



marinamuca01

www.wuolah.com/student/marinamuca01

7843

Tema-2-FS.pdf

Apuntes FS



1º Fundamentos del Software



Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada



INESEM
BUSINESS SCHOOL

Escuela de LÍDERES

Master BIM Management



60 Créditos ECTS



AUTODESK®
REVIT®

AUTODESK®
NAVISWORKS™

Jose María Girela
Bim Manager.



TEMA 2

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

EVOLUCIÓN DE LOS SISTEMAS OPERATIVOS

PROCESAMIENTO EN SERIE

Programa da interacciones directamente con la máquina \Rightarrow S.O.

Problemas principales:

- Elevado tiempo de planificación y configuración
- Tiempo de procesamiento de CPU desaprovechado.

SISTEMAS EN LOTES SENCILLOS (sistemas Batch)

Se introduce una pieza de software denominada monitor, de esta forma el programador no tiene que acceder directamente a la máquina, sino que envía un trabajo a través de una tarjeta o cinta al operador del computador que crea un sistema por lotes agrupando trabajos similares, para que lo utilice el monitor. Cuando finaliza un programa devuelve el control al monitor, que carga el siguiente programa.

Problema:

- CPU sin hacer nada durante operaciones E/S.
- Sobrecarga a causa del consumo de parte de la memoria principal y tiempo de máquina por parte del computador

SISTEMAS EN LOTES MULTIPROGRAMADOS

Los sistemas donde existe un único programa se denomina monoprogramación: el procesador ejecuta durante un cierto tiempo hasta que llega a una instrucción de E/S, entonces debe esperar a que ésta concluya para continuar.

Esta ineficiencia se puede evitar ya que hay memoria suficiente para alojarse al S.O. y varios programas usuario. De modo que cuando un proceso tiene que esperar por la E/S se asigna el procesador a otro proceso. Esto es lo que se conoce como multiprogramación. Existen 2 tipos:

- No apropiativa: proceso activo se ejecuta hasta que:
 - Termina
 - Se bloquea por operación E/S o un servicio solicitado por el S.O.
 - Hace una llamada al S.O. para ceder el procesador a otro proceso.
- Apropiativa o preferente:
 - El SO puede interrumpir en cualquier momento el proceso en ejecución
 - Las decisiones de cuándo se detiene la ejecución de un proceso y de cuál pasa a ejecutarse se efectúa siguiendo un algoritmo de planifi-



Disfruta ahora de manera online
Cursos intensivos
Clases particulares

30 HORAS

120€

**Tu academia de idiomas online y tu
centro examinador de Cambridge.**

Cursos super-intensivos de preparación B1, B2, C1 y C2. Comienzo 6 de abril. Fin 30 de abril.



Cambridge Assessment
English
Authorised Exam Centre
Official Examination Centre No: ES815

www.clgranada.com

📞 | +34 958 53 52 53 📍 | C/ Puentezuelas, nº 32, 1^a planta (Granada)

✉️ | info@clgranada.com

cación determinado (ejemplo: turno rotatorio, donde se asigna un periodo de tiempo a cada proceso).

SISTEMAS DE TIEMPO COMPARTIDO

Se utiliza la multiprogramación para gestionar múltiples trabajos interactivos. Compartiendo el tiempo de procesador entre múltiples usuarios, que acceden simultáneamente a través de terminales, el S.O. se encarga de entrelazar la ejecución de cada programa de usuario.

Tabla 2.3. Multiprogramación en lotes frente a tiempo compartido.

	Multiprogramación en lotes	Tiempo compartido
Objetivo principal	Maximizar el uso del procesador	Minimizar el tiempo de respuesta
Fuente de directivas al sistema operativo	Mandatos del lenguaje de control de trabajos proporcionados por el trabajo	Mandatos introducidos al terminal

CONCEPTO DE PROCESO

Se han dado muchas definiciones de este concepto, quizá la más adecuada sea: "Unidad de actividad caracterizada por un hilo secuencial de ejecución, un estado actual y un conjunto de recursos del sistema asociados."

Un proceso está formado por:

- Un programa ejecutable
- Datos que necesita el S.O. para ejecutar el programa:
 - del programa
 - del contexto de ejecución.

Principal herramienta programadora = **INTERRUPCIÓN** ⇒ cualquier trabajo puede suspender su ejecución por la ocurrencia de un evento como la finalización de una operación de E/S. La CPU guarda el contexto y salta a una rutina de tratamiento de interrupción, después continua el procesamiento del proceso interrumpido o cualquier otro.

Causas principales de errores:

- **Sincronización inapropiada**, rutina se suspende esperando por algún evento en el sistema (ej. E/S)
- **Violación de la exclusión mutua**: más de un programa/usuario intentan hacer uso de recursos compartidos simultáneamente.
- **Operación no determinista de un programa**, programas que comparten memoria e interfieren entre ellos sobrescribiendo zonas de memoria comunes
- **Interbloqueos**, dos o + programas se quedan bloqueados esperando se entre si.

CONTEXTO O ESTADO DE UN PROCESO

Conjunto de datos interno por el cual el S.O. es capaz de supervisar y controlar un proceso. Esta información está separada del proceso para que sólo pueda acceder a ella el S.O.

Incluye información

- Para gestionar el proceso por parte del S.O.
- Para ejecutar el proceso por parte del procesador
- El contenido de los registros del procesador (PC, de datos, etc.)
- De uso del S.O. (prioridad, si está esperando por E/S, ...).

IMPLEMENTACIÓN DE PROCESOS

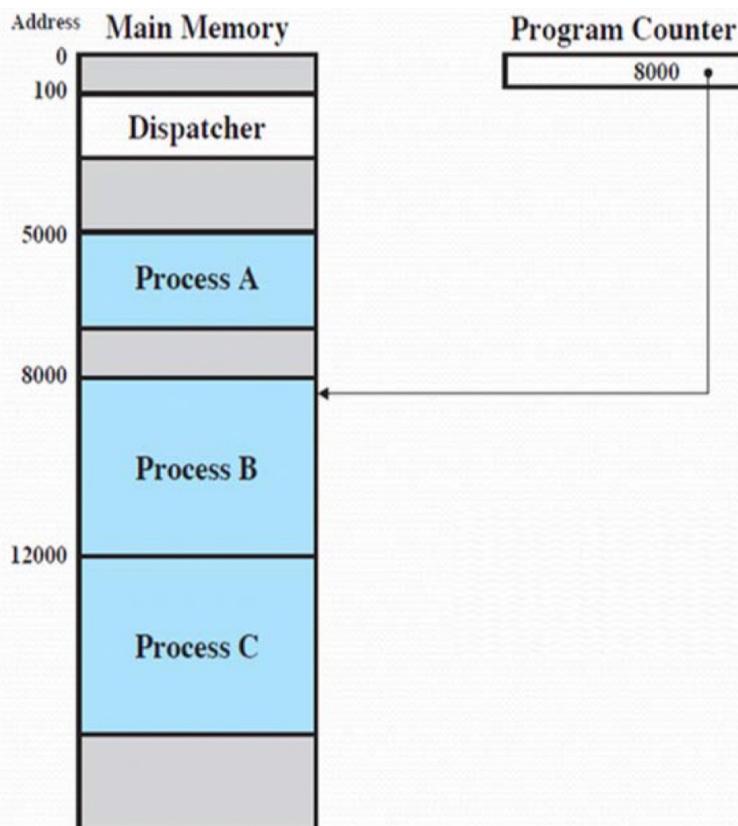
- Se le asigna un bloque de memoria y se incluye en la lista de procesos del S.O. incluyendo un puntero a la ubicación del bloque de memoria que contiene el proceso.
- Se almacena el estado en el PCB.

El proceso puede verse como una estructura de datos, ésta permite el desarrollo de técnicas que aseguran la coordinación y cooperación entre procesos.

TRAZA DE EJECUCIÓN

Es un listado de la secuencia de las instrucciones de un programa que realiza el procesador para un proceso. Desde el punto de vista del procesador se entremezclan las trazas de ejecución de los procesos y las del código del S.O.

Aclaración: El dispatcher se encarga de ver cuál es el siguiente proceso que se va a llevar a memoria y ejecutar el siguiente proceso.



Ya puedes sacarte tu B1/B2/C1 de inglés desde casa



LLAMADAS AL SO

forma de comunicación de programas usuario y SO en tiempo de ejecución. Son peticiones de servicio que hace un proceso al SO.

Ejemplos: Solicitud E/S, gestión procesos, gestión memoria, etc.

Se implementan a través de una trampa o "interrupción de software" (TRAP)

Pueden requerir o no una espera:

→ Cuando no hay espera:

- La instrucción TRAP genera la interrupción
- El procesador la acepta.
- Se ejecuta la rutina de interrupción y se determina el punto de acceso al servicio solicitado.
- Llama al servicio y ejecuta el correspondiente código.
- Se retorna a la rutina genérica restituyendo los registros y volviendo a la instrucción siguiente al TRAP.

→ Cuando hay espera = tratamiento en 2 fases:

1) Inicia el servicio = caso anterior

2) Termina el servicio.

- Se ejecuta el planificador, se bloquee el proceso y se pone en ejecución el proceso seleccionado ⇒ cambio contexto
- Evento = fin espera ⇒ interrupción ejecuta en el contexto de otro proceso y tendrá una parte aplazada.
- Si la operación se completa => Proceso Listo
- Planificador selecciona proceso de nuevo => continua ejecución completando la 2ª fase.
- Finalmente se genera un argumento de retorno del servicio y se restituyen los registros visibles y se retorna al proceso que sigue su ejecución en modo usuario.

CREACIÓN DE UN PROCESO

Se construye estructuras de datos que se usan para manejar el proceso y reserva espacio de direcciones en MP para el proceso.

Eventos que llevan a la creación de un proceso:

- Entorno lote = respuesta a solicitud de trabajo.
- Entorno interactivo = cuando un nuevo usuario entra en el sistema.
- A petición de una aplicación.
- Creado por otro proceso existente. (padre → hijo)

Procedimiento:

- ① Asignación de identificador
- ② Reservar espacio en memoria para el proceso
- ③ Inicialización del BPIC.
- ④ Establecer los enlaces apropiados

⑤ Creación o expansión de estructuras de datos.

TERMINACIÓN DE PROCESOS

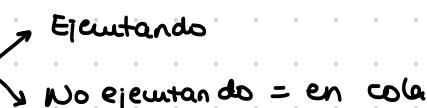
Mecanismos mediante los cuales un proceso indica su finalización:

- Entorno lote ⇒ instrucción HALT o llamada a un servicio de S.O.
- Entorno interactivo ⇒ acciones del usuario.
- Ordenador personal ⇒ salir de una aplicación
- Condición de error o fallo.

Tabla 3.2. Razones para la terminación de un proceso.

Finalización normal	El proceso ejecuta una llamada al sistema operativo para indicar que ha completado su ejecución
Límite de tiempo excedido	El proceso ha ejecutado más tiempo del especificado en un límite máximo. Existen varias posibilidades para medir dicho tiempo. Estas incluyen el tiempo total utilizado, el tiempo utilizado únicamente en ejecución, y, en el caso de procesos interactivos, la cantidad de tiempo desde que el usuario realizó la última entrada.
Memoria no disponible	El proceso requiere más memoria de la que el sistema puede proporcionar.
Violaciones de frontera	El proceso trata de acceder a una posición de memoria a la cual no tiene acceso permitido.
Error de protección	El proceso trata de usar un recurso, por ejemplo un fichero, al que no tiene permitido acceder, o trata de utilizarlo de una forma no apropiada, por ejemplo, escribiendo en un fichero de sólo lectura.
Error aritmético	El proceso trata de realizar una operación de cálculo no permitida, tal como una división por 0, o trata de almacenar números mayores de los que la representación hardware puede codificar.
Límite de tiempo	El proceso ha esperado más tiempo que el especificado en un valor máximo para que se cumpla un determinado evento.
Fallo de E/S	Se ha producido un error durante una operación de entrada o salida, por ejemplo la imposibilidad de encontrar un fichero, fallo en la lectura o escritura después de un límite máximo de intentos (cuando, por ejemplo, se encuentra un área defectuosa en una cinta), o una operación inválida (la lectura de una impresora en línea).
Instrucción no válida	El proceso intenta ejecutar una instrucción inexistente (habitualmente el resultado de un salto a un área de datos y el intento de ejecutar dichos datos).
Instrucción privilegiada	El proceso intenta utilizar una instrucción reservada al sistema operativo.
Uso inapropiado de datos	Una porción de datos es de tipo erróneo o no se encuentra inicializada.
Intervención del operador por el sistema operativo	Por alguna razón, el operador o el sistema operativo ha finalizado el proceso (por ejemplo, se ha dado una condición de interbloqueo).
Terminación del proceso padre	Cuando un proceso padre termina, el sistema operativo puede automáticamente finalizar todos los procesos hijos descendientes de dicho padre.
Solicitud del proceso padre	Un proceso padre habitualmente tiene autoridad para finalizar sus propios procesos descendientes.

ESTADOS DE LOS PROCESOS

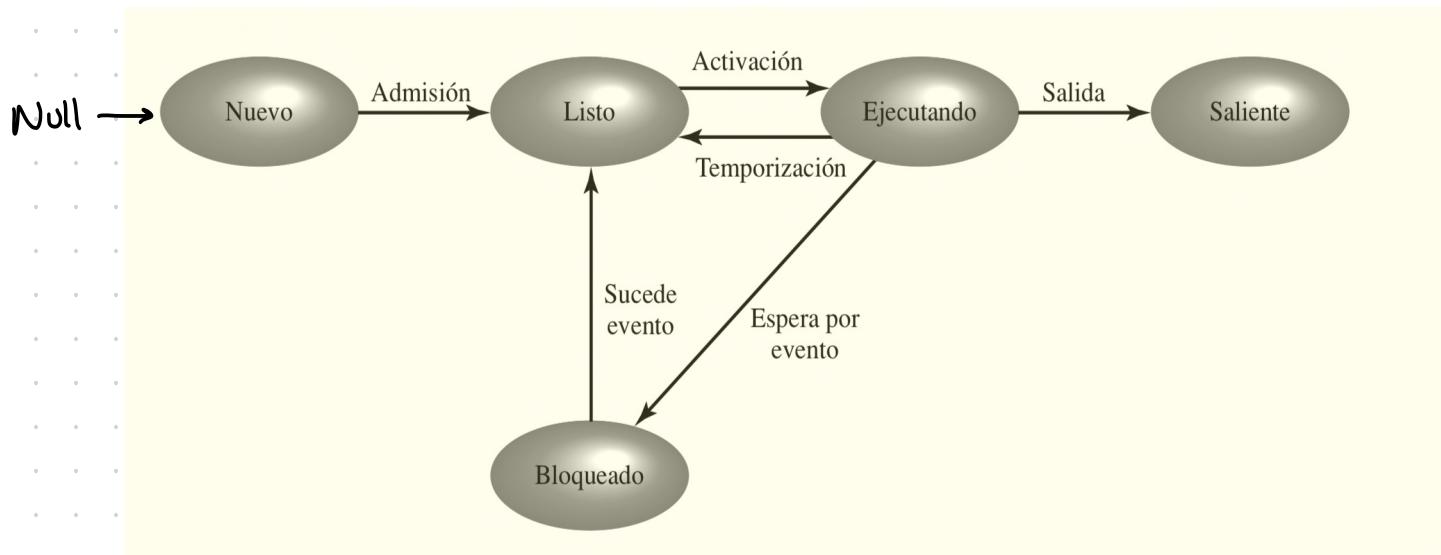
→ MODELO DE DOS ESTADOS 
Ejecutando
No ejecutando = en cola

- Cola = lista enlazada de bloques de datos que representan procesos.
- Proceso interrumpido → cola, si ha finalizado o ha sido abortado → fuera del sist.
- Activador: selecciona nuevo proceso a ejecutar de la cola.

→ MODELO DE 5 ESTADOS

Cola = Lista FIFO (first in first out) ⊕ procesado opera siguiendo estrategia cíclica
 Algunos procesos en estado No ejecutando están listos para ejecutar, mientras que otros = bloqueados, esperando a que se complete una operación E/S, por tanto utilizando una sola cola, el activador no puede seleccionar únicamente los procesos que lleven más tiempo en cola. ⇒ nuevo modelo de estados.

- EJECUTANDO (en ejecución)
- LISTO (se prepara para ejecutar cuando tenga oportunidad).
- BLOQUEADO (no puede ejecutar hasta que se cumpla un evento).
- NUEVO (se acaba de crear y aun no ha sido admitido en Listo por el S.O.)
- FINALIZADO (ha sido liberado de Listo, detenido o abortado por alguna razón).



TRANSICIONES DE ESTADO:

- Null → Nuevo = se crea un nuevo proceso
- Nuevo → Listo = listo para ejecutar
- Listo → Ejecutando = planificador lo selecciona para ejecutar.
- Ejecutando → Saliente = ha finalizado su ejecución o se ha abortado.
- Ejecutando → Listo = ha consumido el tiempo máx de ejecución, el SO lo ha expulsado o ha dejado de utilizar el procesador voluntariamente.
- Ejecutando → Bloqueado = solicita algo por lo que debe esperar.
- Bloqueado → Listo = sucede el evento por el cual estaba esperando.
- Listo → Saliente = Padre termina hijo o termina el padre.
- Bloqueado → Saliente = Listo → Saliente.

ESQUEMA DE DOS COLAS ↗ Listos ↘ Bloqueados

- Cada proceso admitido → Listo
- Cuando termina de utilizar procesador ↗ finalizo ↘ Listo ↘ Bloqueado
- Fin evento → Listo

BLOQUE DE CONTROL DE PROCESOS (PCB)

Estructura de datos que incluye la siguiente información sobre un proceso:

- **de identificación:** (para distinguirlo del resto)
 - Identificador del proceso
 - Identificador del proceso padre
 - Info sobre el usuario
- **Estado del procesador:** valores iniciales de los registros del procesador o su valor en el instante en el que el proceso fue expulsado.
- **Información de control del proceso:** permite gestionar el proceso:
 - Info de planificación y estado:
 - Estado
 - Evento por el que espera
 - Prioridad
 - Info de planificación
 - Descripción de las regiones de memoria asignadas al proceso
 - Recursos asignados (ficheros, puertos, temporizadores, etc.)
 - Punteros para estructurar los procesos en colas o anillos.
 - Comunicación entre procesos.

Contiene info suficiente para interrumpir un proceso y posteriormente restaurar su estado de ejecución como si no hubiera habido interrupción.

CAMBIO DE PROCESO

Programa en ejecución se interrumpe para que el S.O. asigne otro proceso y establecer un turno entre procesos. Se realiza en cualquier instante en el que el SO obtiene el control sobre el proceso actualmente en ejecución. Es resultado de:

- Una excepción.
- Una llamada al sistema.
- Una interrupción.

Esta última puede ser de reloj (determina si ha excedido o no tiempo de ejecución), de E/S (evento por el cual esperaban 1 o + procesos) o un fallo de memoria (referencia a zona externa a la MP)

Mecanismo	Causa	Uso
Interrupción	Externa a la ejecución del proceso actualmente en ejecución.	Reacción ante un evento externo asíncrono.
Trap	Asociada a la ejecución de la instrucción actual.	Manejo de una condición de error o de excepción.
Llamada al sistema	Solicitud explícita.	Llamada a una función del sistema operativo.



Ya puedes sacarte tu B1/B2/C1 de inglés desde casa

CAMBIO DE MODO

Si hay una interrupción, el proceso actúa de la siguiente forma:

- ① Coloca el CP en la dirección de comienzo de la rutina del manejador de instrucciones, habiendo salvado previamente su valor y el del PSW.
- ② Cambia a modo kernel para que el código de tratamiento de la interrupción pueda incluir instrucciones privilegiadas.
- ③ Comienza fase de búsqueda de instrucción y se guardan los registros del procesador en el PCB (si es necesario).
- ④ El manejador de interrupción lleva a cabo su función.
- ⑤ El hardware restaura automáticamente en el procesador la información del PC y PSW previamente salvada.

Un cambio de modo no implica un cambio de contexto \Rightarrow tras rutina de interrupción puede que se reanude con el mismo proceso.

CAMBIO DE CONTEXTO

Proceso en Ejecutándose cambia a otro estado \Rightarrow S.O. debe realizar los cambios sustanciales en su entorno. Pasos:

- ① Se guarda el estado del procesador en el PCB.
- ② Se actualiza el PCB y se cambia el estado del proceso a uno de los otros cuatro.
- ③ Se mueve el PCB a la cola apropiada.
- ④ Se selecciona un nuevo proceso a ejecutar.
- ⑤ Se actualiza el PCB del proceso elegido (a Ejecutando)
- ⑥ Se actualizan las estructuras de datos de gestión de memoria.
- ⑦ Restaurar el estado del procesador al que tenía en el momento en el que el proceso seleccionado salió de Ejecutando por última vez.

MODOS DE EJECUCIÓN

\rightarrow Modo usuario: ejecuta solamente un subconjunto de las instrucciones máquina, quedando prohibidas las demás. También queda restringido el acceso a determinados registros y zonas del mapa de memoria y E/S. Es el modo de ejecución de los programas de usuario para garantizar la seguridad. Este modo no permite ejecutar operaciones E/S, modificar parte del registro de estado, ni los registros de soporte de gestión de memoria.

\rightarrow Modo privilegiado: ejecuta todas las instrucciones sin restricción y tiene permitido el acceso a todos los registros y mapas de direcciones. Es el modo de ejecución del S.O.

El modo de ejecución está determinado por uno o varios bits del registro de estado y sirve para dar soporte al S.O.

HEBRAS O HILOS

Proceso =

{ Propiedad de recursos (incluye un espacio de direcciones virtuales para su manejo). Está formado por programa, datos, pila y otros atributos incluidos en el PCB.

Planificación/ ejecución: sigue una traza a través de uno o más programas. Puede estar intercalada con otros procesos, de manera que un proceso consta de un estado y una prioridad de activación.

Para distinguir dichas características \Rightarrow la ud que se activa = hilo o proceso ligero, mientras que la ud de la propiedad de recursos se suele denominar proceso o tarea.

MULTIHILLO

Capacidad de un sistema operativo de dar soporte a múltiples hilos de ejecución en un solo proceso. El enfoque tradicional se conoce como estrategia monohilo.

Otros S.O. soportan múltiples procesos de usuario, pero sólo un hilo por proceso, mientras que algunos soportan un único proceso con múltiples hilos. Y otros, múltiples procesos y múltiples hilos.

En entornos multihilo:

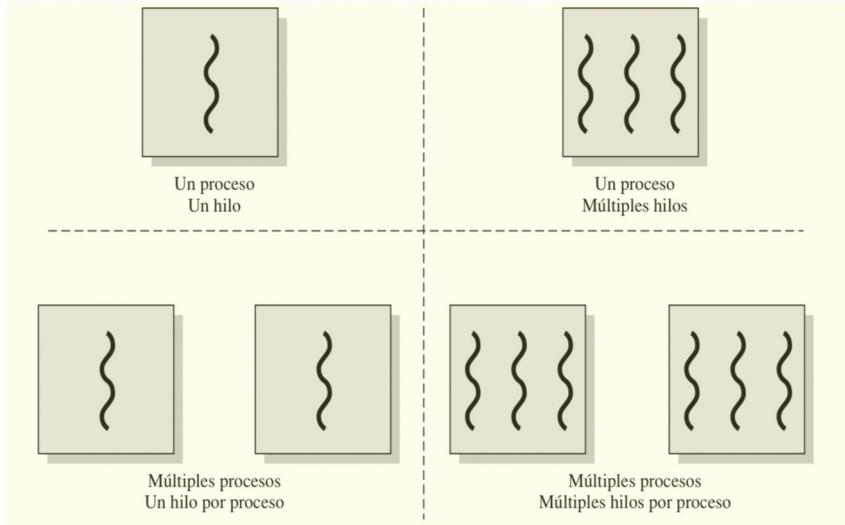
→ Se asocian con procesos el espacio de direcciones virtuales y el acceso protegido a procesadores, otros procesos y recursos E/S.

→ Se asocian con hilos el estado de ejecución (uno por hilo), el contexto que se almacena cuando no está en ejecución, una pila de ejecución un espacio de almacenamiento para variables locales y acceso a la memoria y recursos de su proceso, compartido con todos los hilos de su mismo proceso.

REPRESENTACIÓN PROCESOS:

Monohilo \rightarrow PCB y espacio direcciones + pilas de usuario y núcleo de gestión de llamadas al sistema.

Multihilo \rightarrow un único PCB y espacio de direcciones pero varias pilas para cada hilo y bloque control para cada hilo con los valores de los



registros, la prioridad y otra info relativa al estado del hilo.

BENEFICIOS DE LOS HILOS

- ① Lleva menos tiempo crear un nuevo hilo en un proceso existente que crear un nuevo proceso.
- ② Lleva menos tiempo finalizar un hilo que un proceso.
- ③ Lleva menos tiempo cambiar entre 2 hilos del mismo proceso.
- ④ Mejoran la eficiencia de la comunicación entre programas en ejecución, ya que la comunicación entre procesos requiere la intervención del S.O. mientras que, como los hilos de un mismo proceso comparten memoria y archivos, se pueden comunicar sin necesidad de invocar al Sist. operativo
- ⑤ Ayudan a simplificar la estructura de programas que realizan varias funciones diferentes.
- ⑥ Permiten aprovechar las técnicas de programación concurrente y multiprociamiento simétrico

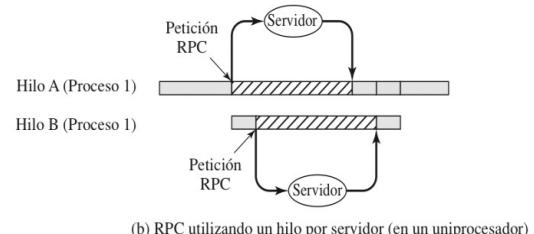
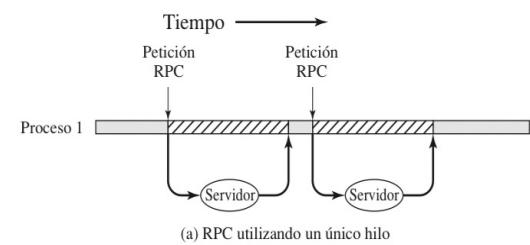
SMP (multiprocesam. simétrico)

Multiples procesadores cada uno con su UC, ALU y registros. Cada proceso tiene acceso a una MP compartida y dispositivos E/S a través de algún mecanismo de interconexión; el bus compartido es común a todos los procesadores. Pueden comunicarse entre sí, a través de la memoria.

La memoria caché es privada para cada procesador y contiene la imagen de una porción de MP, por tanto si se altera una, podría invalidar al resto.

Las claves de diseño de estos sistemas son:

- Procesos o hilos simultáneos concurrentes
- Planificación
- Sincronización
- Gestión de Memoria
- Fiabilidad y tolerancia a fallos.



■■■■ Bloqueado, esperando respuesta RPC
■■■■■ Bloqueado, esperando el procesador, que está en uso por Hilo B
■■■ Ejecutando

Figura 4.3. Llamadas a Procedimiento Remoto (RPC) utilizando hilos.

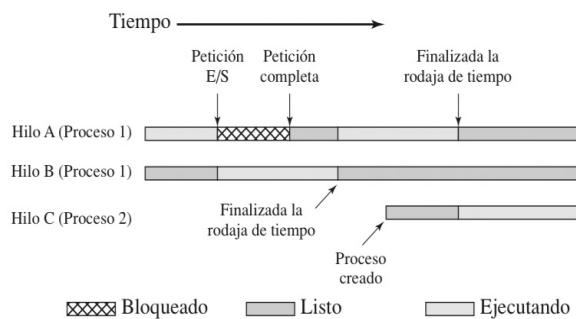


Figura 4.4. Ejemplo multihilo en un uniprocesador.

GESTIÓN DE MEMORIA

REUBICACIÓN

Posibilidad de intercambiar procesos en la MP para maximizar la utilización de la CPU. Una vez llevado un programa a memoria, sería limitante tener que colocarlo siempre en la misma región, podría ser necesario reubicarlo en un área de memoria diferente cuando se trae de nuevo a la memoria. Se debe permitir que los programas se puedan mover en MP, debido al intercambio o swap. El SO debe conocer la info del PCB y la pila de ejecución, así como el punto de entrada que utilizará el proceso para iniciar la ejecución. Como el SO es el encargado de gestionar la memoria, estas direcciones son fáciles de adquirir.

CARGA

Primer paso en la creación de un proceso activo es cargar el programa en MP y crear una imagen del proceso. De la carga se encarga el Cargador, que coloca el módulo de carga en la MP, satisfaciendo el requisito de direccionamiento. Existen 3 técnicas de carga:

→ Carga absoluta: requiere que el módulo de carga se cargue siempre en la misma ubicación de la MP, por lo que todas las referencias a direcciones deben ser específicas o absolutas. La asignación de dichas direcciones las hace un programador o se hacen en tiempo de compilación. El programa no es reubicable.

(a) Cargador

Tiempo de asociación	Función
Tiempo de programación	El programador especifica directamente en el propio programa todas las direcciones físicas reales.
Tiempo de compilación o ensamblado	El programa contiene referencias a direcciones simbólicas y el compilador o ensamblador las convierte a direcciones físicas reales.
Tiempo de carga	El compilador o ensamblador produce direcciones relativas. El cargador las traduce a direcciones absolutas cuando se carga el programa.
Tiempo de ejecución	El programa cargador retiene direcciones relativas. El hardware del procesador las convierte dinámicamente a direcciones absolutas.

(b) Montador

Tiempo de montaje	Función
Tiempo de programación	No se permiten referencias a programas o datos externos. El programador debe colocar en el programa el código fuente de todos los subprogramas que invoque.
Tiempo de compilación o ensamblado	El ensamblador debe traer el código fuente de cada subrutina que se referencia y ensamblarlo como una unidad.
Creación de módulo de carga	Todos los módulos objeto se han ensamblado utilizando direcciones relativas. Estos módulos se enlazan juntos y todas las referencias se restablecen en relación al origen del módulo de carga final.
Tiempo de carga	Las referencias externas no se resuelven hasta que el módulo de carga se carga en memoria principal. En ese momento, los módulos con enlace dinámico referenciados se adjuntan al módulo de carga y el paquete completo se carga en memoria principal o virtual.
Tiempo de ejecución	Las referencias externas no se resuelven hasta que el procesador ejecuta la llamada externa. En ese momento, el proceso se interrumpe y el módulo deseado se enlaza al programa que lo invoca.

Ya puedes sacarte tu B1/B2/C1 de inglés desde casa

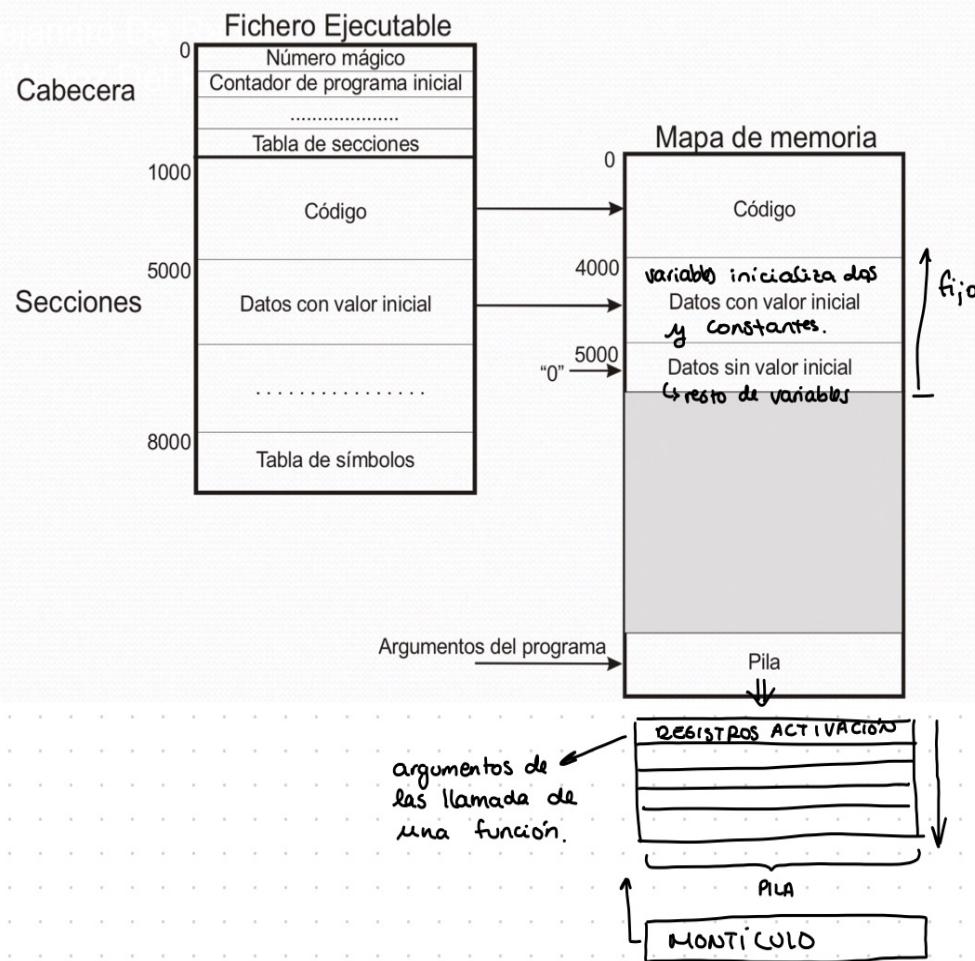


#LinguaskillEnCasa

→ Carga reubicable: Se enlazan las referencias en tiempo de carga, de modo que el módulo de carga pueda ubicarse en cualquier lugar de la MP. Para ello, compilador no produce direcciones de memoria reales, sino relativas a algún punto conocido (ej. inicio del programa). Con todas las referencias en formato relativo es más fácil colocar el módulo de carga simplemente hay que borrar a las direcciones relativas, la dirección de la ubicación donde ha sido cargado.

→ Carga dinámica en tiempo real: Para maximizar la utilización de memoria, es importante poder intercambiar los procesos en diferentes localizaciones en diferentes momentos. Por tanto, un programa una vez cargado puede intercambiarse a disco y memoria en diferentes ubicaciones. Para ello se pospone el cálculo de una dirección absoluta hasta que se necesita en tiempo de ejecución. Hasta que una instrucción no se ejecuta, no se calcula su dirección absoluta. Esto proporciona flexibilidad: un programa se puede interrumpir y posteriormente intercambiarse en una localización diferente.

EJEMPLO MAPA DE MEMORIA



ALGORITMO DE UBICACIÓN

PARTICIONAMIENTO FIJO

Particiones mismo tamaño \Rightarrow ubicación procesos = partición disponible \rightarrow proceso se carga en dicha partición. Todas las particiones mismo tamaño. Si todas se encuentran ocupadas, uno de los procesos se lleva a disco para dejar espacio a uno nuevo.

Particiones diferente tamaño \Rightarrow dos formas asignar procesos

- \rightarrow asignar cada proceso a la partición más pequeña dentro de la cual cabe (necesaria cola de planificación para cada partición) \Rightarrow puede haber colas vacías en un dpt. momento
- \rightarrow Una única cola de planificación para todos los procesos en el momento de cargar se selecciona la partición más pequeña. Si todas ocupadas \Rightarrow swap

Ventajas \rightarrow flexibilidad frente a particiones = tamaño

Desventajas

- \rightarrow n° particiones limita n° procesos activos. (en = tamaño)
- \rightarrow Tamaños preestablecidos = trabajos pequeños no utilizan todo el espacio de las particiones. = FRAGMENTACIÓN INTERNA

PARTICIONAMIENTO DINÁMICO

Longitud y n° variable = proceso MP se le asigna exactamente la memoria que necesita. Inicialmente MP vacía \Rightarrow primeros procesos contiguos \Rightarrow hueco al final de la memoria. Fin proceso \Rightarrow hueco nuevo proceso (más pequeño anterior) \Rightarrow hueco entre procesos.

El método comienza correctamente pero finalmente lleva a una situación en la cual existen muchos huecos pequeños en la memoria, a medida que pasa el tiempo se fragmenta cada vez más y la utilización decrementa \Rightarrow FRAGMENTACIÓN EXTERNA (se fragmenta de forma incremental \neq a lo que ocurre con la interna, descrita anteriormente).

Una técnica para eliminar la fragmentación \Rightarrow COMPACTACIÓN = S.O. desplaza los procesos en memoria de forma que se encuentren contiguos y toda la memoria libre se encuentre unida en un bloque suficientemente grande como para cargar un proceso adicional.

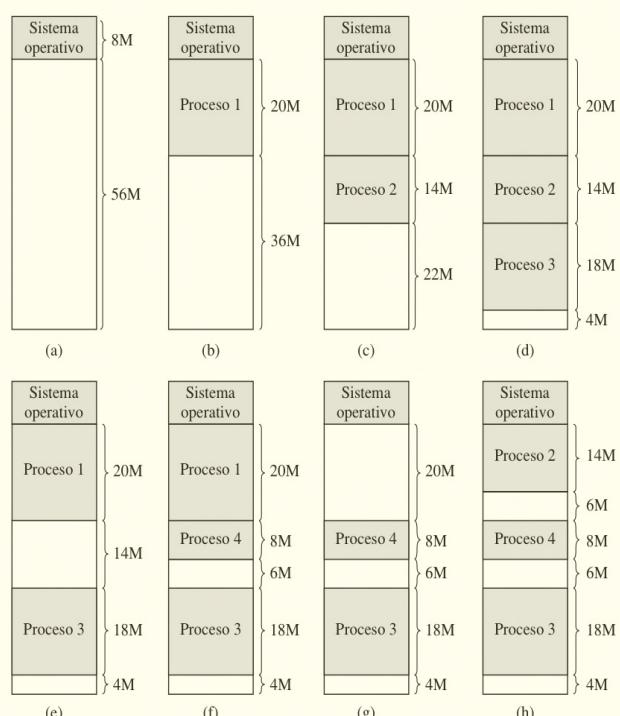


Figura 7.4. El efecto del particionamiento dinámico.

PAGINACION

MP dividida en porciones de tamaño fijo relativamente pequeños (marcos de págs), cada proceso se divide también en porciones pequeñas del mismo tamaño (páginas). De manera que el espacio malgastado debido a frag. interna corresponde sólo a una porción de la última página. ↗ frag. externa. Para evitar fragmentación = reducir tamaño de página \Rightarrow deterioro rendimiento \oplus mayor tabla de páginas.

El SO mantiene una tabla de páginas por cada proceso que muestra la ubicación del marco por cada página del proceso. Cada entrada de la tabla de páginas contiene el nº de marco en MP que contiene la página + bits protección.

$$\text{Dirección lógica} = \text{Nº PÁGINA} + \text{DESPLAZAMIENTO}$$

$$\text{Dirección física} = \text{Nº MARCO} * \text{tamaño pag} + \text{DESPLAZAMIENTO}$$

Cuando la CPU genera una d. lógica, utiliza tabla de páginas para producir una dirección física.

Paginación = similar particionamiento estático con la diferencia de que un proceso puede ocupar más de una partición y dichas particiones pueden no ser contiguas.

Tamaño página potencia de 2 \Rightarrow cada dirección lógica de un programa es idéntica a la relativa \oplus es más sencillo implementar una función que ejecute el hardware para llevar a cabo la traducción de direcciones en tiempo de ejecución.

$$n = \text{nº página}$$

$$m = \text{desplazamiento}$$

$$k = \text{nº marco}$$

dirección física inicial del marco = $k \cdot 2^m$ y dirección física del byte referenciado es $k \cdot 2^m + m$

$$\text{número total págs (o marcos)} = \frac{\text{memoria total (lógica)}}{m}$$

SEGMENTACION

Programa y datos se dividen en segmentos (\neq longitud) \exists longitud máxima es

$$\text{Dirección lógica} = \text{nº segmento} + \text{desplazamiento}$$

similar al particionamiento dinámico \Rightarrow diferencia = programa puede ocupar + de una partición y no necesitan ser particiones contiguas.

Paginación invisible a programador, segmentación visible \oplus utilidad para organizar programas y datos.

No hay relación simple entre direcciones. SO. = tabla de segmentos \oplus lista bloques libres. Entrada tabla = dirección inicial de la MP del segmento, longitud segmento \oplus bits protección

$$\text{dirección física} = \text{dirección física inicial} + \text{desplazamiento}$$

BUFFER DE TRADUCCIÓN ANTICIPADA (TLB)

Referencia a memoria virtual puede causar 2 accesos a memoria física: 1 para q entra de la tabla y otro q datos solicitados. \Rightarrow se duplica el tiempo de acceso a memoria.

Solución: caché hardware de consulta rápida (TLB) \Rightarrow contiene entradas de la tabla usadas + recientemente \Rightarrow dada una dirección virtual, el procesador examina TLB si esta presente la entrada solicitada \Rightarrow se construye la dirección real, si no el procesador utiliza el nº pagina(o segmento) para indexar la tabla de páginas del proceso y examinar la correspondiente entrada. Si bit de presente = 1 página en MP \Rightarrow recupera número marrón y construye dirección real, si bit=0 \Rightarrow página solicitada \notin MP \Rightarrow fallo de página \Rightarrow S.O. carga página necesaria y actualiza tabla.

Entrada TBL incluye nº página y entrada tabla de páginas completa.

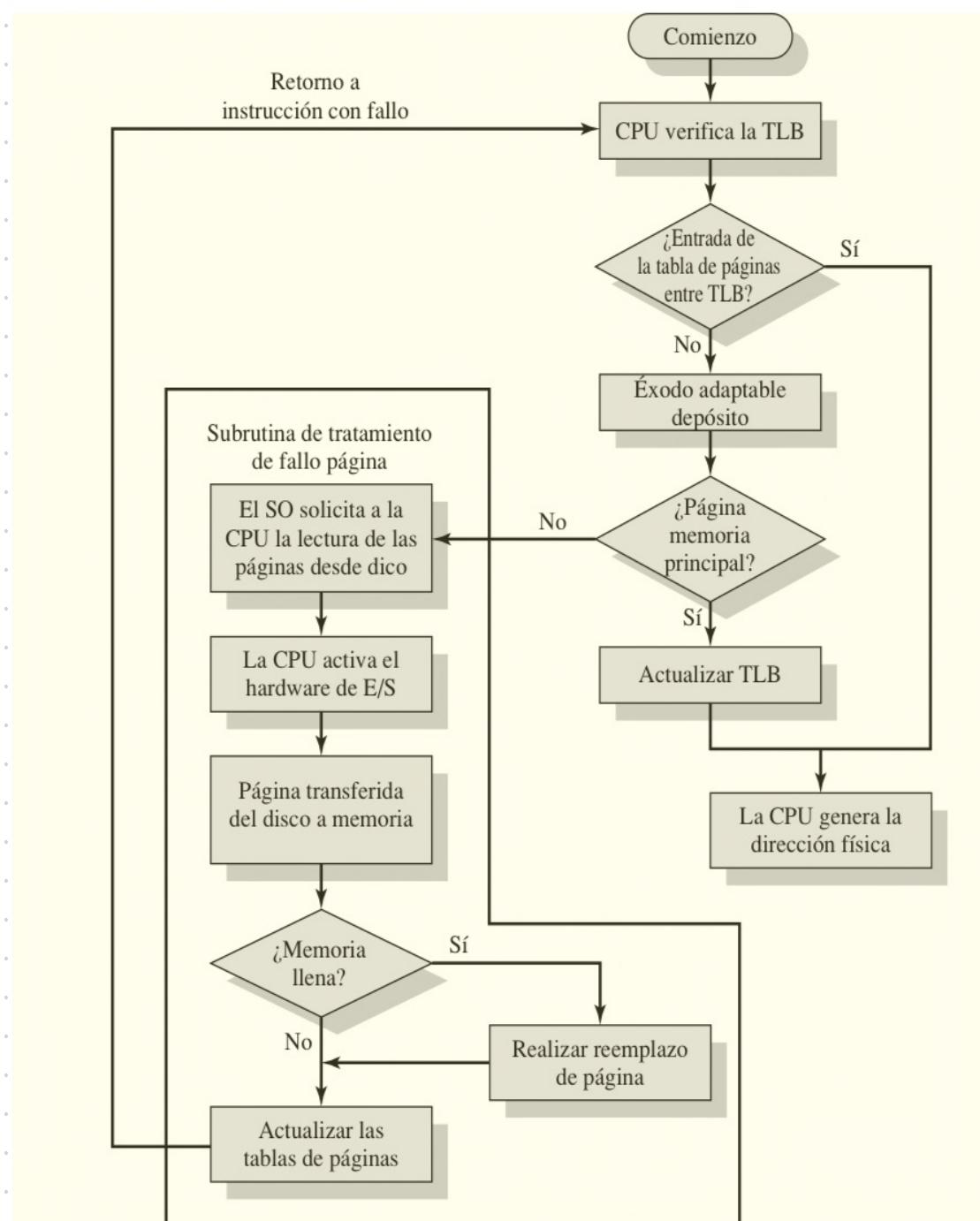


Figura 8.8. Operación de paginación y TLB [FURH87].