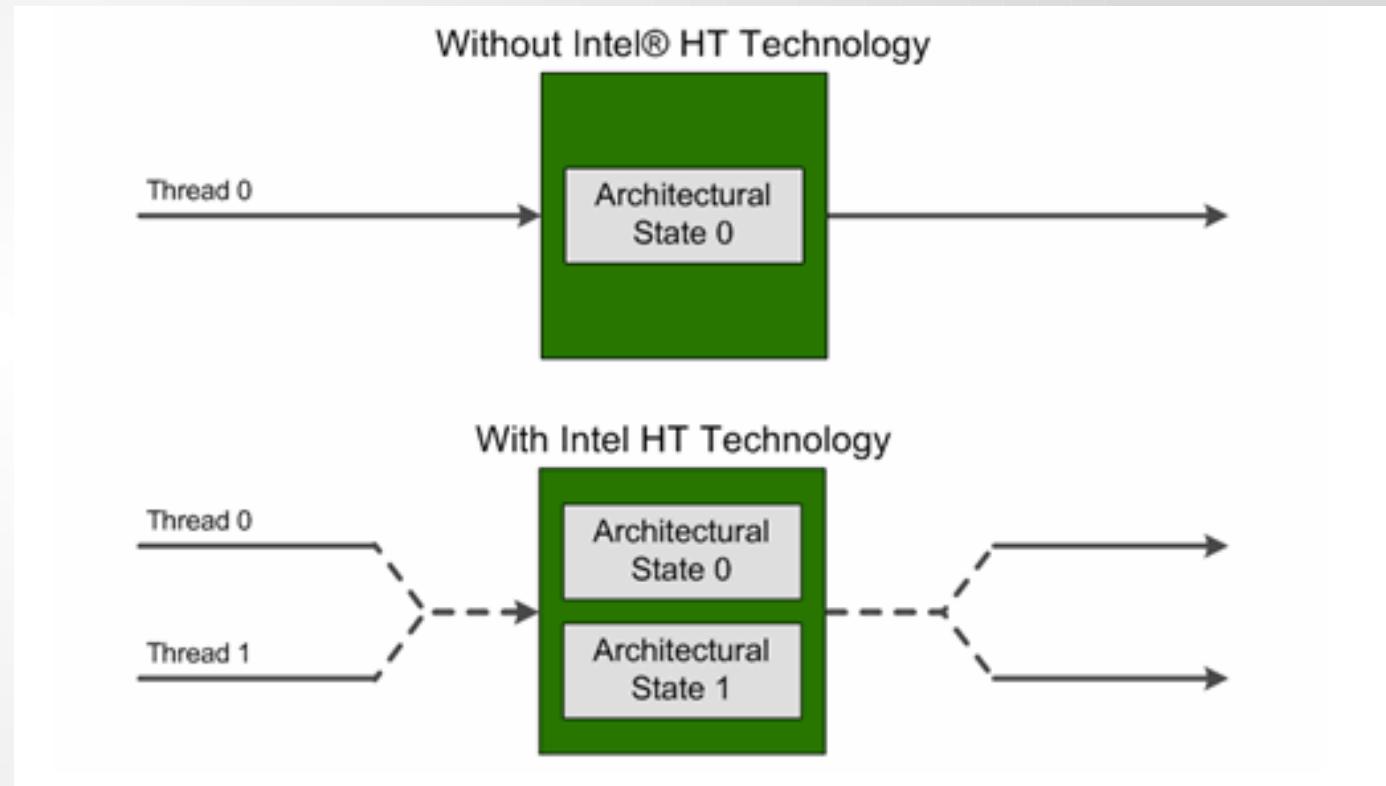
Anexo Procesadores

- Los primeros procesadores estaban conformados por un solo núcleo.
- Esta es la **parte del procesador en la que se realizan todos los cálculos**, es el cerebro que hace que nuestra pc funcione
- Cada uno de los núcleos podía manejar un hilo o thread de datos.

Evolución

- La **tecnología HyperThreading de Intel** que consiste en **duplicar algunos elementos dentro del procesador** como los registros o las memorias caché de primer nivel, esto **permite al núcleo del procesador poder manejar dos tareas a la vez** (2 hilos o threads) y da lugar a la aparición de los **núcleos lógicos**.
- Algo que mejora el rendimiento de forma importante ya que, si un proceso necesita quedar a la espera de una operación o algún dato, otro proceso puede seguir haciendo uso del procesador sin que este se quede parado

HyperThreading



HyperThreading

- Esta tecnología HyperThreading “engaña” al sistema operativo al hacerle creer que existen dos núcleos cuando en realidad solo existe uno, el que existe de verdad es el núcleo físico y el que aparece fruto de HyperThreading es el virtual. El núcleo virtual tiene mucha menos capacidad de procesamiento que el núcleo físico por lo que el rendimiento no es equivalente a tener dos núcleos físicos ni mucho menos, pero proporciona un buen extra

Multicore

- El siguiente paso en la evolución de los procesadores era dar el **salto a la aparición de los procesadores con dos núcleos físicos**, esto fue posible gracias a la miniaturización de todos los elementos que hay dentro del procesador, es decir que se hacen más pequeños y por tanto podemos meter muchos más en el mismo espacio. En esencia un procesador de dos núcleos **es como tener dos procesadores trabajando juntos, pero con una comunicación entre ellos mucho más rápida y eficiente**, lo que hace que el rendimiento sea muy superior a los sistemas con dos sockets y dos procesadores.

Multicore Hyperthreading

- A diferencia del HyperThreading, en los procesadores de dos núcleos cada uno de ellos tiene todos los elementos necesarios para poder realizar todo tipo de tareas por lo que **un procesador de dos núcleos es muy superior en rendimiento a un procesador de un núcleo con HyperThreading**. El siguiente paso fue lograr procesadores de más núcleos, algo posible a una miniaturización de sus componentes cada vez más grande. Hoy en día existen procesadores con hasta 18 núcleos físicos.
- **Es posible combinar el uso de varios núcleos con la tecnología HyperThreading** por lo que podemos lograr procesadores con un número enorme de núcleos lógicos, así un procesador de 18 núcleos físicos con HyperThreading tiene un total de 36 núcleos lógicos (18 núcleos físicos + 18 núcleos virtuales).

SISTEMAS OPERATIVOS

Unidad 6

Sincronización de Procesos

Sincronización de Procesos

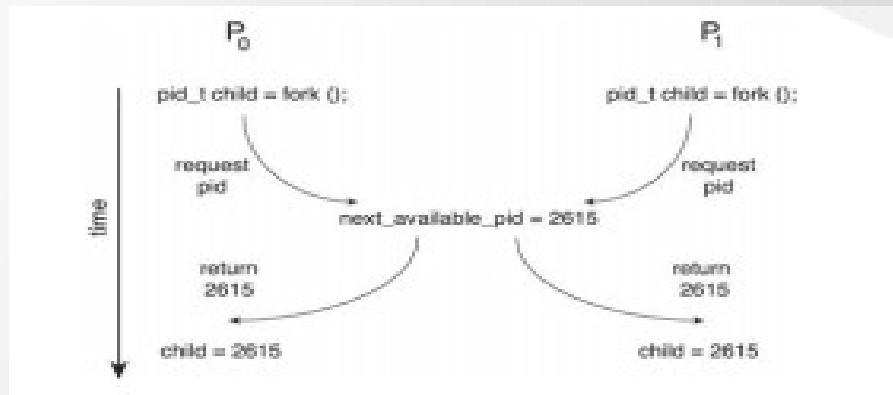
- Fundamentos
- El Problema de la Sección Crítica
- Solución a la sección crítica para 2 procesos (Algoritmo de Peterson) y para n procesos (Algoritmo del Panadero)
- Soluciones por Hardware
- Soluciones por Software
 - Locks
 - Semáforos
 - Monitores
- Problemas Clásicos
- Ejemplos de Sincronización en los Sistemas Operativos

Fundamentos

- El acceso concurrente a datos compartidos puede resultar en inconsistencias.
- Mantener la consistencia de datos requiere mecanismos para asegurar la ejecución ordenada de procesos cooperativos.
- Caso de análisis: problema del buffer limitado. Una solución, donde todos los N buffers son usados, no es simple.

Problema

- Procesos P0 y P1 están creando un proceso hijo utilizando la llamada al sistema fork().
- El kernel tiene la variable next_available_pid la cual representa el próximo identificador disponible para un proceso (pid).



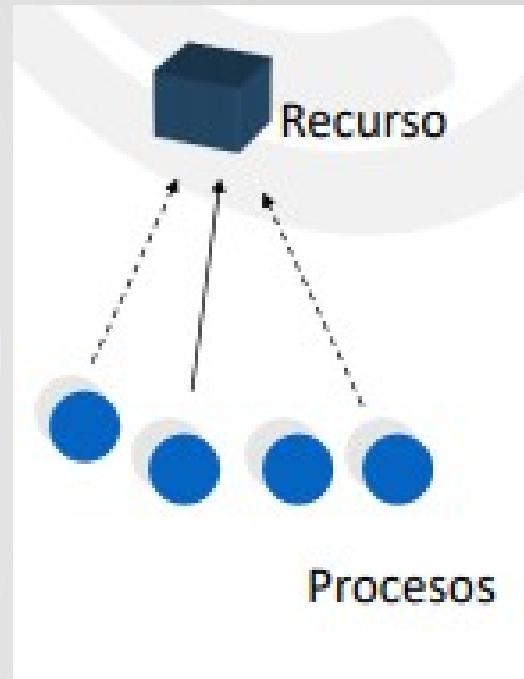
- Podría ocurrir que el mismo pid sea asignado a dos procesos diferentes

Condición de Carrera

- Condición de carrera – Es la situación donde varios procesos acceden y manejan datos compartidos concurrentemente. El valor final de los datos compartidos depende de que proceso termina último.
- Para prevenir las condiciones de carrera, los procesos concurrentes cooperativos deben ser sincronizados.

Problema de la Sección Crítica

- n procesos todos compitiendo para usar datos compartidos
- Cada proceso tiene un segmento de código llamado sección crítica, en la cual los datos compartidos son accedidos
- Problema – asegurar que cuando un proceso está ejecutando en su sección crítica, no se le permite a otro proceso ejecutar en su respectiva sección crítica.



Solucion al Prob. De la Sección Critica

CONDICIONES PARA UN BUEN ALGORITMO

- **Exclusión Mutua:** Si el proceso P_j está ejecutando en su sección crítica, entonces ningún otro proceso puede estar ejecutando en la sección crítica.
- **Progreso:** Si ningún proceso está ejecutando en su sección crítica y existen algunos procesos que desean entrar en la sección crítica, entonces la selección de procesos que desean entrar en la sección crítica no puede ser pospuesta indefinidamente.
- **Espera Limitada:** Debe existir un límite en el número de veces que a otros procesos les está permitido entrar en la sección crítica después que un proceso ha hecho un requerimiento para entrar en la sección crítica y antes que ese requerimiento sea completado.
 - Asuma que cada proceso ejecuta a velocidad distinta de cero.
 - No se asume nada respecto a la velocidad relativa de los n procesos.

Resolución del Problema

- Estructura general del proceso P_i :

repeat

protocolo de entrada

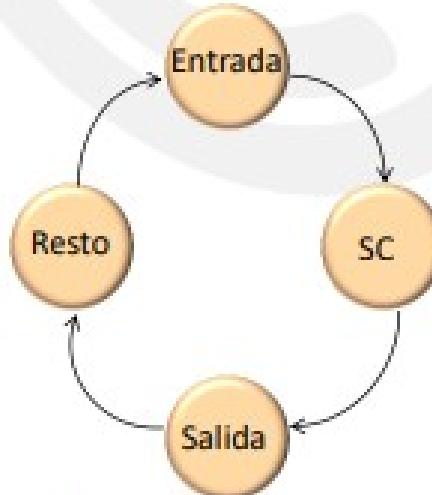
sección crítica (SC)

protocolo de salida

sección resto

until falso

- Los procesos pueden compartir algunas variables comunes para sincronizar sus acciones.



Solución de Peterson para 2 procesos

```
bandera[0] = false
bandera[1] = false
turno      // No es necesario asignar un turno
p0: bandera[0] = true
    turno = 1
    while( bandera[1] && turno == 1 );
        { //no hace nada; espera. }
    // sección crítica

    // fin de la sección crítica
    bandera[0] = false

p1: bandera[1] = true
    turno = 0
    while( bandera[0] && turno == 0 );
        { //no hace nada; espera. }
    // sección crítica

    // fin de la sección crítica
    bandera[1] = false
```

Los procesos p0 y p1 no pueden estar en la sección crítica al mismo tiempo: si p0 está en la sección crítica, entonces bandera[0] = true, y ocurre que bandera[1] = false, con lo que p1 ha terminado la sección crítica, o que la variable compartida turno = 0, con lo que p1 está esperando para entrar a la sección crítica. En ambos casos, p1 no puede estar en la sección crítica.

Algoritmo para N Procesos – Alg. Del Panadero

Sección crítica para n procesos

- Antes de entrar en su sección crítica, el proceso recibe un número. El poseedor del número mas chico entra en su sección crítica.
- Si los procesos P_i y P_j reciben el mismo número, si $i < j$, entonces P_i es servido primero; sino lo es P_j .
- El esquema de numeración siempre genera números en orden incremental de enumeración;
 - p.e., 1,2,3,3,3,4,5...

Algoritmo para N Procesos – Alg. Del Panadero

N es el número de hilos que hay en el sistema.

Para conocer los números de turno se utiliza un vector de enteros. `número[i]` es el turno correspondiente al hilo *i*. Si `número[i] = 0` significa que el hilo *i* no está interesado en entrar en sección crítica.

```
/* Variables globales */
Número: vector [1..N] de enteros = {todos a 0};
Eligiendo: vector [1..N] de booleanos = {todos a falso};

/* Código del hilo i */
Hilo(i) {
    loop {

        /* Calcula el número de turno */
        Eligiendo[i] = cierto;
        Número[i] = 1 + max(Número[1],..., Número[N]);
        Eligiendo[i] = falso;

        /* Compara con todos los hilos */
        for j in 1..N {

            /* Si el hilo j está calculando su número, espera a que termine */
            while ( Eligiendo[j] ) { /* nada */ }

            /* Si el hilo j tiene más prioridad, espera a que ponga su número a cero */
            /* j tiene más prioridad si su número de turno es más bajo que el de i, */
            /* o bien si es el mismo número y además j es menor que i */
            while ( (Número[j] != 0) && ((Número[j], j) < (Número[i], i)) ) { /* nada */ }
        }

        /* Sección crítica */
        ...
        /* Fin de sección crítica */

        Número[i] = 0;

        /* Código restante */
    }
}
```

Sincronizacion de Hardware

- Monoprocesador – pueden deshabilitarse las interrupciones
 - El código que esta corriendo debe ejecutar sin apropiación
 - Generalmente es demasiado ineficiente en sistemas multiprocesador
 - Los SOs que usan esto no son ampliamente escalables
- Las computadoras modernas proveen instrucciones especiales que se ejecutan atómicamente
 - Atómico = no-interrumpible
 - Sea por verificación de una palabra de memoria y su inicialización o por intercambio de dos palabras de memoria
- Instrucciones por hardware:
 - test-and-set: verifica y modifica el contenido de una palabra atómicamente.
 - Swap, intercambia el valor de dos variables en memoria en una sola instrucción

Test and Set

```
booleano TS(int i)
{
    if (i==0)
    {
        i=1;
        return cierto;
    }
    else
    {
        return falso;
    }
}
```

SWAP

```
void intercambiar (int registro, int memoria)
{
    int temp;
    temp = memoria;
    memoria = registro;
    registro = temp;
}
```

Exclusión Mutua con Test and Set

- Dato compartido : **var lock: boolean (inicialmente false)**
- Proceso P_i

```
repeat
    while !Test-and-Set (lock) do no-op;
        sección crítica
        lock := false;
        sección resto
    until false;
```

Exclusión Mutua con Swap

- La variable booleana compartida es inicializada en FALSE; Cada proceso tiene una variable local booleana key
- Solución:

```
while (true) {  
    key = TRUE;  
    while ( key == TRUE)  
        Swap (&lock, &key );  
        sección crítica  
    lock = FALSE;  
    sección restante  
}
```

Exclusión Mutua usando Locks

```
do {  
    acquire lock  
    sección crítica  
    release lock  
    sección resto  
} while (TRUE);
```

Semáforos

- Es una herramienta de sincronización.
- Semáforo S – variable entera
- Dos operaciones standard modifican S: wait() y signal()
Originalmente llamadas P() y V()
- Puede ser accedido solo por dos operaciones indivisibles
(deben ser atómicas):

wait (S):

```
while S <= 0 do no-op;
```

```
S := S - 1;
```

signal (S):

```
S := S + 1;
```

Semaforo – Herramienta General de Sincronización

- Semáforo de Cuenta (Contador) – el valor entero puede tener un rango sobre un dominio sin restricciones.
- Semáforo Binario – el valor entero puede tener un rango solo entre 0 y 1; puede ser más simple de implementar
- Puede utilizarse para alcanzar exclusión mutua
 - Semáforo S; // inicializado en 1
 - wait (S);
 - Sección Crítica
 - signal (S);
- Semaforo Binario o Mutex Locks – se utilizan específicamente para la exclusión mutua

Semaforo – Herramienta General de Sincronización

- Para sincronización
 - Ejecute B en P_j solo después que A ejecute en P_i
 - Use el semáforo flag inicializado a 0
 - Código:



Implementación del Semáforo

- Debe garantizar que dos procesos no puedan ejecutar wait () y signal () sobre el mismo semáforo al mismo tiempo
- Entonces, la implementación se convierte en el problema de la sección crítica donde el código del wait y el signal son la sección crítica.
 - Podemos tener ahora espera ocupada en la implementación de la sección crítica porque:
 - El código de implementación es corto
 - Poca espera ocupada si la sección crítica es raramente invocada
- Note que las aplicaciones pueden pasar y gastar mucho tiempo en secciones críticas, entonces no es una buena solución utilizar semáforos implementados con espera ocupada.

Implementación de Semáforos sin espera ocupada

- Con cada semáforo hay asociada una cola de espera. Cada entrada en dicha cola tiene dos datos:
 - valor (de tipo entero)
 - puntero al próximo registro en la lista
- Dos operaciones:
 - block – ubica el proceso invocando la operación en la apropiada cola de espera.
 - wakeup – remueve uno de los procesos en la cola de espera y lo ubica en la cola de listos.

Implementación de Semáforos sin espera ocupada

- Las operaciones del semáforo se definen como

```
wait(S): S.value := S.value - 1;  
          if S.value < 0  
              then begin  
                  agregue este proceso a S.L;  
                  block;  
              end;  
signal(S): S.value := S.value + 1;  
          if S.value ≤ 0  
              then begin  
                  remueva un proceso P de S.L;  
                  wakeup(P);  
              end;
```

Interbloqueo e Inanición

- INTERBLOQUEO – dos o más procesos están esperando indefinidamente por un evento que puede ser causado por solo uno de los procesos que esperan.
- Sean S y Q dos semáforos inicializados a 1

P_0	P_1
<i>wait(S);</i>	<i>wait(Q);</i>
<i>wait(Q);</i>	<i>wait(S);</i>
\vdots	\vdots
<i>signal(S);</i>	<i>signal(Q);</i>
<i>signal(Q);</i>	<i>signal(S);</i>

- INANICIÓN – bloqueo indefinido. Un proceso no puede ser removido nunca de la cola del semáforo en el que fue suspendido.
- INVERSIÓN DE PRIORIDADES

Problemas Clásicos de Sincronización

- Problema del Buffer Limitado
- Problema de Lectores y Escritores
- Problema de los Filósofos Cenando

Problema del Buffer Limitado

■ Datos compartidos

```
type item = ...
var buffer = ...
    full, empty : semáforo de conteo;
    mutex: semáforo binario;
    nextp, nextc: item;
```

INICIALIZACIÓN SEMÁFOROS

```
full :=0; empty :=n; mutex :=1;
```

Problema del Buffer Limitado

- Proceso Productor
 - repeat

...
produce un ítem en *nextp*

...
wait(empty);
wait(mutex);

...
agregue *nextp* al buffer

...
signal(mutex);
signal(full);

until false;

- Proceso Consumidor
 - repeat

...
wait(full)
wait(mutex);

...
remueve un ítem de *buffer* a *nextc*

...
signal(mutex);
signal(empty);

...
consume el ítem en *nextc*

until false;

Semáforos

- Incorrecto uso de las operaciones sobre semáforos:
 - signal (mutex) wait (mutex)
 - wait (mutex) ... wait (mutex)
 - Omitir el wait (mutex) o el signal (mutex) (o ambos)
- Extensión
 - Incorporación de la operación wait-try sobre un semáforo.

Monitores

- Es un constructor de sincronización de alto nivel que permite compartir en forma segura un tipo de dato abstracto entre procesos concurrentes.

```
monitor monitor-nombre
```

```
{
```

```
    // declaración de variables compartidas
```

```
    procedure P1 (...) { ... }
```

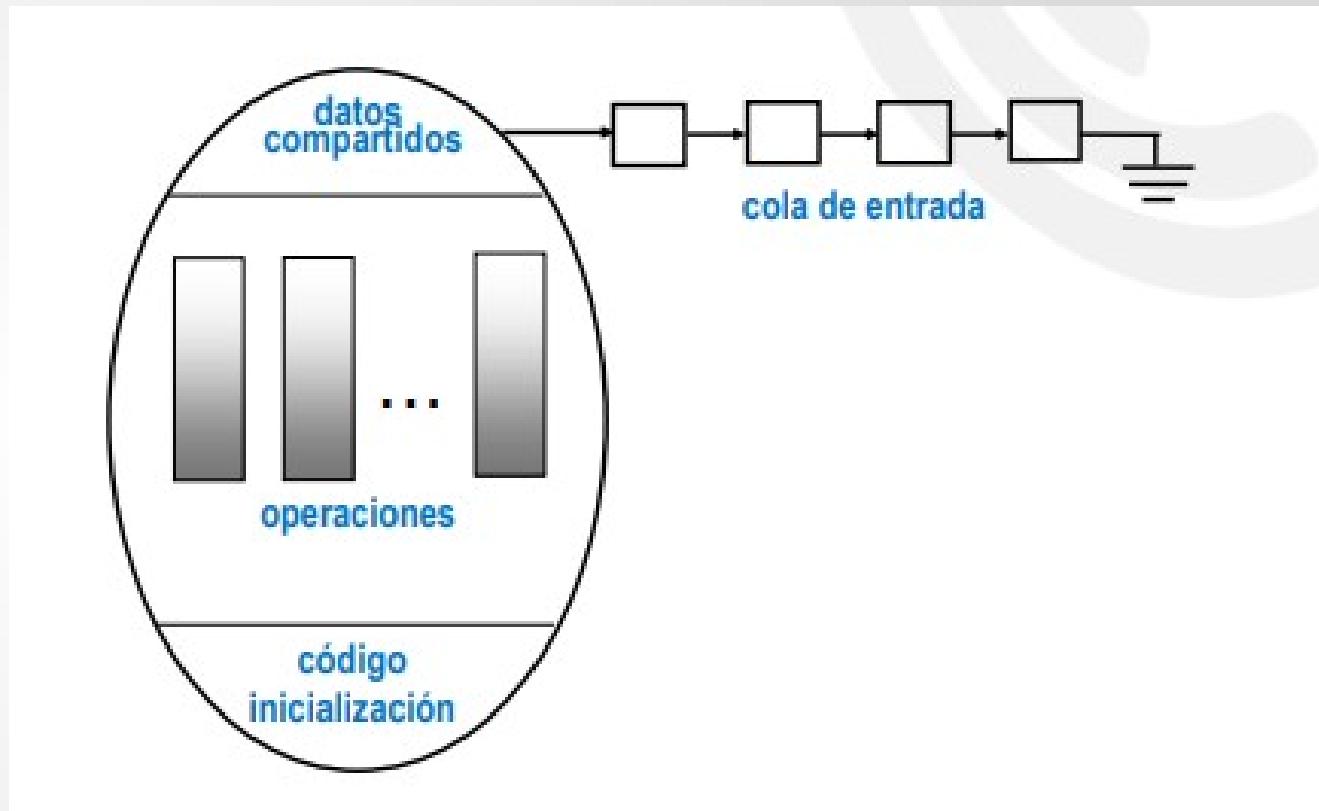
```
    procedure Pn (...) {.....}
```

```
    código de inicialización(...) { ... }
```

```
}
```

```
}
```

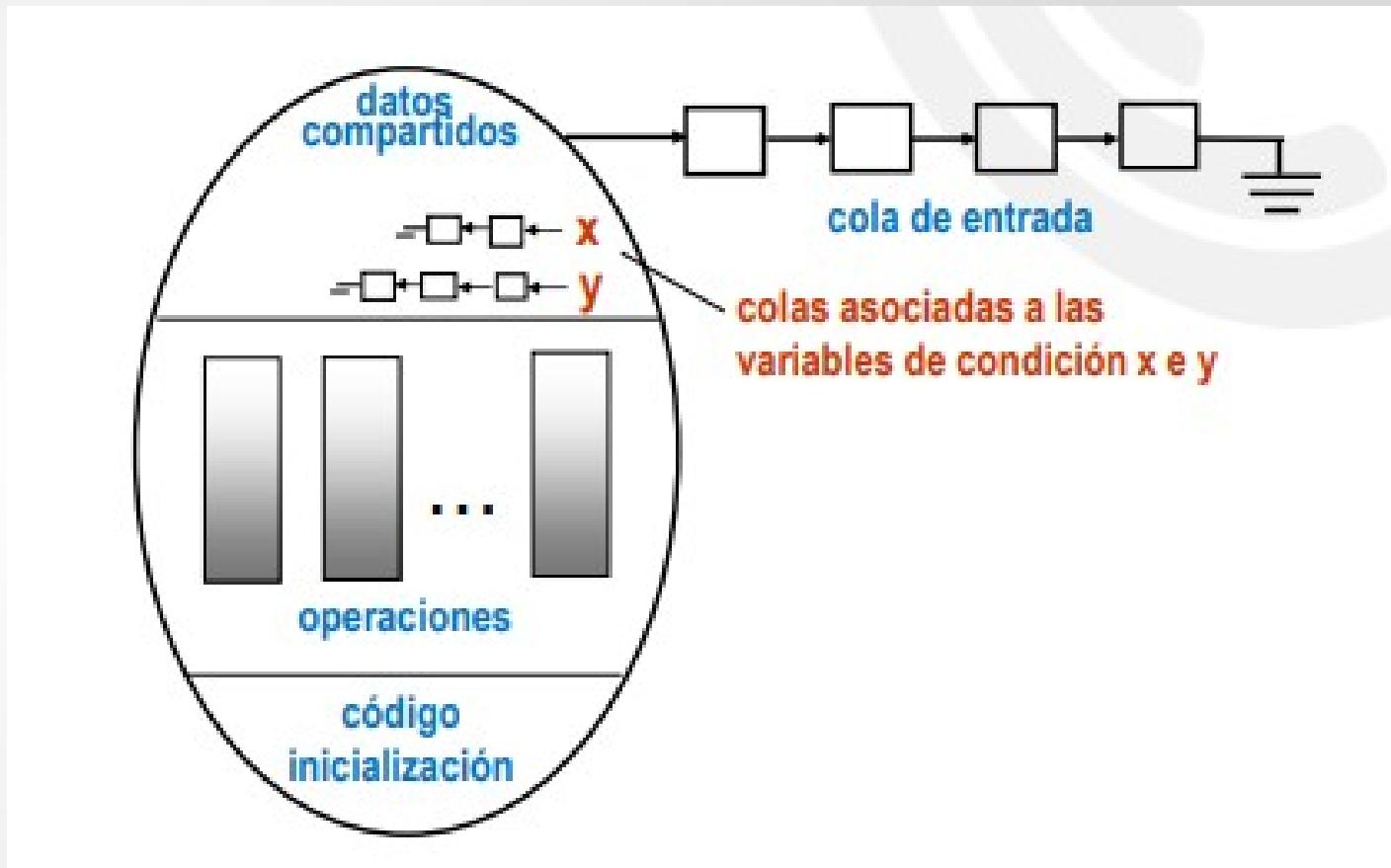
Vista Esquemática de un monitor



Variables de Condición

- Condición x, y;
- Dos operaciones sobre una variable de condición
 - x.wait () – el proceso que invoca esa operación es suspendido
 - x.signal () – reinicia uno de los procesos (si hay alguno) que invocó x.wait ()

Monitor con Variables de Condición



Problema del Buffer Limitado

```
monitor ProdCons
```

```
    condition full, empty;  
    integer contador;
```

```
procedure insertar(item: integer)  
begin  
    if contador == N then full.wait();  
    Insertar_item(item);  
    contador := contador + 1;  
    if contador == 1 then empty.signal()  
end;
```

```
    contador := 0;  
end monitor;
```

```
function remover: integer  
begin  
    if contador == 0 then empty.wait();  
    Remover = remover_item;  
    contador := contador -1;  
    if contador == N-1 then full.signal()  
end;
```

Problema del Buffer Limitado

```
procedure productor
begin
  while true do
    begin
      item = produce_item;
      ProdCons.insertar(item);
    end
  end;
```

```
procedure consumidor
begin
  while true do
    begin
      item = ProdCons.remover;
      Consume_item(item);
    end
  end;
```

Sincronización de Windows

En el kernel

- Usa máscaras de interrupción para proteger el acceso a recursos globales en sistemas mono procesador
- Usa **spinlocks** en sistemas multiprocesador

Sincronización hilos

- También provee **dispatcher objects** los cuales actúan como mutex locks, semáforos, eventos y timers.
- Los *Dispatcher objects* pueden proveer **eventos**
 - Un evento actúa como una variable de condición

Los estados del **MUTEX**
DISPATCHER OBJECT



Spinlock

- Un spinlock es un bloqueo que hace que un hilo que intenta adquirirlo simplemente espere en un bucle ("girar" en inglés "spin") mientras comprueba repetidamente si el bloqueo(lock) está disponible. Como el hilo permanece activo pero no está realizando una tarea útil, el uso de dicho bloqueo es una especie de espera ocupada . Una vez adquiridos, los spinlocks generalmente se mantendrán hasta que se liberen explícitamente, aunque en algunas implementaciones se pueden liberar automáticamente si el hilo que está esperando se (lo que contiene el bloqueo) bloquea, o "se va a dormir"

Sincronización en Linux

■ Linux:

- Antes de la versión 2.6 no era un kernel totalmente apropiativo.
- Deshabilita las interrupciones para implementar secciones críticas cortas

■ Linux provee:

- mutex-locks
- semáforos
- spin locks

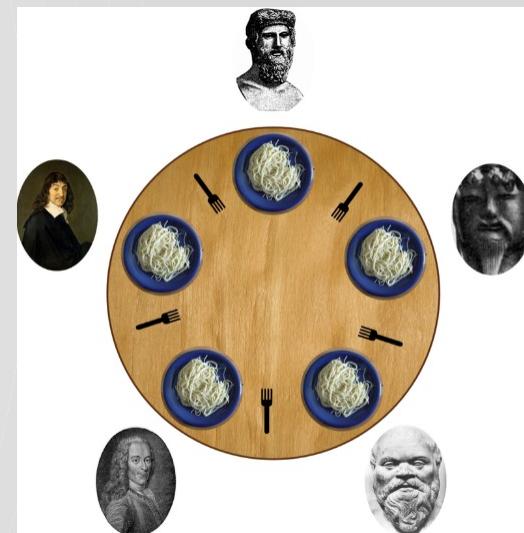
Un procesador	Múltiples procesadores
Deshabilitar apropiación kernel	Acquire spin lock
Habilitar apropiación kernel	Release spin lock

Anexo – Filosofos Cenando

- Cinco filósofos se sientan alrededor de una mesa y pasan su vida cenando y pensando. Cada filósofo tiene un plato de fideos y un tenedor a la izquierda de su plato. Para comer los fideos son necesarios dos tenedores y cada filósofo sólo puede tomar los que están a su izquierda y derecha. Si cualquier filósofo toma un tenedor y el otro está ocupado, se quedará esperando, con el tenedor en la mano, hasta que pueda tomar el otro tenedor, para luego empezar a comer.
- Si dos filósofos adyacentes intentan tomar el mismo tenedor a una vez, se produce una condición de carrera: ambos compiten por tomar el mismo tenedor, y uno de ellos se queda sin comer.

Anexo – Filosofos Cenando

- Si todos los filósofos toman el tenedor que está a su derecha al mismo tiempo, entonces todos se quedarán esperando eternamente, porque alguien debe liberar el tenedor que les falta. Nadie lo hará porque todos se encuentran en la misma situación (esperando que alguno deje sus tenedores). Entonces los filósofos se morirán de hambre. Este bloqueo mutuo se denomina interbloqueo o deadlock.
- El problema consiste en encontrar un algoritmo que permita que los filósofos nunca se mueran de hambre.



Anexo – Filosofos Cenando - Soluciones

- **Por turno cíclico**

Se empieza por un filósofo, que si quiere puede comer y después pasa su turno al de la derecha. Cada filósofo sólo puede comer en su turno. Problema: si el número de filósofos es muy alto, uno puede morir de hambre antes de su turno.

- **Varios turnos**

Se establecen varios turnos. Para hacerlo más claro supongamos que cada filósofo que puede comer (es su turno) tiene una ficha que después pasa a la derecha. Si por ejemplo hay 7 comensales podemos poner 3 fichas en posiciones alternas (entre dos de las fichas quedarían dos filósofos).

Se establecen turnos de tiempo fijo. Por ejemplo cada 5 minutos se pasan las fichas (y los turnos) a la derecha.

Con base al tiempo que suelen tardar los filósofos en comer y en volver a tener hambre, el tiempo de turno establecido puede hacer que sea peor solución que la anterior. Si el tiempo de turno se aproxima al tiempo medio que tarda un filósofo en comer esta variante da muy buenos resultados. Si además el tiempo medio de comer es similar al tiempo medio en volver a tener hambre la solución se aproxima al óptimo.

Anexo – Filosofos Cenando - Soluciones

- **Colas de tenedores**

Cuando un filósofo quiere comer se pone en la cola de los dos tenedores que necesita. Cuando un tenedor está libre lo toma. Cuando toma los dos tenedores, come y deja libre los tenedores.

Visto desde el otro lado, cada tenedor sólo puede tener dos filósofos en cola, siempre los mismos.

Esto crea el problema comentado de que si todos quieren comer a la vez y todos empiezan tomando el tenedor de su derecha se bloquea el sistema (*deadlock*).

- **Resolución de conflictos en colas de tenedores**

Cada vez que un filósofo tiene un tenedor espera un tiempo *aleatorio* para conseguir el segundo tenedor. Si en ese tiempo no queda libre el segundo tenedor, suelta el que tiene y vuelve a ponerse *en cola* para sus dos tenedores.

Si un filósofo A suelta un tenedor (porque ha comido o porque ha esperado demasiado tiempo con el tenedor en la mano) pero todavía desea comer, vuelve a ponerse *en cola* para ese tenedor. Si el filósofo adyacente B está ya en esa cola de tenedor (tiene hambre) lo toma y si no vuelve a cogerlo A.

Es importante que el tiempo de espera sea aleatorio o se mantendrá el bloqueo del sistema.

Anexo – Filosofos Cenando - Soluciones

- **El portero del comedor**

Se indica a los filósofos que abandonen la mesa cuando no tengan hambre y que no regresen a ella hasta que vuelvan a estar hambrientos (cada filósofo siempre se sienta en la misma silla). La misión del portero es controlar el número de filósofos en la sala, limitando su número a $n-1$, pues si hay $n-1$ comensales seguro que al menos uno puede comer con los dos tenedores.

Anexo – Problema Lectores y Escritores

- Otro problema famoso es el de los lectores y escritores (Courtois *et al.*, 1971), que modela el acceso a una base de datos. Supóngase una base de datos, con muchos procesos que compiten por leer y escribir. Se puede permitir que varios procesos lean de la base de datos al mismo tiempo, pero si uno de los procesos está escribiendo (es decir, modificando) la base de datos, ninguno de los demás debería tener acceso a ésta, ni siquiera los lectores. La pregunta es ¿ cómo programaría los lectores y escritores ?

SISTEMAS OPERATIVOS

Unidad 7

Administración de Memoria

Administración de Memoria

- Base
- Alocación Contigua
- Segmentación
- Paginado
- Estructura de la Tabla de Páginas
- Intercambio (Swapping)

Base

- El programa debe ser traído del disco a la memoria y ubicado dentro de un proceso para ser corrido.
- La memoria principal y los registros son las únicas unidades de almacenaje accedidas directamente por la CPU.
- El acceso a los registros se hace en un pulso de reloj de CPU o menos.
- La memoria principal puede tomar muchos ciclos.
- El Caché se ubica entre la memoria principal y los registros de CPU.
- Para asegurar una correcta operación se requiere cierta protección de memoria.

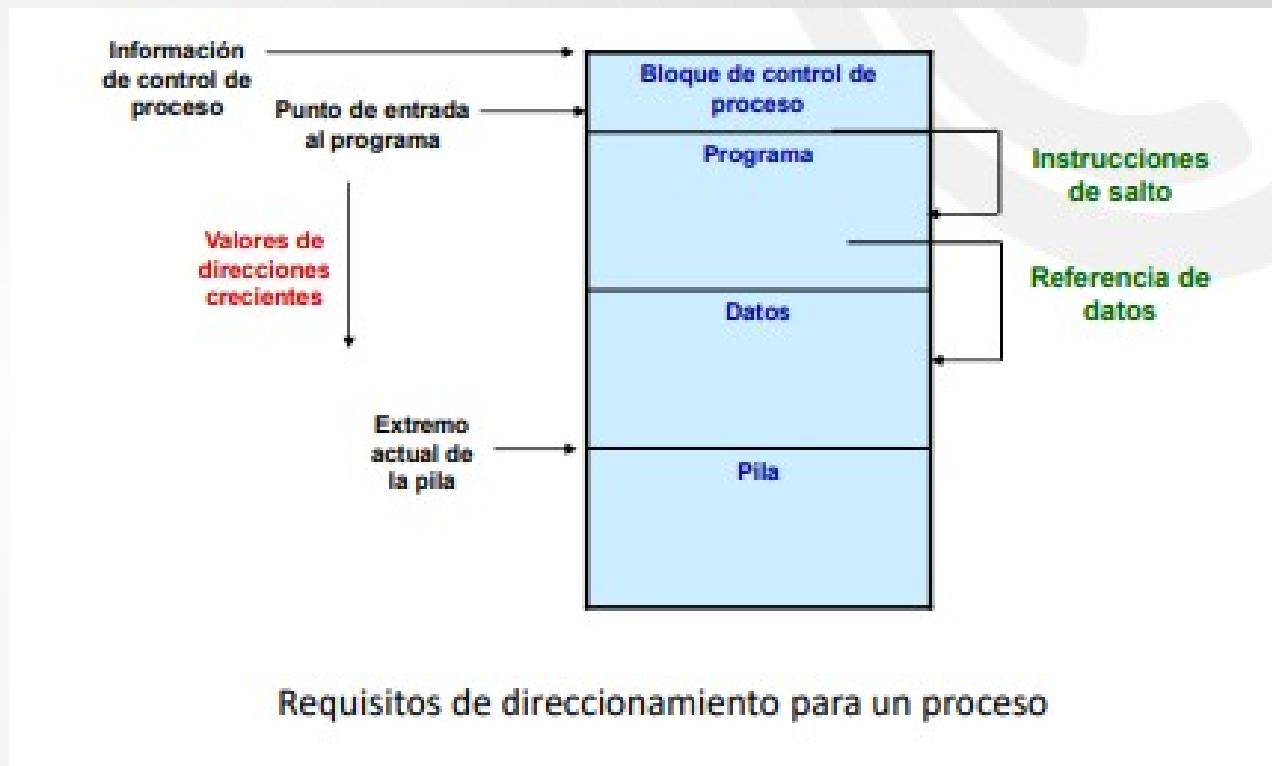
Gestión de la Memoria

- Subdividir la memoria para acomodar múltiples procesos
- Es necesario asignar la memoria para asegurar una cantidad razonable de procesos listos que consuman el tiempo de procesador disponible

Requisitos de la Gestión de la memoria

- Reubicación
 - El programador no sabe en qué parte de la memoria principal se situará el programa cuando se ejecute
 - Mientras el programa se está ejecutando, éste puede llevarse al disco y traerse de nuevo a la memoria principal en un área diferente (reubicado)
 - Deben traducirse las referencias de memoria encontradas en el código del programa en direcciones de memoria físicas

Requisitos de la Gestión de la memoria



Requisitos de la Gestión de la memoria

- Protección
 - Los procesos no deberían ser capaces de referenciar sin permiso posiciones de memoria principal de otro proceso
 - Es imposible comprobar las direcciones absolutas en el tiempo de compilación, deben comprobarse en el tiempo de ejecución
 - Es el procesador (hardware), en lugar del sistema operativo, el que debe satisfacer el requisito de protección de memoria. El sistema operativo no puede anticipar todas las referencias de memoria que un programa hará

Requisitos de la Gestión de la memoria

- Compartición
 - Permite a varios procesos acceder a la misma porción de memoria principal
 - Es mejor permitir que cada proceso pueda acceder a la misma copia del programa en lugar de tener su propia copia separada

Requisitos de la Gestión de la memoria

- Organización lógica
 - Los programas están escritos en módulos
 - Los módulos se pueden escribir y compilar independientemente
 - Se pueden proporcionar diferentes grados de protección a los módulos (sólo lectura, sólo ejecución)
 - Se pueden compartir módulos entre los procesos

Mapeo de Instrucciones y datos a direcciones de memoria

El mapeo de instrucciones y datos a direcciones de memoria puede ocurrir en 3 etapas

- Tiempo de Compilación: Si la locación de memoria es conocida a priori, puede ser generado código absoluto; debe recompilar si la dirección de comienzo cambia.
- Tiempo de Carga: Debe generar código reubicable si la locación de memoria no es conocida en tiempo de compilación.
- Tiempo de Ejecución: El mapeo es retrasado hasta el momento de corrida, el proceso puede ser movido durante su ejecución de un segmento de memoria a otro. Necesita soporte de hardware para el mapeo de las direcciones (p.e., registros base - límite).

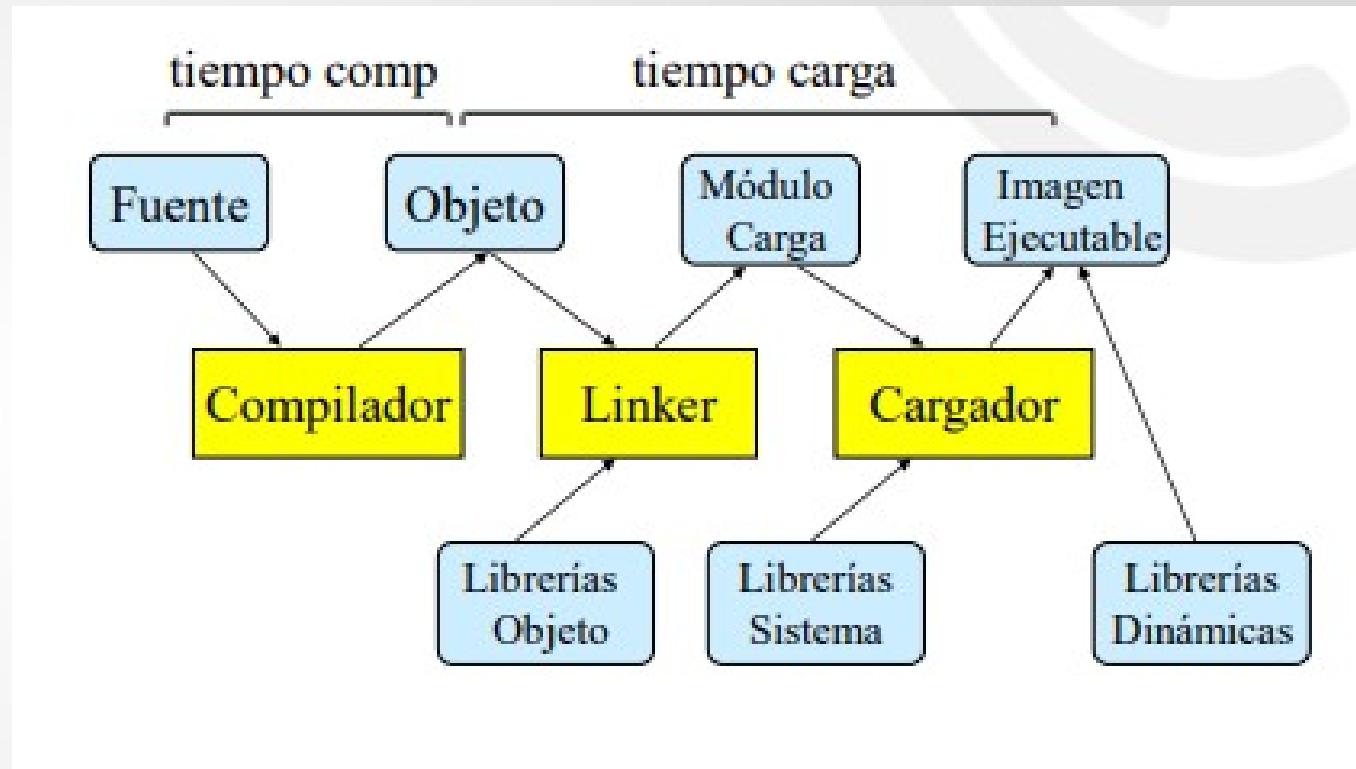
Carga Dinámica

- Las rutinas no son cargadas hasta que son llamadas.
- Mejor utilización del espacio de memoria; las rutinas no usadas no son cargadas.
- Es útil cuando hay que manejar grandes cantidades de código para casos poco frecuentes.
- No requiere un soporte especial del sistema operativo

Enlace Dinámico

- El enlace es pospuesto hasta el tiempo de ejecución.
- Se usan pequeños pedazos de código, stub, para localizar las rutinas apropiadas de la librería residente en memoria.
- El stub se reemplaza a sí mismo con la dirección de la rutina y la ejecuta.
- El sistema operativo necesita verificar si la rutina está en las direcciones de memoria del proceso.

Mapeo de Direcciones



Espacio de Direcciones Lógico VS Físico

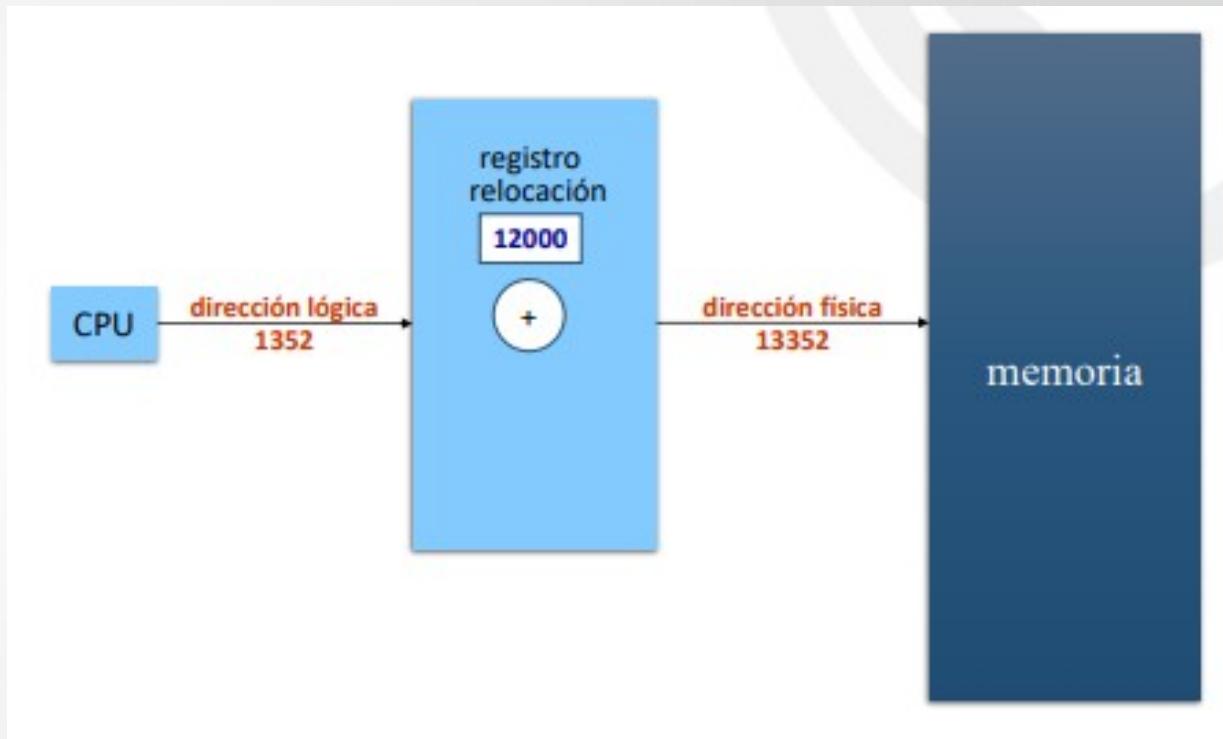
- El concepto de espacio de direcciones lógico que está limitado a un espacio de direcciones físicas separado es central a la administración de la memoria.
 - Dirección Lógicas – generadas por la CPU; también llamadas direcciones virtuales.
 - Dirección Física – dirección vista por la unidad de memoria.
- Las direcciones lógicas y físicas son las mismas en tiempo de compilación y en los esquemas de mapeo de direcciones en tiempo de carga; las direcciones lógicas (virtuales) y físicas difieren en el esquema de mapeo de direcciones en tiempo de ejecución.

Unidad de Administración de Memoria (MMU)

- Dispositivo hardware que mapea las direcciones virtuales a las físicas.
- En el esquema MMU, el valor en el registro de locación es agregado a cada dirección generada por el proceso del usuario en el momento que es presentada a la memoria.
- Los programas de usuario ven direcciones lógicas, nunca ven direcciones físicas reales.



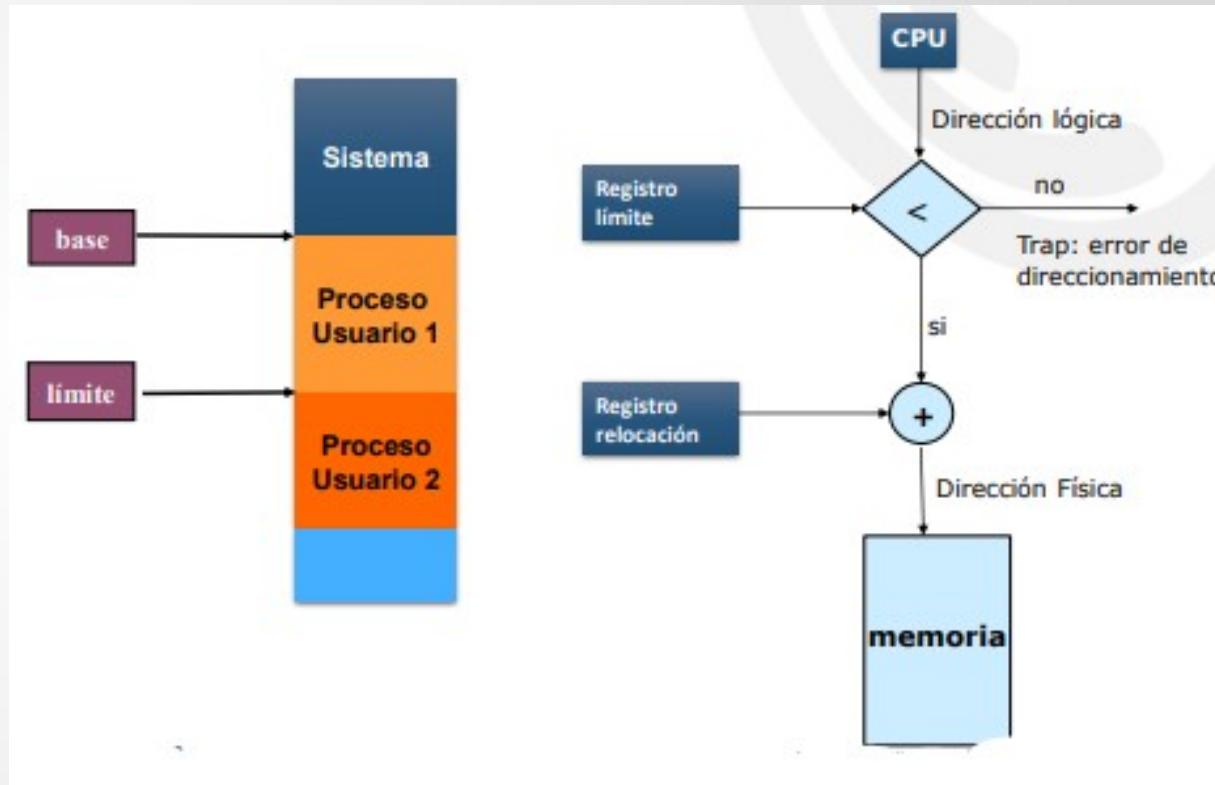
Unidad de Administracion de Memoria (MMU)



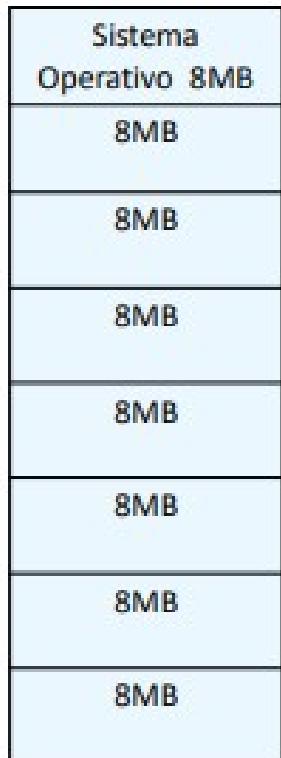
Alocación Contigua

- La memoria principal se divide, usualmente en dos particiones:
 - Parte residente del sistema operativo, generalmente en memoria baja con el vector de interrupciones.
 - Los procesos de usuario se mantienen en la parte alta de la memoria.
- Alocación en partición simple
 - Se usa un esquema de registro de relocación para proteger los procesos de usuario uno de otro.
 - El registro de relocación contiene el valor de la dirección física más pequeña; el registro límite contiene el rango de las direcciones lógicas – cada dirección lógica debe ser menor que el registro límite.

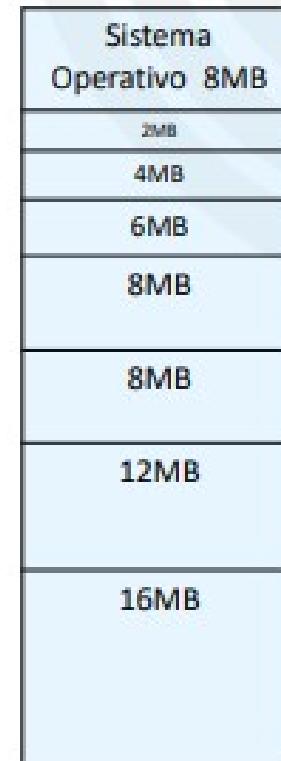
Alocación Contigua



Alocación Contigua – Particionamiento Fijo

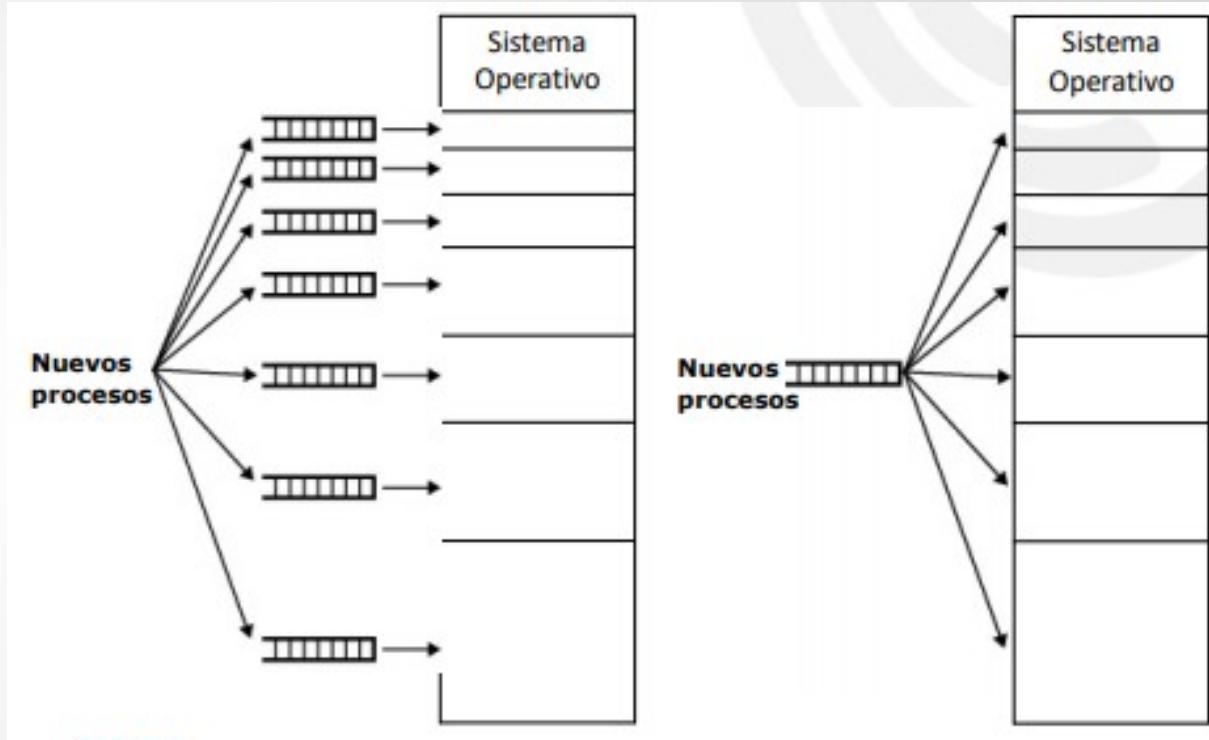


Particiones de Igual Tam.



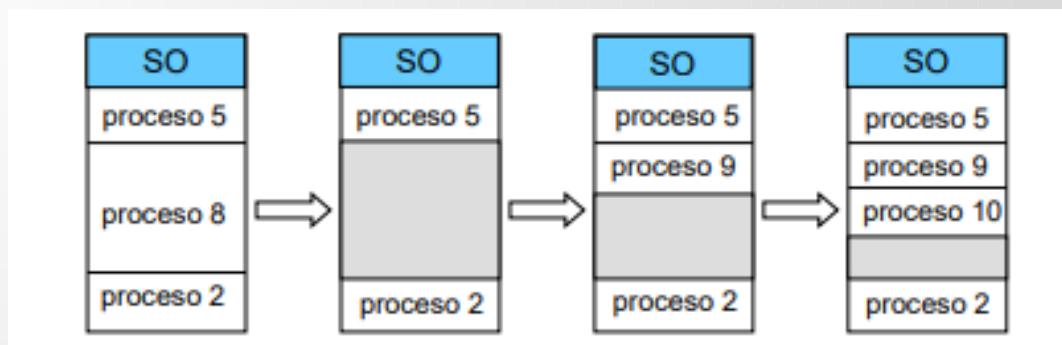
Particiones de Diferentes Tam.

Alocación Contigua – Particionamiento Fijo



Alocación Contigua Dinámica

- Alocación en múltiple partición
 - Agujero – bloque de memoria disponible; hay agujeros de variados tamaños esparcidos por la memoria.
 - Cuando un proceso llega, es alocado en memoria en un agujero suficientemente grande para alojarlo.
 - El SO mantiene información sobre: a) particiones alojadas b) particiones libre (agujero)



Problema de Alocación Dinámica

Como satisfacer un requerimiento de tamaño n de una lista de agujeros libres

- Primer lugar: Aloca en el primer agujero lo suficientemente grande.
- Próximo lugar: Aloca en el próximo lugar luego de haber usado el primero.
- Mejor lugar: Aloca en el agujero más chico que lo puede alojar; debe buscar en la lista completa, salvo que sea ordenada por tamaño. Produce el agujero restante más chico.
- Peor lugar: Aloca el agujero más grande; debe buscar en la lista completa. Produce el agujero restante más grande

El primer lugar y el mejor lugar son mejores que el peor lugar en términos de velocidad y utilización del almacenamiento.

Buddy System (Alocación)

Es un sistema de alocación de memoria utilizado por algunos sistemas operativos.

- El espacio completo disponible es tratado como un bloque de tamaño 2^U
- Si un requerimiento de tamaño s tal que $2^{U-1} < s \leq 2^U$, es alocado el bloque completo.
 - De otra manera el bloque es dividido en dos “compañeros” iguales.
 - El proceso continua hasta que es generado el bloque más pequeño mayor o igual que s .

Buddy System (Alocación)

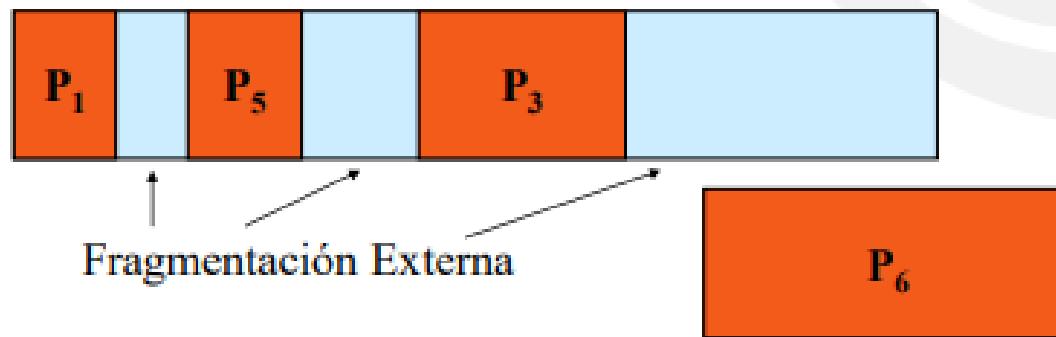
1 M				
Solicitar 100K	A = 128 K	128 K	256 K	512 K
Solicitar 240K	A = 128 K	128 K	B = 256 K	512 K
Solicitar 64K	A = 128 K	C = 64 K	B = 256 K	512 K
Solicitar 256K	A = 128 K	C = 64 K	B = 256 K	D = 256 K
Liberar B	A = 128 K	C = 64 K	256 K	D = 256 K
Liberar A	128 K	C = 64 K	256 K	D = 256 K
Solicitar 75K	E = 128 K	C = 64 K	256 K	D = 256 K
Liberar C	E = 128 K	128 K	256 K	D = 256 K
Liberar E	512 K		D = 256 K	256 K
Liberar D	1 M			

Ejemplo de sistema Buddy

Fragmentación

- **Fragmentación Externa** – existe espacio de memoria para satisfacer una demanda, pero no es contiguo.
- **Fragmentación Interna** – la memoria ocupada puede ser ligeramente más grande que la demandada; esta diferencia de tamaño es memoria interna de la partición, pero no es usada.
- Se reduce la fragmentación externa por compactación
 - Mueva el contenido de memoria de modo de dejar toda la memoria libre en un solo bloque.
 - La compactación es posible solo si la relocación es dinámica, y es hecha en tiempo de ejecución.
 - Problema de E/S
 - Dejar el job en memoria mientras está involucrado en una E/S.
 - Hacer E/S solo en los buffers del SO.

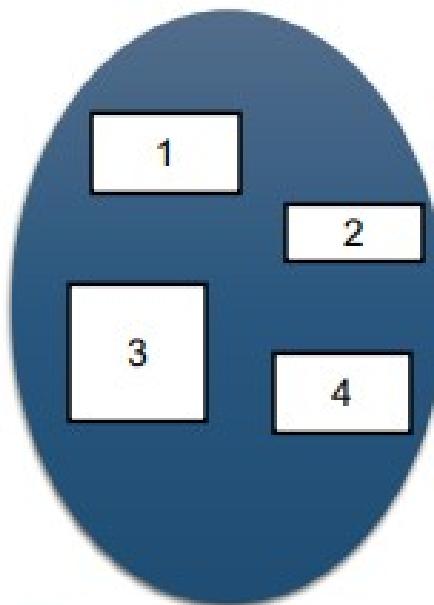
Fragmentación



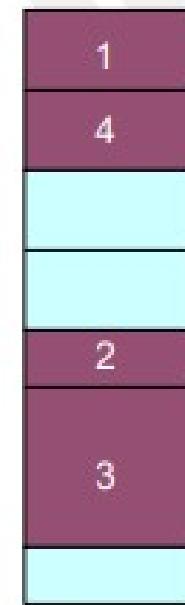
Segmentación

- Esquema de administración de memoria que soporta la visión de usuario de la memoria.
- Un programa es una colección de segmentos. Un segmento es una unidad lógica como:
 - programa principal,
 - procedimiento,
 - función,
 - variables locales,
 - variables globales,
 - bloque común,
 - stack,
 - tabla de símbolos, arreglos

Visión Lógica de la Segmentación



Espacio de usuario



Espacio de memoria física

Arquitectura de Segmentación

- Dirección lógica consistente de dupla:
 < número-segmento, offset>,
- **Tabla de Segmentos** – mapean direcciones físicas en dos dimensiones; cada entrada en la tabla tiene:
 - **base** – contiene la dirección física inicial donde el segmento reside en memoria.
 - **límite** – especifica la longitud del segmento.
- **Registro base de la tabla de segmentos (STBR)** apunta a la locación de la tabla de segmentos en memoria.
- **Registro de longitud de la tabla de segmentos (STLR)** indica el número de segmentos usados por el programa; el número de segmentos es legal si $s < STLR$.

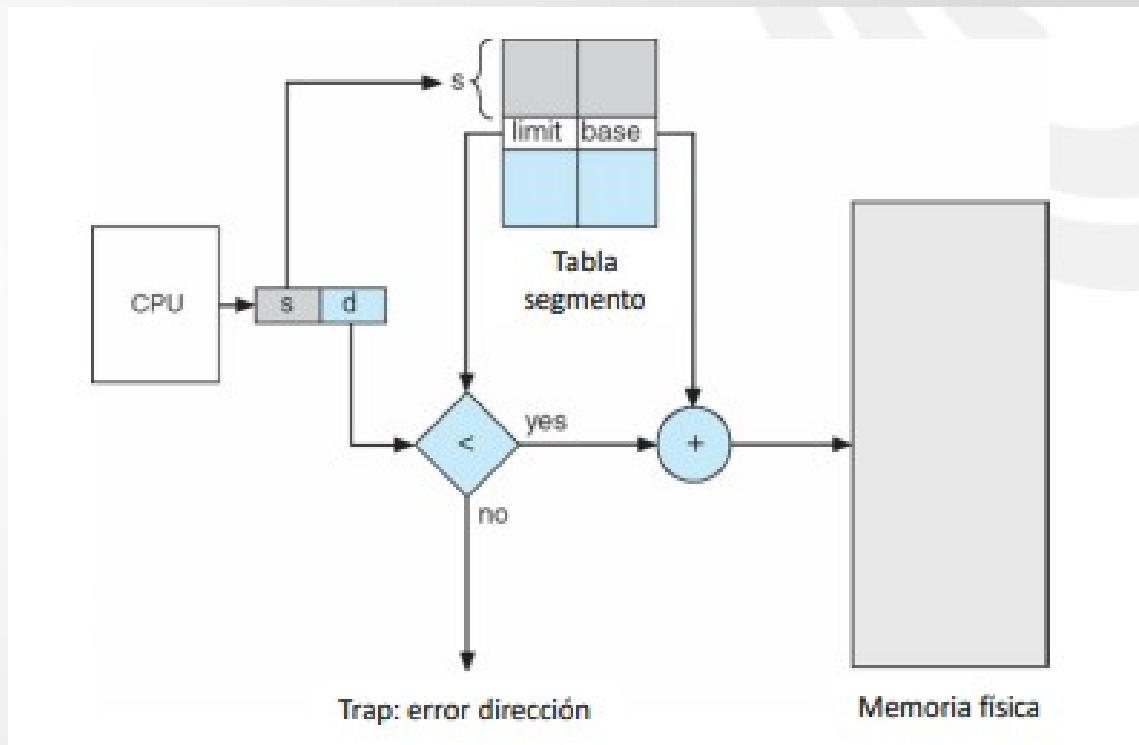
Arquitectura de Segmentación

- Relocación.
 - dinámica
 - por tabla de segmentos
- Compartir
 - segmentos compartidos
 - Igual número de segmento
- Alocación.
 - Primer lugar/mejor lugar
 - fragmentación externa

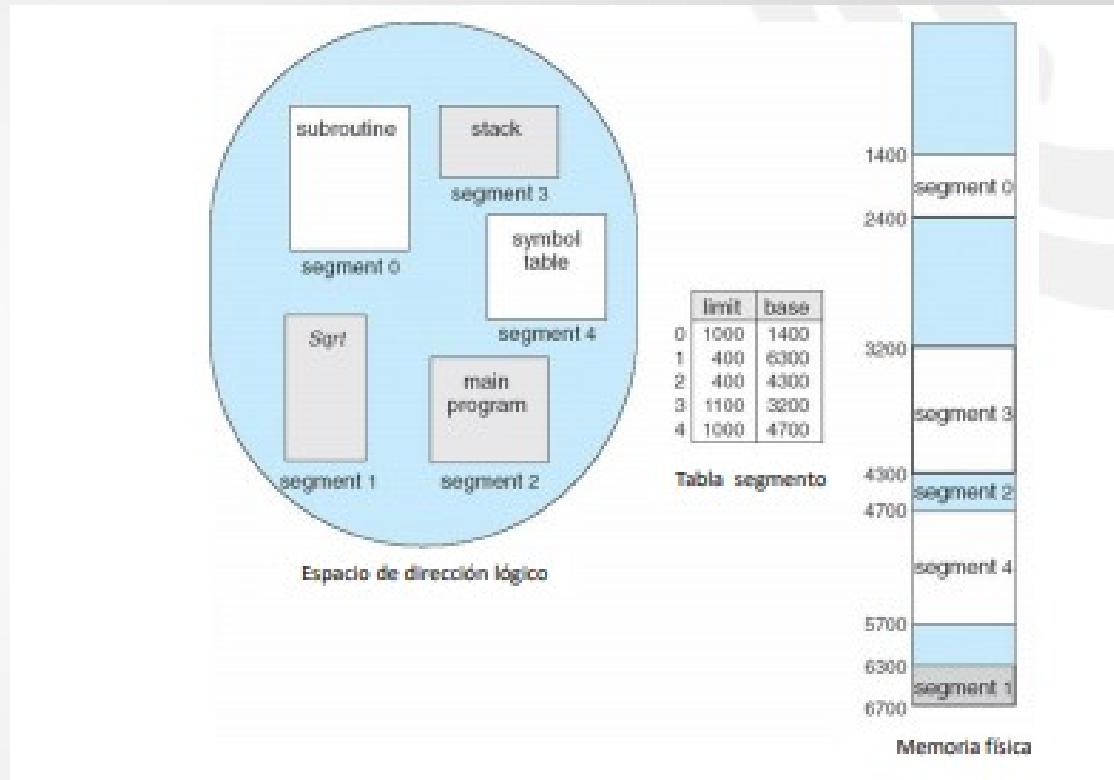
Arquitectura de Segmentación

- Protección. Con cada entrada en la tabla de segmentos se asocia:
 - bit de validación = 0 -> segmento ilegal
 - privilegios read/write/execute
- Bits de protección asociados con segmentos; el código compartido ocurre a nivel de segmento.
- Dado que los segmentos varían en longitud, la alocación de memoria es un problema de alocación dinámica de almacenaje.
- Un ejemplo de segmentación se muestra en el siguiente diagrama

Hardware de Segmentación



Ejemplo de Segmentación

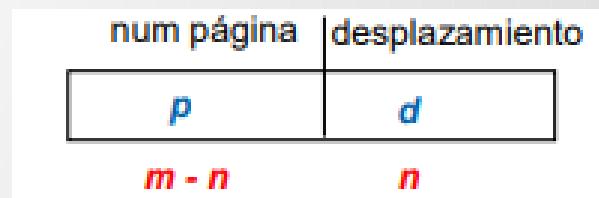


Paginado

- El espacio de direcciones puede no ser contiguo; el proceso es alojado en la memoria física donde haya lugar.
- Se divide a la memoria física en bloques de tamaño fijo llamados marcos (frames) (el tamaño es potencia de 2, entre 512 bytes y 8192 bytes).
- Se divide a la memoria lógica en bloques de tamaño similar llamados páginas.
- Se guarda información de todos los marcos libres.
- Para correr un programa de n páginas, se necesita encontrar n marcos libres y cargar el programa.
- Se establece una tabla de páginas para traducir las direcciones lógicas a físicas.
- Fragmentación interna.

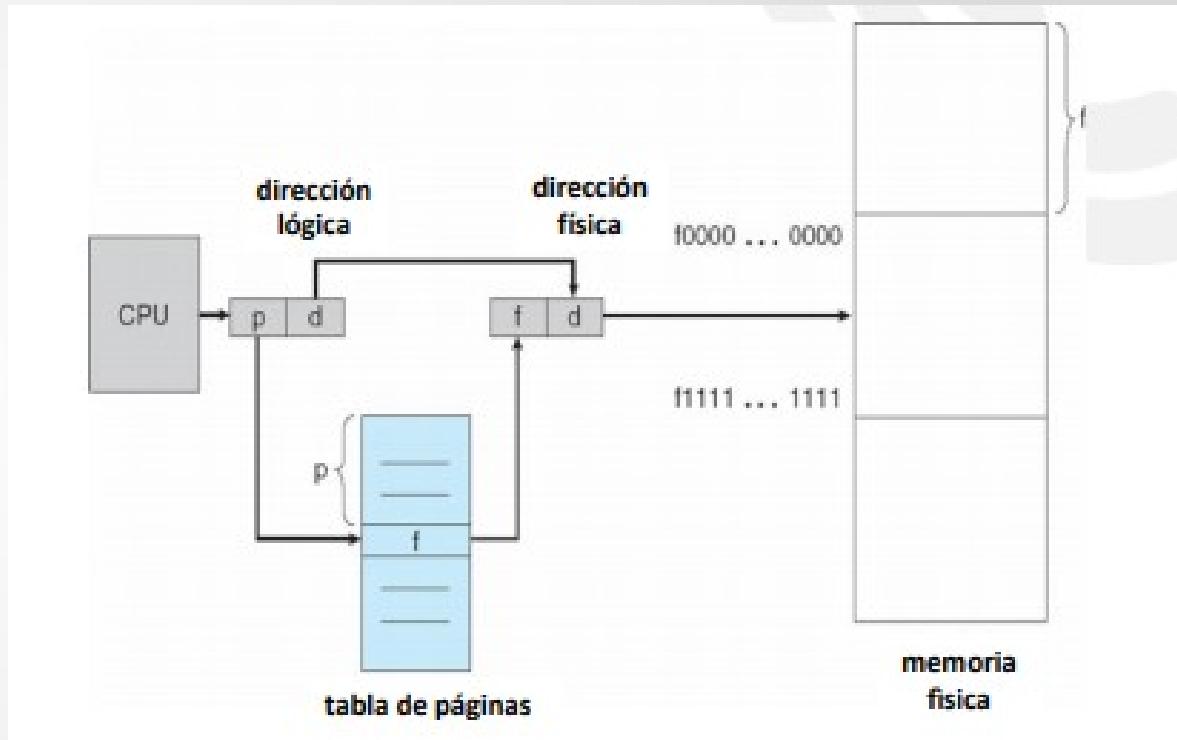
Esquema de Traducción de Direcciones

- La dirección generada por la CPU está dividida en:
 - Número de página (p) – usado como un índice en la tabla de páginas la cual contiene la dirección base de cada página en la memoria física.
 - Desplazamiento en la página (d) – combinado con la dirección base para definir dirección física de memoria que es enviada a la unidad de memoria.

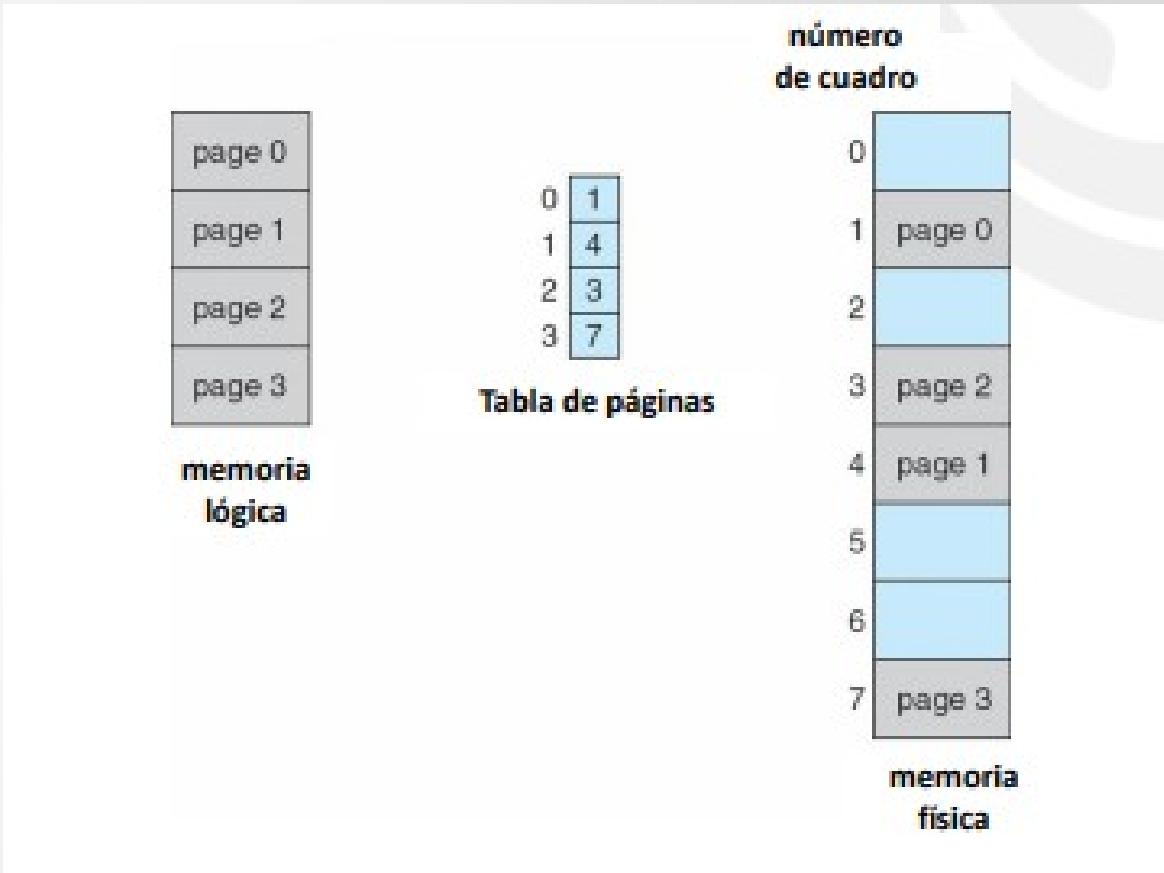


Para un espacio de direcciones 2^m y tamaño de página 2^n

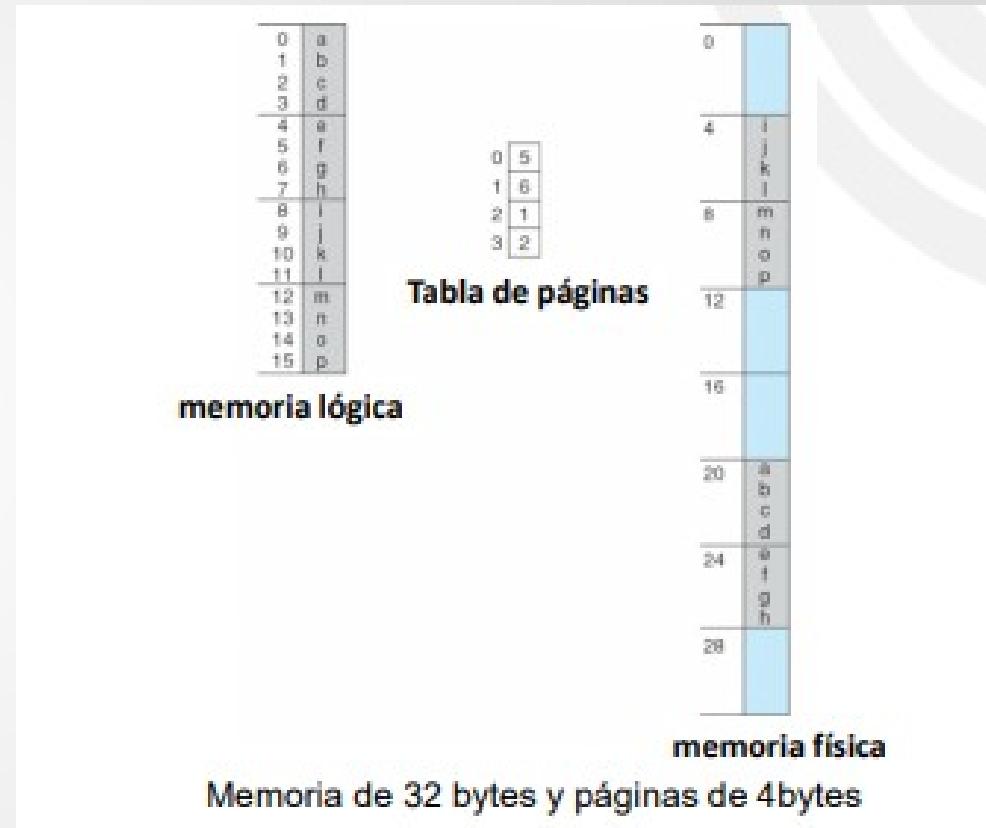
Hardware de Paginado



Modelo de Paginado de Memoria Lógica y Física



Ejemplo de Paginado



Implementación de la Tabla de Páginas

- La tabla de páginas es guardada en memoria principal. }
- El registro base de tablas de páginas (PTBR) apunta a la tabla de páginas.
- El registro de longitud de la tabla de páginas (PRLR) indica el tamaño de la tabla de páginas.
- En este esquema cada acceso a instrucción/dato requiere dos accesos a memoria. Uno para la tabla de páginas y otro para la instrucción/dato.
- El problema del doble acceso puede ser resuelto por el uso de un cache hardware especial de adelantamiento llamado registro asociativo o translation look-aside buffers (TLBs)

Registro Asociativo

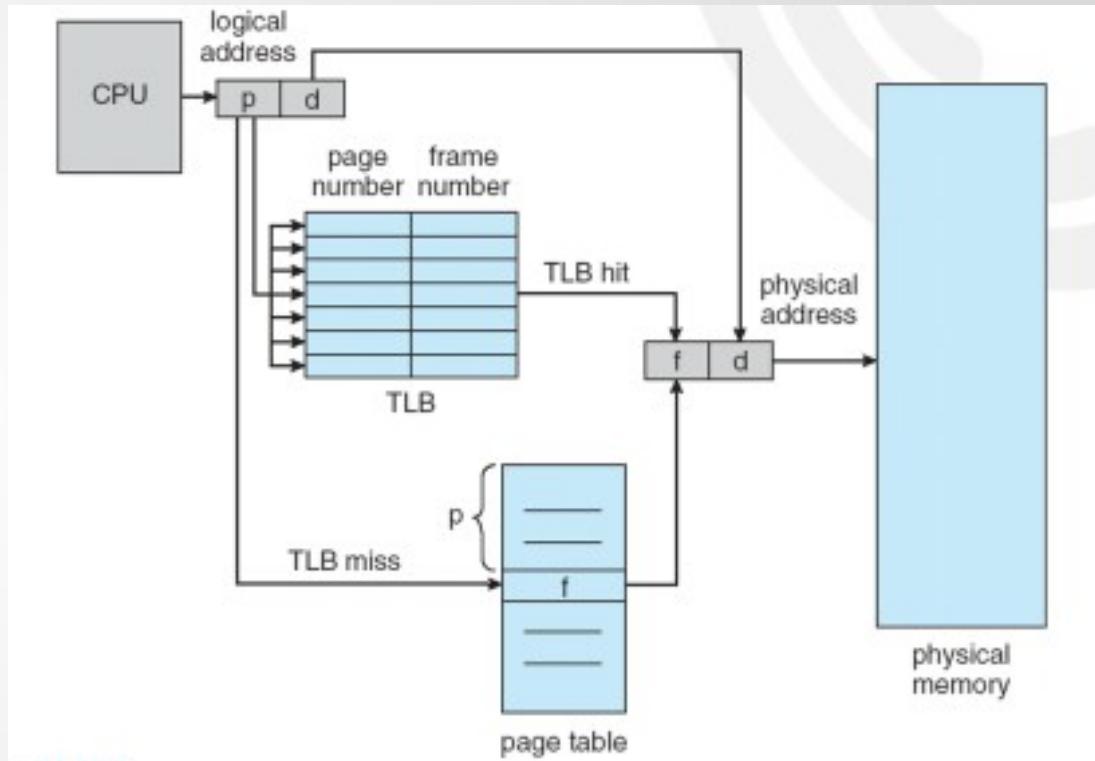
- Registro Asociativo – búsqueda paralela

página #	marco #

Traducción de dirección (p, d)

- Si p es un registro asociativo, tome el marco #.
- Sino tome el marco # desde la tabla de páginas en memoria.

Harward de Paginado con TLB



Tiempo Efectivo de Acceso

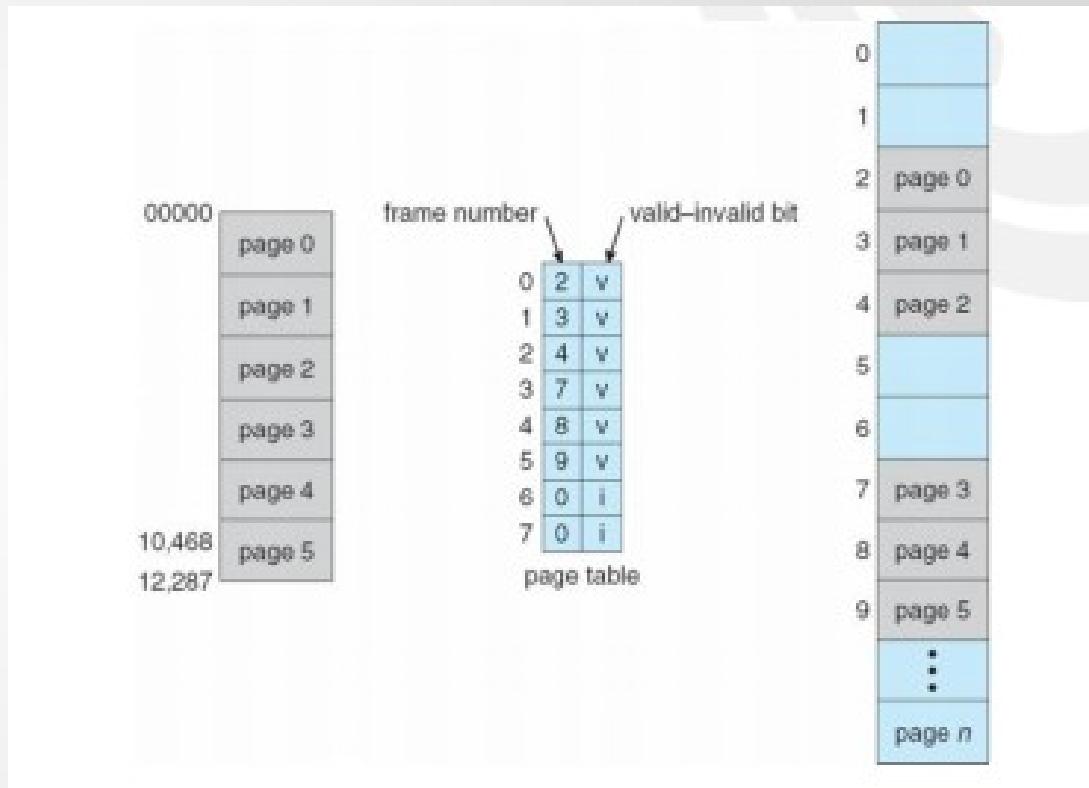
- Búsqueda Asociativa = ε (unidad de tiempo)
- Suponga que el ciclo de memoria es de 1 microsegundo
- Relación de acierto – porcentaje de veces que la página es encontrada en los registros asociativos; está relacionado con el número de registros asociativos.
- Relación de acierto = α
- Tiempo Efectivo de Acceso (TEA)

$$\begin{aligned} \text{TEA} &= (1 + \varepsilon) \alpha + (2 + \varepsilon)(1 - \alpha) \\ &= 2 + \varepsilon - \alpha \end{aligned}$$

Protección de Memoria

- La protección de memoria se implementa asociando un bit con cada marco.
- Un bit de válido-inválido agregado a cada entrada en la tabla de páginas:
 - Válido: indica que la página asociada está en el espacio de direcciones lógicas del proceso, por lo tanto es una página legal.
 - Inválido: indica que la página no está en el espacio de direcciones lógicas del proceso.

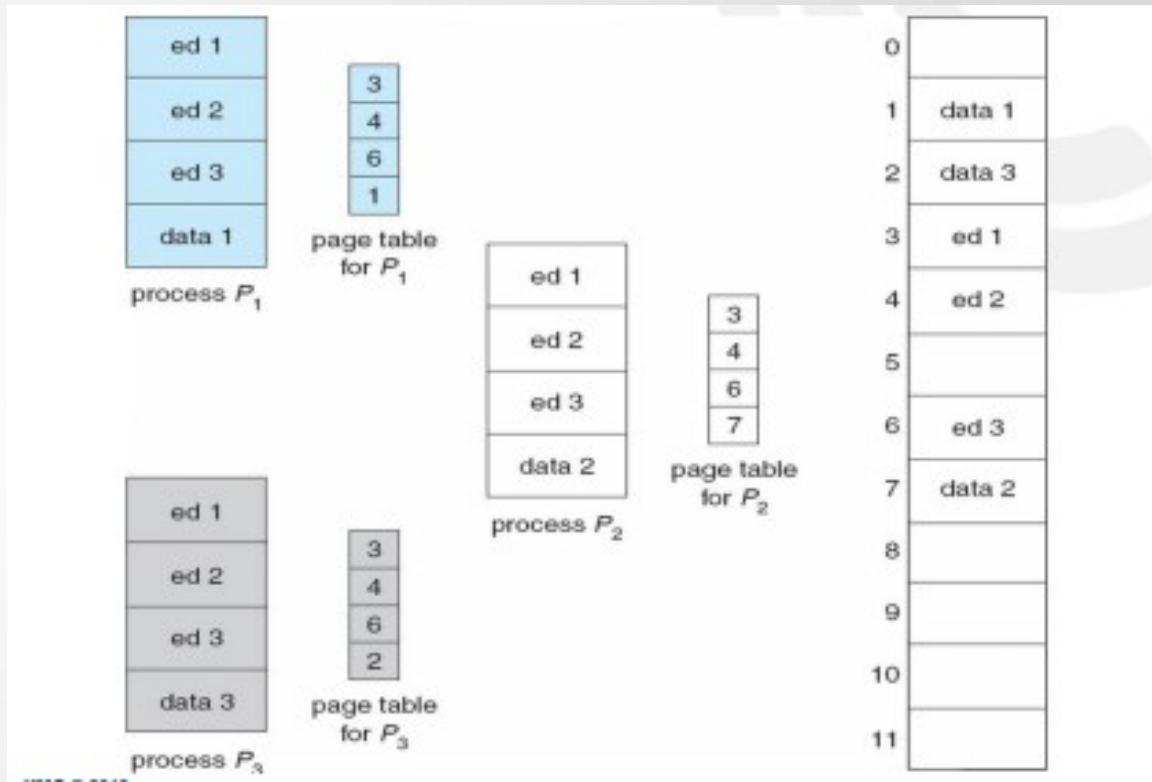
Bit de Válido(V) o Inválido (I) en una tabla de Páginas



Páginas Compartidas

- Código compartido
 - Una copia de código de lectura solamente (reentrant) compartido entre procesos (p.e., editores de texto, compiladores, sistemas de ventanas).
 - El código compartido debe aparecer en la misma locación en el espacio de direcciones de todos los procesos.
- Código privado y datos
 - Cada proceso guarda una copia separada de código y datos.
 - Las páginas del código privado y datos puede aparecer en cualquier lugar del espacio de direcciones lógico.

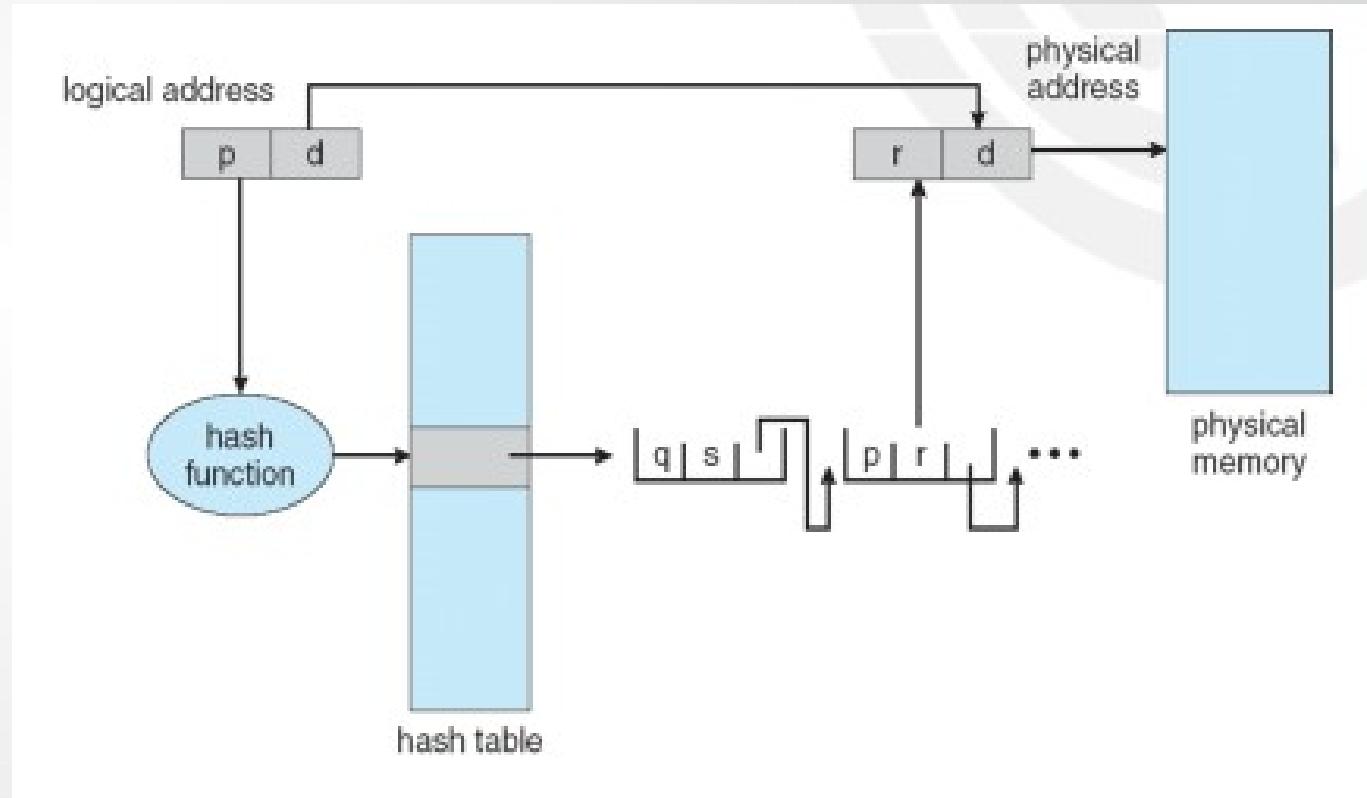
Ejemplo de Páginas Compartidas



Tablas de Páginas con HASH

- Común en espacio de direcciones > 32 bits
- Un número de página virtual es pasada por hash en la tabla de páginas. Esta tabla de páginas contiene una cadena de elementos pasados por hash a la misma locación.
- Los números de páginas virtuales son comparados con lo que existe en estas cadenas. Si se encuentra se extrae el correspondiente marco físico.

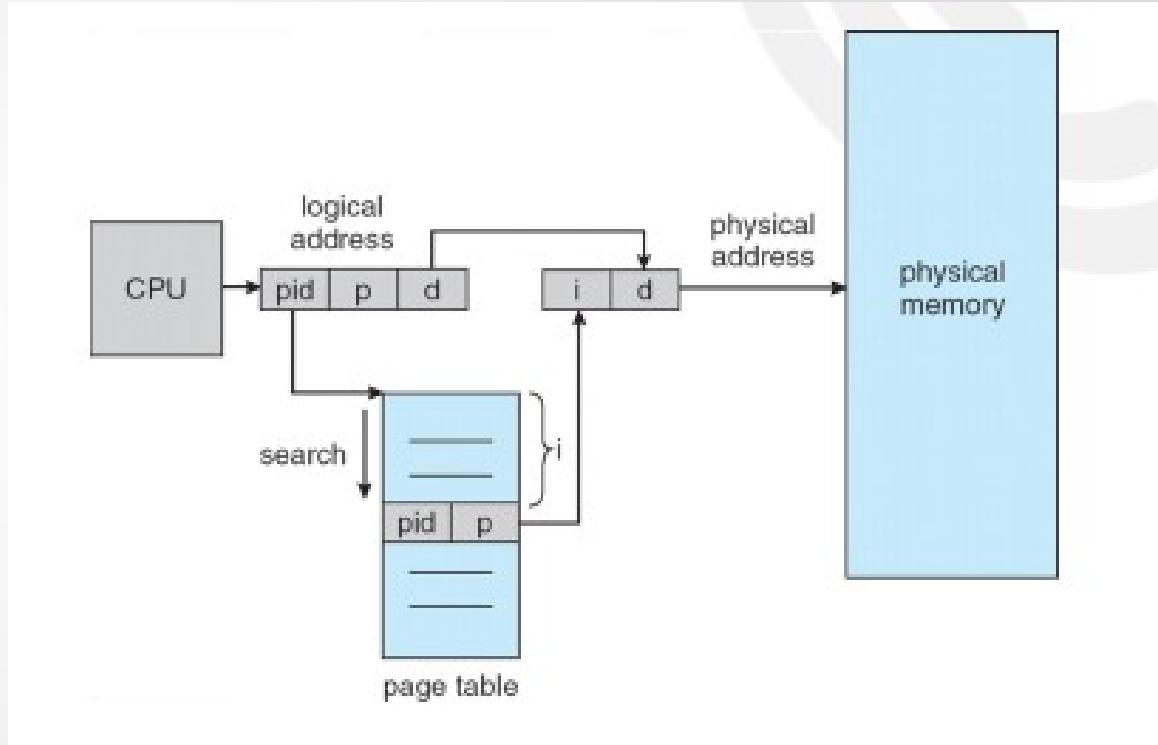
Tablas de Páginas con HASH



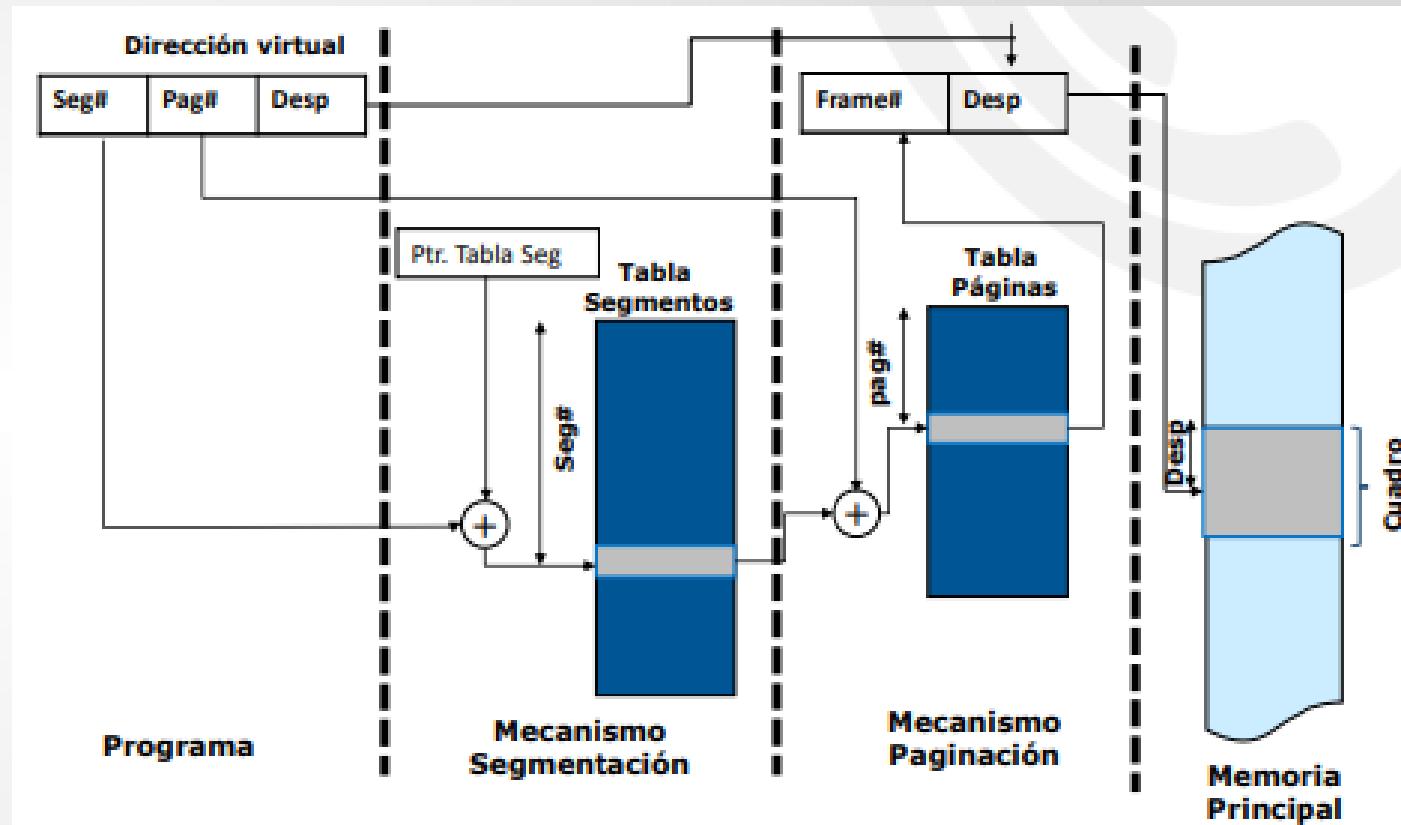
Tablas de Páginas Invertida

- Una entrada por cada página real de memoria.
- La entrada consiste de la dirección virtual de la página almacenada en la locación real de memoria, con información del proceso que es dueño de esa página.
- Decrementa la memoria necesaria para almacenar cada tabla de páginas, pero incrementa el tiempo necesario para buscar en la tabla cuando ocurre una referencia a una página.
- Se usa una tabla hash para limitar la búsqueda a una — o a lo sumo unas pocas — entrada a la tabla de páginas.

Arquitectura de la Tabla de Página Invertida



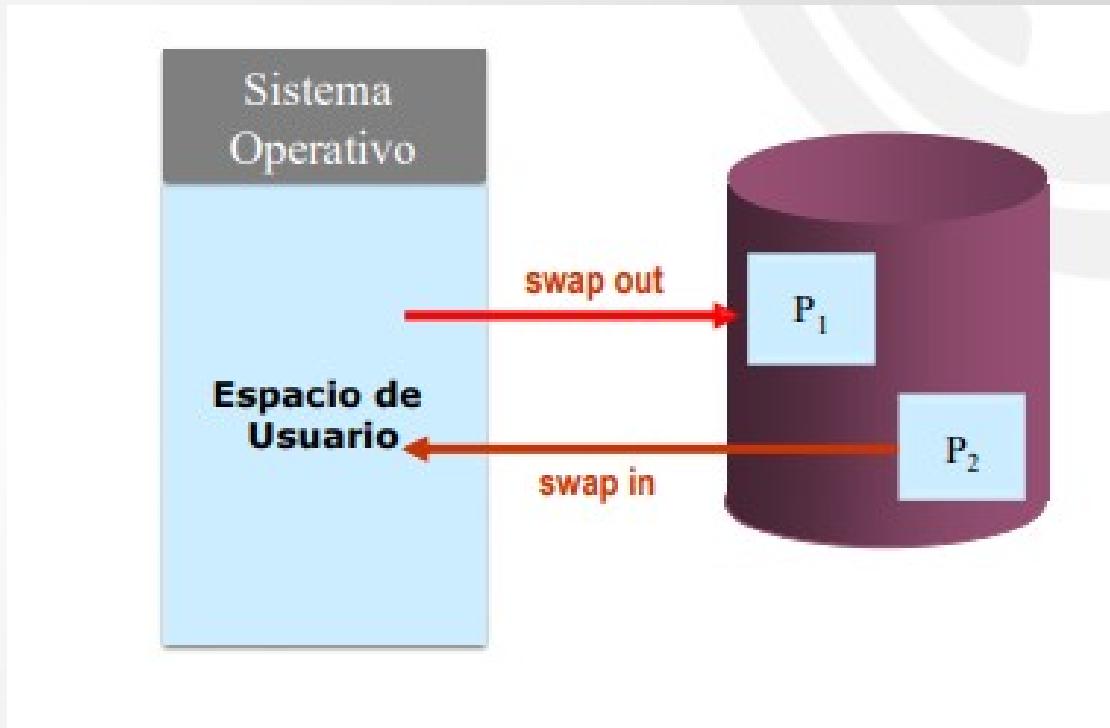
Segmentación y Paginación



Intercambio (Swapping)

- Un proceso puede ser intercambiado (swapped) temporalmente fuera de la memoria a un almacenamiento de respaldo (backing store), y luego ser vuelto a la misma para continuar su ejecución.
- Backing store – espacio en disco lo suficientemente grande para acomodar copias de todas las imágenes de memoria de todos los usuarios, debe proveer acceso directo a todas estas imágenes de memoria.
- Roll out, roll in – variante del intercambio usado en algoritmos de planificación basados en prioridades; procesos con baja prioridad son intercambiados con procesos de alta prioridad que pueden ser cargados y ejecutados.
- La mayor parte del tiempo de intercambio es tiempo de transferencia y es directamente proporcional a la cantidad de memoria intercambiada.

Visión Esquemática del Intercambio





SISTEMAS DE ARCHIVOS

SISTEMA DE ARCHIVOS

REQUERIMIENTOS ESENCIALES

- 1.- Debe ser posible almacenar gran cantidad de información.
- 2.- La información debe sobrevivir a la finalización del proceso que está utilizándola.
- 3.- Múltiples procesos pueden acceder simultáneamente a la información.

SISTEMA DE ARCHIVOS: INTERFAZ

- ▶ Concepto de archivos
- ▶ Métodos de Acceso
- ▶ Estructura de Directorio
- ▶ Montaje de Sistemas de Archivos
- ▶ Archivos Compartidos

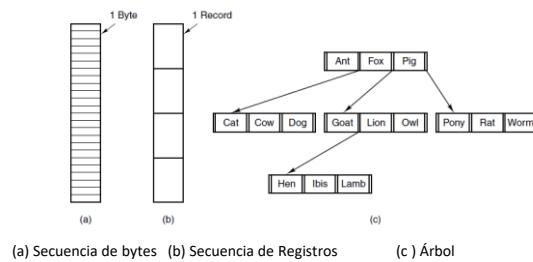
CONCEPTO – ESTRUCTURA. ARCHIVO

Concepto:

- Espacio de direcciones lógicas contiguas. Tipos: Datos ó Programas

Estructura:

- Ninguna – secuencia de palabras, bytes
- Estructura de registros simple
- Estructuras Complejas



(a) Secuencia de bytes (b) Secuencia de Registros

(c) Árbol

ARCHIVO: ATRIBUTOS Y OPERACIONES

ATRIBUTOS

- Nombre
- Tipo
- Locación
- Tamaño
- Protección
- Tiempo, fecha, e identificación de usuario

OPERACIONES

- creación
- escritura
- lectura
- reposición puntero corriente
- borrado
- truncado
- **open(F_i)**
- **close (F_i)**

Información requerida para administrarlos

- Puntero corriente del archivo
- Cuenta de archivo abierto
- Locación en el disco del archivo
- Derechos de acceso

KMC © 2019

LOCKING DE ARCHIVOS ABIERTOS

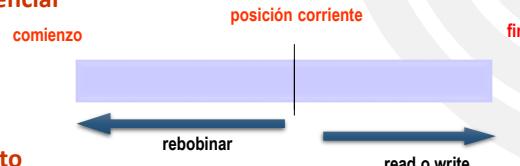
- Provisto por algunos sistemas operativos y sistemas de archivos
- Media en el acceso al archivo
- Mandatorio flexible:
 - **Mandatorio** – el acceso es rechazado dependiendo de los locks que se tienen y requeridos. Ejemplo: Windows
 - **Flexible** – los procesos verifican el estado de los locks y decide qué hacer. Ejemplo: UNIX

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

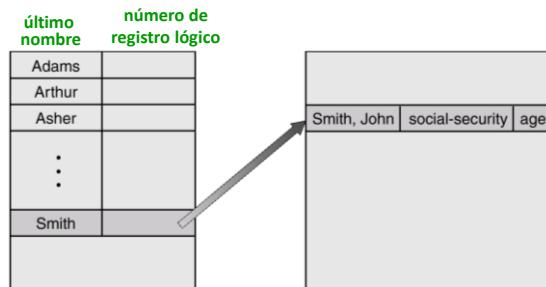
MÉTODOS DE ACCESOS

- **Acceso Secuencial**



- **Acceso Directo**

- **Acceso Indexado**

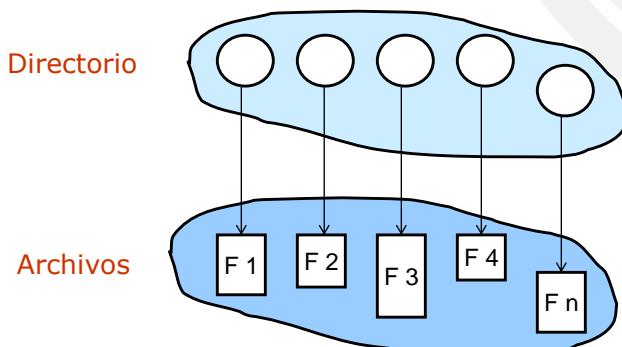


KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESTRUCTURA DE DIRECTORIO

- Una colección de nodos conteniendo información sobre todos los archivos.



- La estructura de directorio y los archivos residen en el disco.

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

DIRECTORIO

INFORMACIÓN

- Nombre
- Dirección
- Longitud corriente
- Máxima longitud
- Fecha del último acceso
- Fecha de la última actualización (para vuelco)
- Tipo
- ID del dueño
- Información de protección

OPERACIONES

- Búsqueda de un archivo
- Creación de un archivo
- Borrado de un archivo
- Listado de un directorio
- Renombrado de un archivo
- Atravesar un sistema de archivos

KMC © 2019

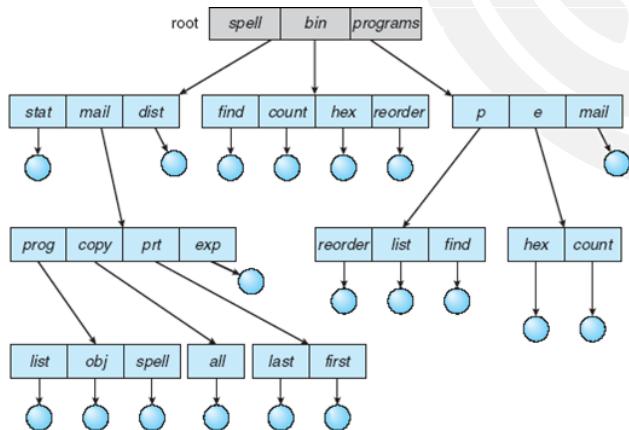
ORGANICE EL DIRECTORIO (LÓGICAMENTE) PARA OBTENER:

- Eficiencia – localizar un archivo rápidamente.
- Nombres – conveniente para los usuarios.
- Agrupamiento – agrupamiento lógico de archivos por propiedades, (p.e., todos los programas C, todos los juegos, ...)

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESTRUCTURA ARBÓREA DE DIRECTORIOS



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESTRUCTURA ARBÓREA DE DIRECTORIOS

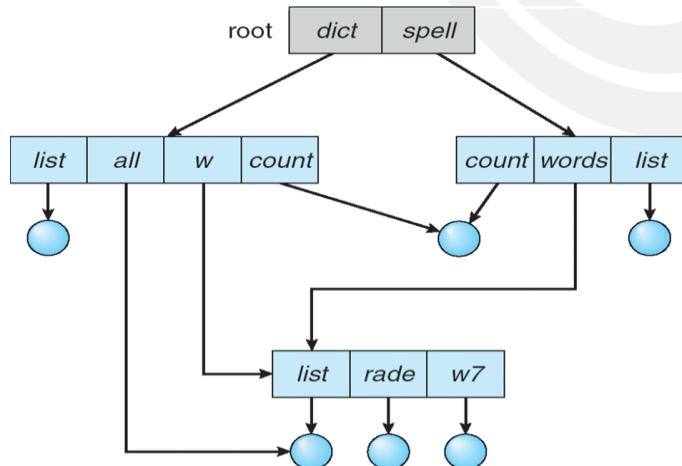
- Búsqueda eficiente
- Capacidad de agrupamiento
- Directorio corriente (directorio de trabajo)
- Camino de nombres absoluto o relativo

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

GRAFO ACÍCLICO DE DIRECTORIOS

- Puede compartir subdirectorios y archivos.



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

GRAFO ACÍCLICO DE DIRECTORIOS

- Dos nombres diferentes (alias)
- Si *dict* borra *list* ⇒ quedan punteros solitarios.

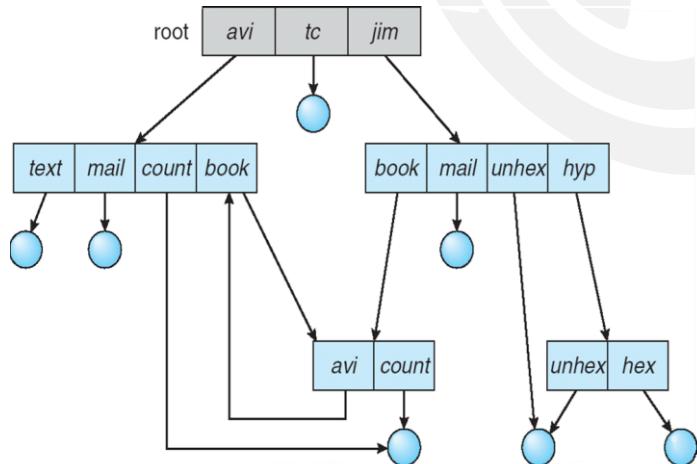
Soluciones:

- ▶ Punteros hacia atrás, así se pueden borrar todos los punteros. Los registros de tamaño variable son un problema.
- ▶ Punteros hacia atrás usando una organización “cadena margarita”.
- ▶ Contador de entradas al archivo.
- Nueva entrada en el directorio
 - ▶ Link – Otro nombre (puntero) a un archivo existente
 - ▶ Resuelva el link – siga el puntero hasta localizar el archivo

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

GRAFO GENERAL DE DIRECTORIO



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

GRAFO GENERAL DE DIRECTORIO

- **¿Cómo se garantiza que no haya ciclos?**
 - ▶ Permite enlaces (links) a archivos y no a sudirectorios.
 - ▶ “Garbage collection”.
 - ▶ Cada vez que se agrega un nuevo enlace (link) se usa un algoritmo de detección de ciclos para determinar si está bien.

KMC © 2019

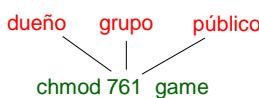
SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

PROTECCIÓN

- El **creador/dueño** del archivo debería poder controlar:
 - ▶ que cosas pueden hacerse
 - ▶ por quién
- Tipos de acceso
 - ▶ Read
 - ▶ Write
 - ▶ Execute
 - ▶ Append
 - ▶ Delete
 - ▶ List

LISTAS DE ACCESO Y GRUPOS

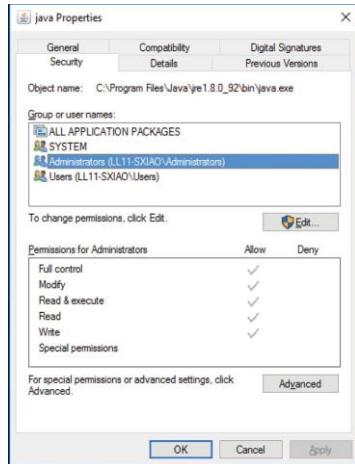
- Modos de acceso: read, write, execute
- Tres clases de usuarios
 - a) acceso dueño 7 ⇒ 1 1 1
 - b) acceso grupos 6 ⇒ 1 1 0
 - c) acceso público 1 ⇒ 0 0 1
- Pedir al administrador crear un grupo (único nombre), sea G, y adicionar algún usuario al mismo.
- Para un archivo particular (sea *game*) o subdirectorio, definir un acceso apropiado.



Agregar un grupo a un archivo

chgrp *G* *game*

EJEMPLOS: WINDOWS Y UNIX-LINUX



-rw-rw-r--	1	pbg	staff	31200	Sep 3 08:30	intro.ps
drwx-----	5	pbg	staff	512	Jul 8 09:33	private/
drwxrwxr-x	2	pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2	pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1	pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1	pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4	pbg	faculty	512	Jul 31 10:31	lib/
drwx-----	3	pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3	pbg	staff	512	Jul 8 09:35	test/

KMC © 2019

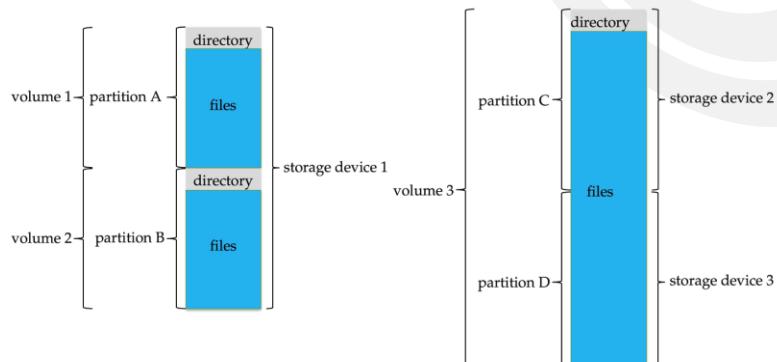
SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

**SISTEMAS DE
ARCHIVOS
INTERNAL**

SISTEMA DE ARCHIVOS

- Las computadoras de uso general pueden tener múltiples dispositivos de almacenamiento.
- Los dispositivos se pueden dividir en particiones, que contienen volúmenes
- Los volúmenes pueden abarcar múltiples particiones.
- Cada volumen generalmente formateado en un sistema de archivos

UNA ORGANIZACIÓN TÍPICA DE UN SISTEMA DE ARCHIVOS



ESTRUCTURA ALMACENAMIENTO SOLARIS

/	ufs
/devices	devfs
/dev	dev
/system/contract	ctfs
/proc	proc
/etc/mnttab	mntfs
/etc/svc/volatile	tmpfs
/system/object	objfs
/lib/libc.so.1	lofs
/dev/fd	fd
/var	ufs
/tmp	tmpfs
/var/run	tmpfs
/opt	ufs
/zpbge	zfs
/zpbge/backup	zfs
/export/home	zfs
/var/mail	zfs
/var/spool/mqueue	zfs
/zpbg	zfs
/zpbg/zones	zfs

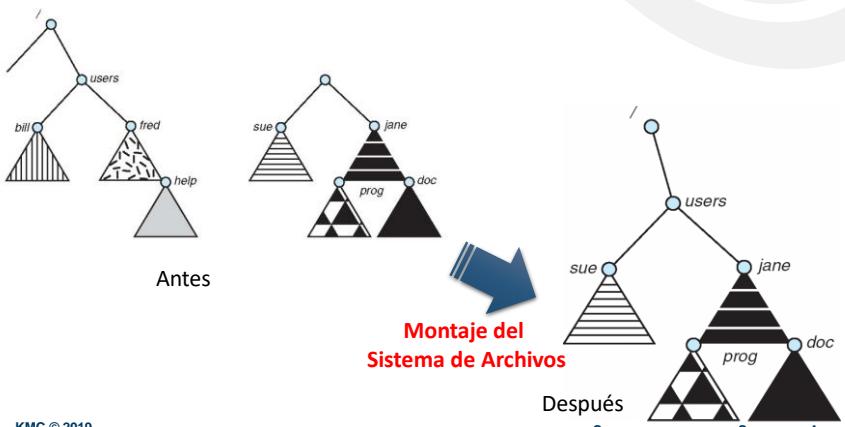
En un sistema operativo
puede mantener múltiples
sistemas de archivos

Sistema de archivo virtual
con información de todos
los procesos

Sistemas de archivo de
propósito general

MONTAJE DE SISTEMA DE ARCHIVOS (MOUNTING)

- Un sistema de archivos debe ser **montado** antes de que pueda ser accedido.
- Un sistema de archivos sin montar es montado en un **PUNTO DE MONTAJE (MOUNT POINT)**.



ARCHIVOS COMPARTIDOS

- La acción de compartir debe ser hecha por medio de un esquema de protección.
- En sistemas distribuidos los archivos pueden ser compartidos a través de la red.
- Network File System (NFS) es un método común de compartir archivos distribuidos.
- Los **User IDs** identifican usuarios, admiten permisos y protección por usuarios.
- Los **Group IDs** admite agrupar usuarios en grupos, permitiendo asignar al mismo derechos de acceso.

ARCHIVOS COMPARTIDOS – SEMÁNTICA DE CONSISTENCIA

- La **semántica de consistencia** especifica como múltiples usuarios acceden a un archivo compartido simultáneamente
 - **SEMÁNTICA UNIX.** El sistema de archivos Unix (UFS) implementa:
 - Las escrituras a un archivo abierto son visibles inmediatamente a los otros usuarios que comparten el mismo archivo abierto
 - El puntero a archivos compartidos permite que múltiples usuarios lean y escriban concurrentemente
 - **SEMÁNTICA DE SESIÓN.** AFS tiene una semántica de sesión
 - Las escrituras son solo visibles solo después que la sesión termina.
 - **SEMÁNTICA DE ARCHIVOS COMPARTIDOS INMUTABLES**



SISTEMAS DE ARCHIVOS IMPLEMENTACIÓN

OBJETIVOS

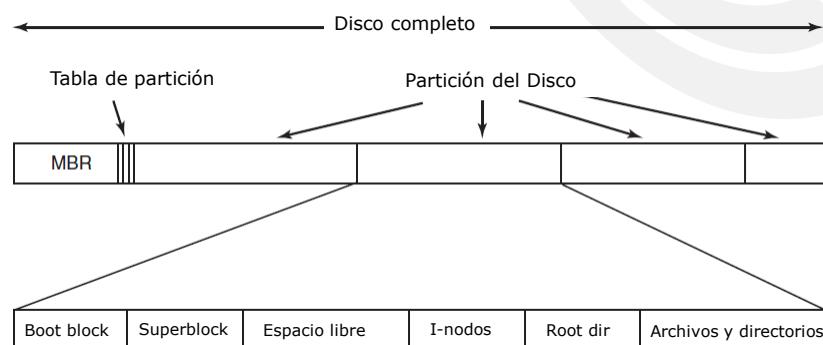
- Describir los detalles locales de la implementación del sistema de archivos y estructuras de directorios
- Discutir algoritmos de alocación de bloques y bloques libres y compromisos

DISEÑO DEL SISTEMA DE ARCHIVOS

- El sistema de archivos se almacena en disco.
- Los discos pueden dividirse en varias particiones.
- MBR (Master Boot Record) se encuentra en el sector 0 del disco.
- Tabla de particiones.
- Boot Block, Superblock

DISEÑO DEL SISTEMA DE ARCHIVOS

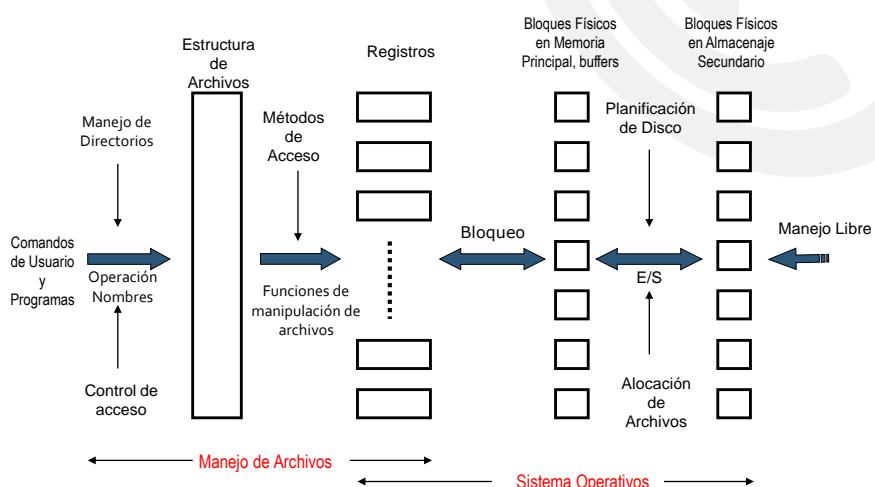
Ejemplo de un posible diseño



ESTRUCTURA DEL SISTEMA DE ARCHIVOS

- Estructura de Archivo
 - Unidad Lógica de almacenamiento
 - Colección de información relacionada
- El sistema de archivos reside en almacenamiento secundario (discos).
- El sistema de archivo está organizado en capas.
- **FILE CONTROL BLOCK (FCB)** – estructura de almacenaje consistente de información sobre el archivo.

SISTEMA DE ARCHIVOS EN GENERAL



UN BLOQUE DE CONTROL DE ARCHIVOS TÍPICO

FCB: File Control Block

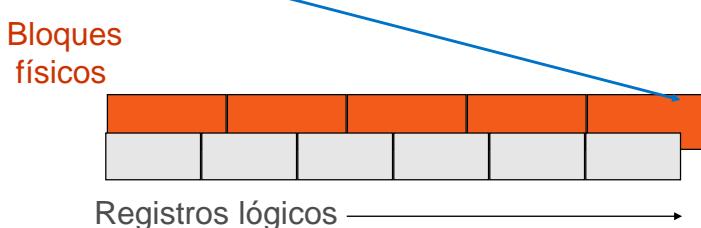
file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESTRUCTURAS DE ARCHIVO

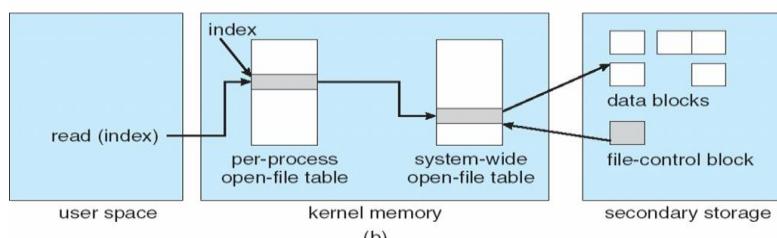
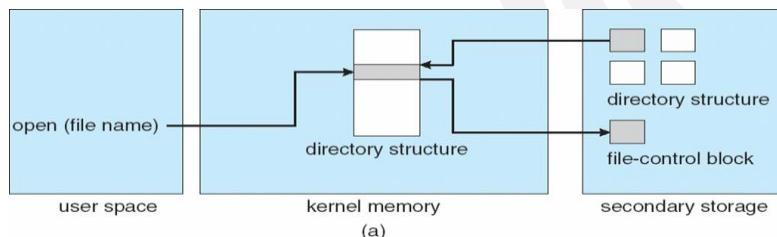
- Bloques Físicos
- Registros Lógicos
- Fragmentación



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESTRUCTURAS DEL SISTEMA DE ARCHIVOS EN MEMORIA



a) Apertura de un archivo

b) Lectura de un archivo

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

IMPLEMENTACIÓN DE DIRECTORIO

- Lista lineal de nombres de archivos con punteros a los bloques de datos.
- Tabla *hash* – Lista lineal con estructura de datos *hash*.

KMC © 2019

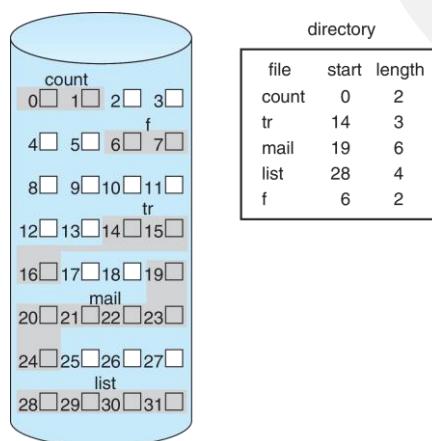
SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

MÉTODOS DE ALOCACIÓN

Un método de alocación se refiere a cómo los bloques de disco de un archivo son ubicados:

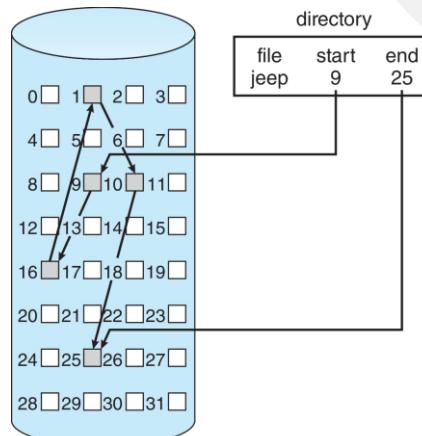
- Alocación Contigua
- Alocación Enlazada
- Alocación Indexada

ALOCACIÓN CONTIGUA



ALOCACIÓN ENLAZADA

- Cada archivo es una lista enlazada de bloques de disco: los bloques pueden estar en cualquier lugar del disco.



KMC © 2019

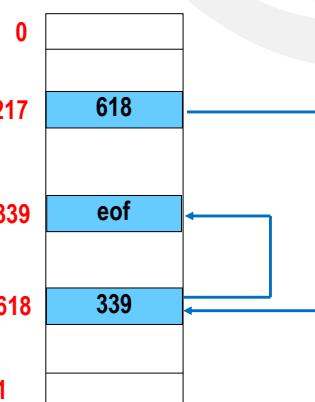
SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ALOCACIÓN ENLAZADA

- **File-Allocation Table (FAT)** – alocación de espacio de disco usado en MS-DOS y OS/2.

entrada de directorio

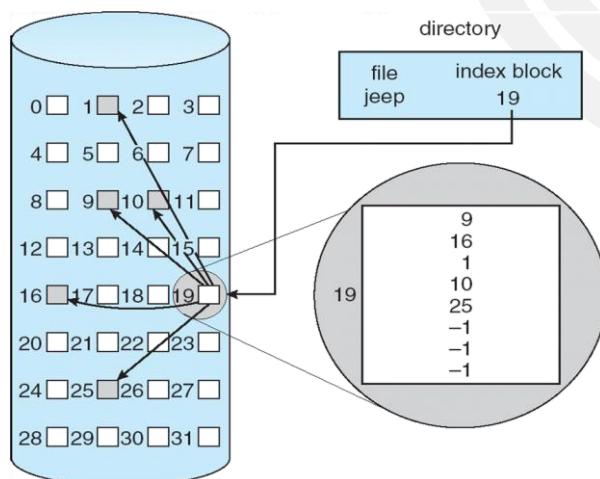
test	...	217
nombre	...	bloque inicial



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

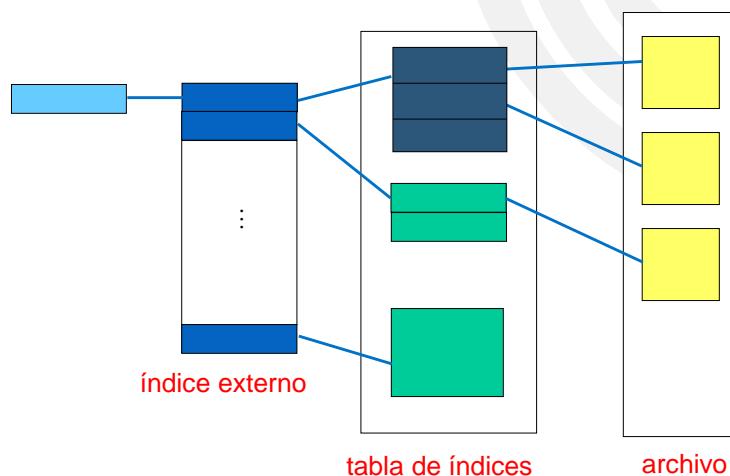
ALOCACIÓN INDEXADA



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

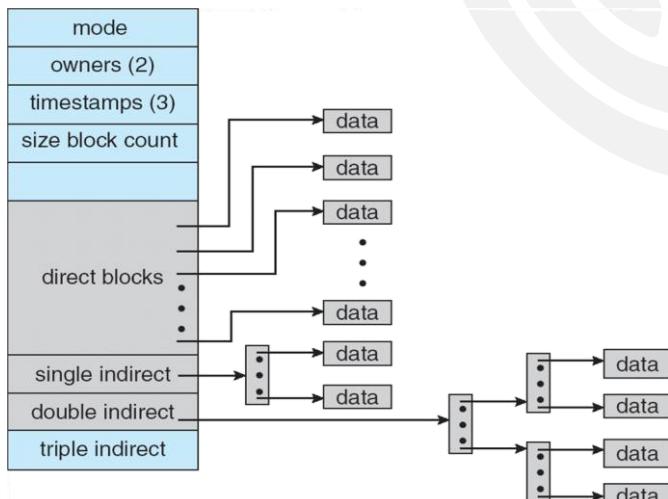
ALOCACIÓN INDEXADA – MAPEO



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESQUEMA COMBINADO: UNIX (4K BYTES POR BLOQUE)



KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ESQUEMA COMBINADO: UNIX

Los pasos para buscar un archivo `/usr/ast/mbox`.

Root directory	
1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up
usr yields
i-node 6

I-node 6 is for /usr	
Mode	
size	
times	
132	

I-node 6
says that
/usr is in
block 132

Block 132 is /usr directory	
6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast
is i-node
26

I-node 26 is for /usr/ast	
Mode	
size	
times	
406	

I-node 26
says that
/usr/ast is in
block 406

Block 406 is /usr/ast directory	
26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox
is i-node
60

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

ADMINISTRACIÓN DE ESPACIO LIBRE

- Vector de Bits – bit map (n bloques)

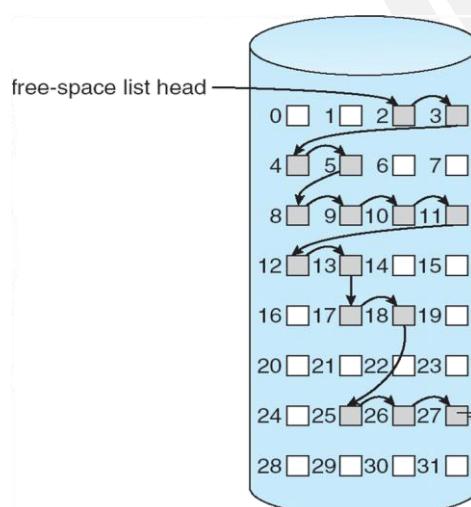


$$\text{bit}[j] = \begin{cases} 1 \Rightarrow \text{bloque}[j] \text{ libre} \\ 0 \Rightarrow \text{bloque}[j] \text{ ocupado} \end{cases}$$

- Cálculo del número de bloque

(número de bits por palabra) *
(número de palabras con valor 0) +
offset del primer bit 1

ADMINISTRACIÓN DE ESPACIO LIBRE



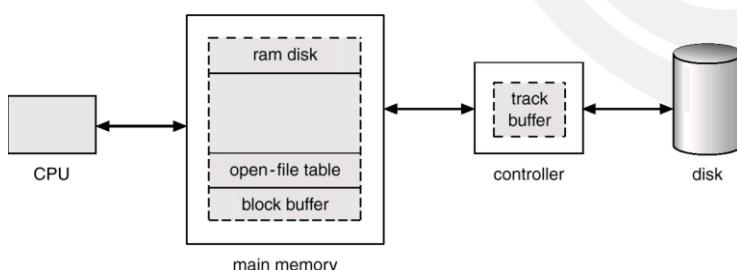
EFICIENCIA Y DESEMPEÑO

- La eficiencia depende de:
 - ▶ alocacion en el disco y algoritmos de directorio
 - ▶ tipos de datos mantenidos en la entrada de directorio del archivos
- Desempeño
 - ▶ caché de disco – sección separada de memoria principal para bloques frecuentemente usados
 - ▶ *free-behind* y *read-ahead* – técnicas para optimizar el acceso secuencial
 - ▶ mejora del desempeño de la PC dedicando una sección de la memoria como disco virtual, o disco RAM.

KMC © 2019

SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

DISTINTAS LOCACIONES DEL CACHÉ DE DISCO



KMC © 2019

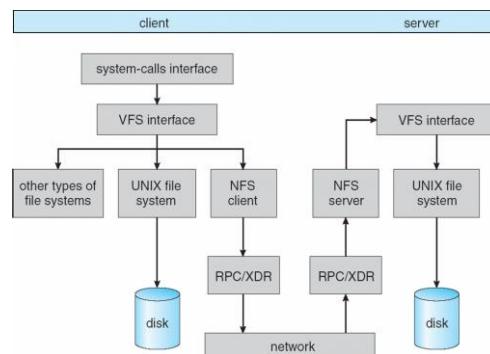
SISTEMAS OPERATIVOS – SISTEMA DE ARCHIVOS

RECUPERACIÓN

- Verificador de Consistencia – compara datos en la estructura de directorio con bloques de datos en el disco, y trata de reparar inconsistencias.
- Uso de programas de sistema para *respaldar* (*back up*) datos del disco a otro dispositivo de almacenaje (cinta magnética, óptica, etc).
- Se recuperan archivos perdidos o disco por *restauración* de datos del backup.

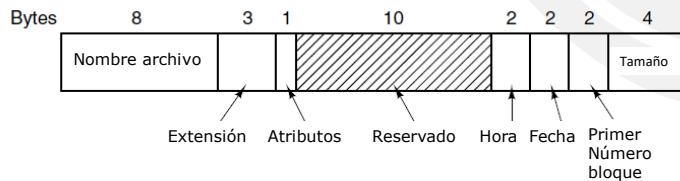
EL SISTEMA DE ARCHIVOS DE RED DE SUN (NFS)

- Es una implementación y una especificación de un sistema de software para acceder a archivos remotos a través de LANs (o WANs).
- La implementación es parte de los sistemas operativos Solaris y SunOS que corre sobre estaciones de trabajo Sun usando un protocolo no confiable datagrama (protocolo UDP/IP) y Ethernet.



SISTEMA DE ARCHIVO – MS-DOS

- Entrada de directorio



Sistema de Archivos FAT tiene tres versiones:

- ✓ FAT-12
- ✓ FAT-16
- ✓ FAT -32

SISTEMA DE ARCHIVO – MS-DOS

- Máximo tamaño de la partición para diferentes tamaños de bloques

Tam. Bloque	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "*Operating System Concepts*", 7^{ma} Edición 2009, 9^{na} Edición 2012, 10^{ma} Edición 2018.
- Tanenbaum, A.; "*Modern Operating Systems*", Addison-Wesley, 3^{ra} Edición 2008, 4^{ta}. Edición 2014.
- Stallings, W. "*Operating Systems: Internals and Design Principles*", Prentice Hall, 6^{ta} Edición 2009, 7^{ma} Edición 2011, 9^{na} Edición 2018.