



Sistemas Operativos 2021-2022

TEMA 2 Gestión de procesos

- 2.1. Introducción
- 2.2. Descripción de procesos
- 2.3. Ciclo de vida de los procesos
- 2.4. Control de procesos
- 2.5. Algoritmos de planificación a corto plazo
- 2.6. Hilos

2.1.2. Definición de proceso

2.1.3. Imagen del proceso



Introducción

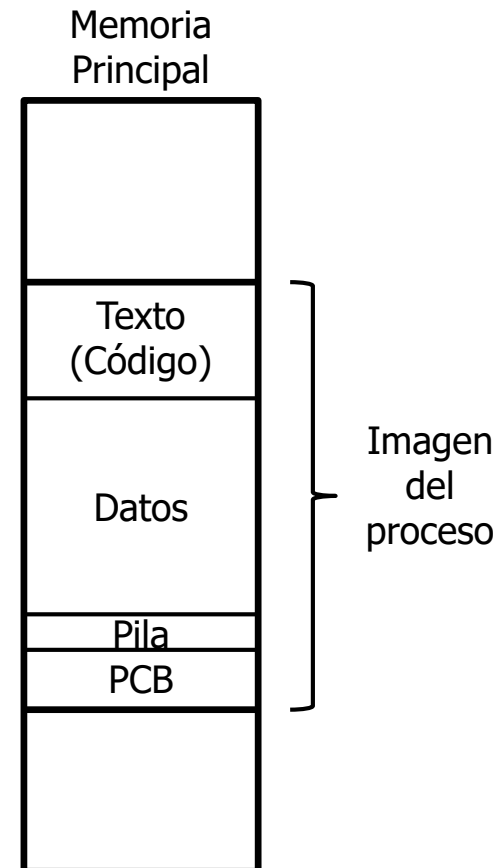
Definición de proceso

- Definición de **proceso**
 - No hay una universalmente aceptada
 - Milenkovic: “*un proceso es una instancia de un programa en ejecución*”
- A partir de ahora nunca diremos **programas** en ejecución sino ***PROCESOS***.

Introducción

Imagen del proceso - *Definición*

- **En Memoria principal:**
 - Texto (código), datos y pila
 - Contexto de ejecución del proceso
 - Información que el SO necesita para administrar el proceso
 - Información que la CPU necesita para ejecutar correctamente el proceso



Introducción

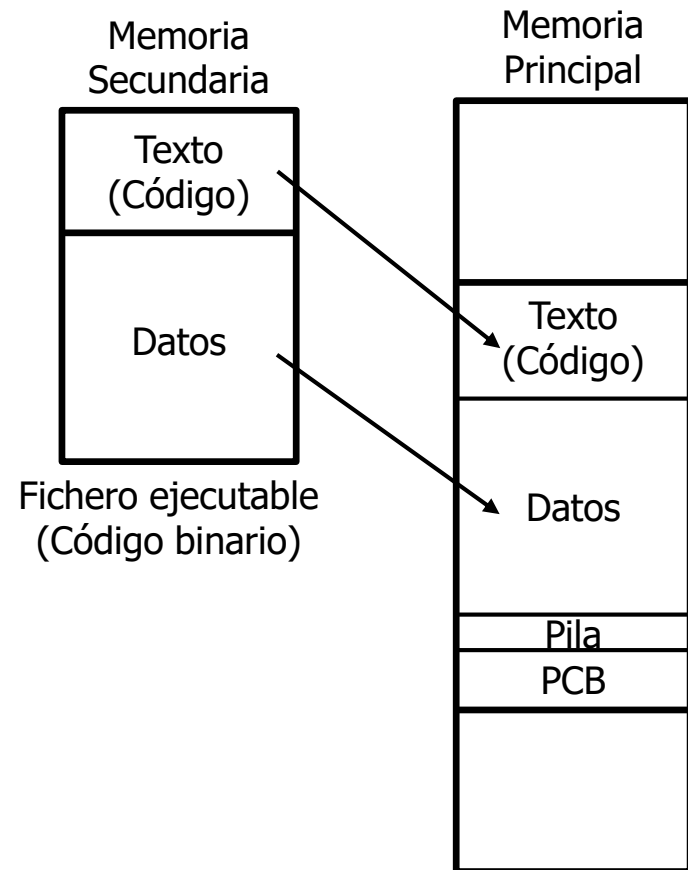
Imagen del proceso - *Creación*

1. Carga de un programa

- Operación que crea parte de la imagen del proceso copiando información almacenada en un fichero ejecutable desde memoria secundaria

2. Mediante clonación

- Copia de otro proceso ya existente (sin cargar ningún programa)



Carga de un programa



Descripción de procesos

Índice

2.2.1. Estructuras de control del sistema operativo

2.2.2. El bloque de control de procesos



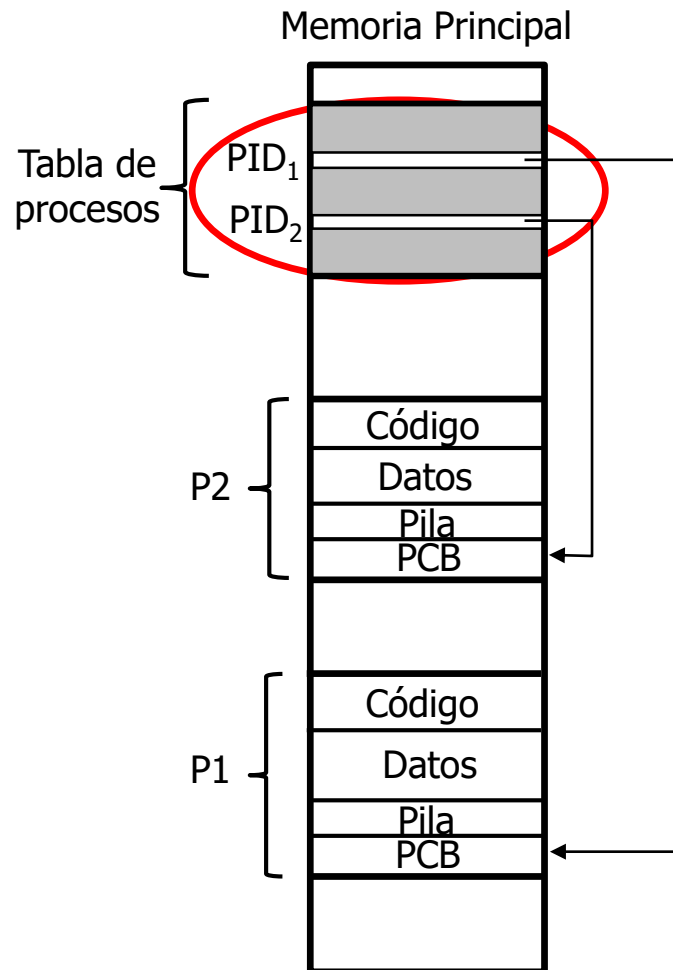
Descripción de procesos

Estructuras de control del SO

- El SO necesita poder acceder a la información de TODOS los procesos en cualquier momento
 - Lo más rápido posible
 - **Tabla de procesos del sistema: Almacena el PCB de TODOS los procesos**
 - **Suele ser** un vector indexado por el identificador del proceso (PID)
 - Puede contener directamente el PCB o APUNTAR al PCB (más frecuente)
 - Reside siempre en memoria principal (al igual que los PCBs)

Descripción de procesos

Estructuras de control del SO





Descripción de procesos

El bloque de control de procesos (PCB)

- Estructura de datos que contiene información diversa y necesaria para que el SO gestione y controle UN proceso
 - Denominada también **PCB** (*Process Control Block*)
 - Un proceso existe, en tanto en cuanto tiene un PCB
- Cada SO le da una implementación diferente
- **Todas las operaciones** que el SO realiza sobre un proceso implican la manipulación de su PCB
- Todos los PCB se organizan en la **tabla de procesos**



Descripción de procesos

El bloque de control de procesos (PCB)

- Elementos básicos de un PCB

- 1. Identificación del proceso**

- Identificador numérico único de proceso (PID)
 - Puede ser usado en otras tablas como referencia
 - También para la identificación del proceso cuando se comunica con otros
 - Si la tabla de procesos se implementa como un vector, puede servir para, usándolo como índice, obtener el PCB del proceso
 - Identificador de su proceso padre (PPID)
 - Sólo si los procesos están autorizados a crear otros procesos
 - Identificador del usuario responsable del mismo (UID)



Descripción de procesos

El bloque de control de procesos

- Elementos básicos de un PCB (continuación)

2. Información de estado del procesador

- **Contenido de los registros del procesador**
- Cuando el proceso abandona la CPU se almacena los valores de sus registros en el PCB del proceso → ***Salvar contexto***
- Cuando el proceso toma la CPU se sobrescriben los valores de sus registros por los obtenidos del PCB del proceso → ***Restaurar contexto***
- **Permite que el intercalado de instrucciones funcione correctamente → *Multiprogramación y Tiempo compartido***



Descripción de procesos

El bloque de control de procesos

- Elementos básicos de un PCB (continuación)

3. Información de control del proceso

- Toda la información adicional necesaria para que el SO controle y coordine los procesos activos
 - Información de planificación y **ESTADO DEL PROCESO**
 - Prioridad, estado, información de planificación, etc.
 - Pertenencia del proceso a estructuras de datos
 - Listas, colas, etc., de (PCBs de) procesos
 - Comunicación entre procesos
 - Privilegios del proceso
 - **Ubicación en memoria**
 - De su código, datos y pila
 - Uso y propiedad de recursos



Ciclo de vida de los procesos

Índice

2.3.1. Introducción

2.3.2. Modelo sencillo

2.3.3. Modelo de cinco estados

2.3.4. Modelo de siete estados



Ciclo de vida de los procesos

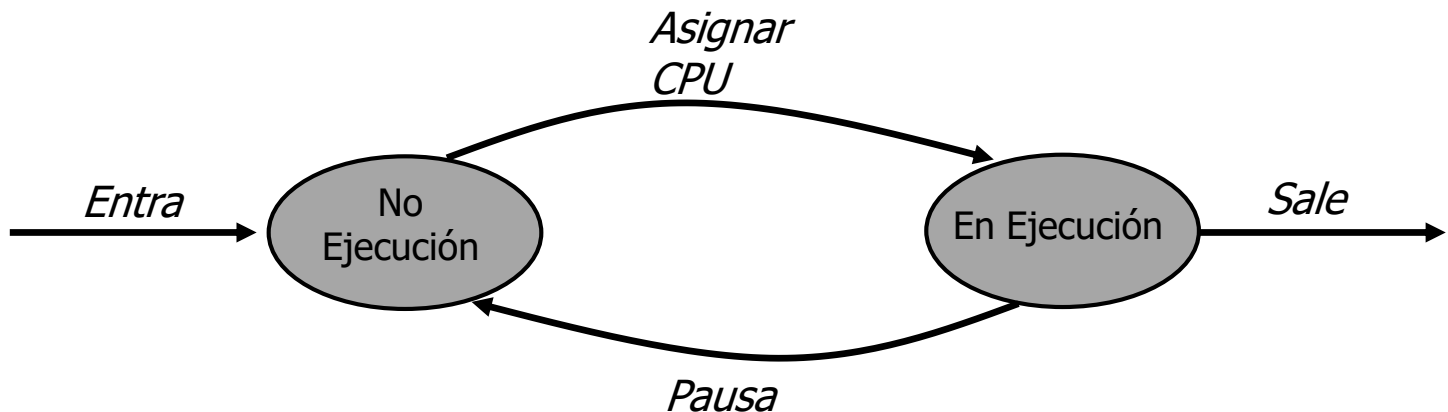
Introducción

- La principal responsabilidad del SO es controlar la ejecución de los procesos
- El primer paso en el diseño de un software que controlará procesos (el SO) consiste en **describir los estados por los que pasa un proceso a lo largo de su vida**

Ciclo de vida de los procesos

Modelo sencillo

- En todo momento, un proceso está ejecutándose en el procesador o no
- Cuando el SO crea un proceso, lo coloca en el estado "No ejecución"
 - En algún momento el procesador quedará libre y el proceso podrá pasar al estado "En ejecución"





Ciclo de vida de los procesos

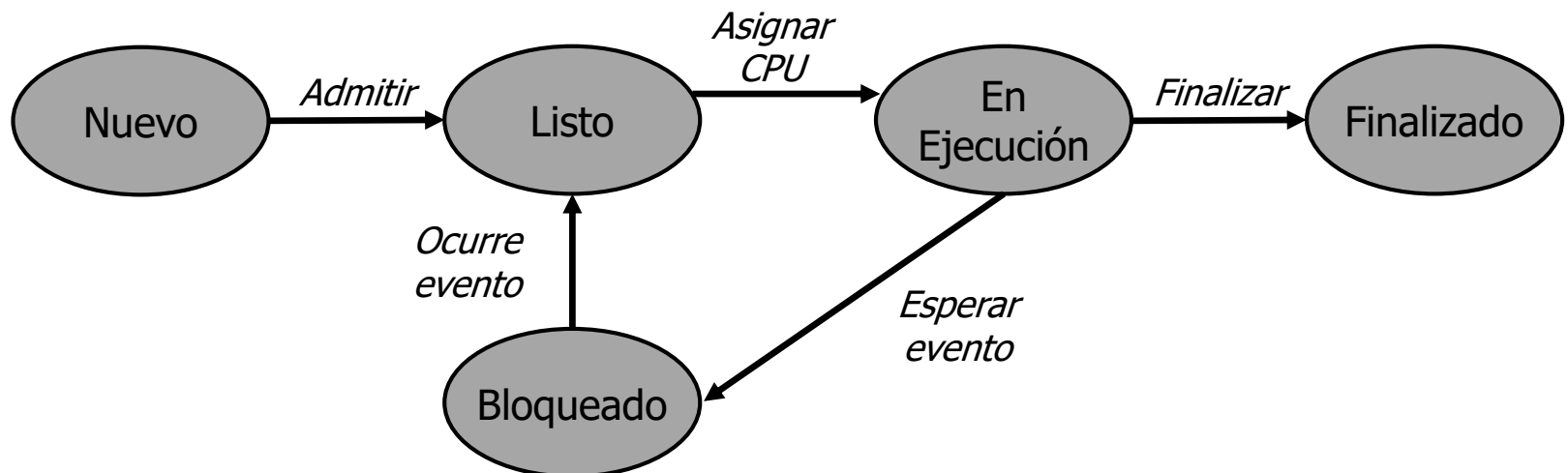
Modelo sencillo

- ¿El modelo sencillo sirve para la multiprogramación?
 - NO SIRVE
- ¿Por qué no sirve?
 - No podemos distinguir entre procesos haciendo E/S y no haciendo E/S
- Solución
 - Dividir “No ejecución” en dos estados
 - “Listo” (NO E/S) y “Bloqueado” (haciendo E/S)
 - Se añaden además otros dos estados (“Nuevo” y “Finalizado”) que resultarán útiles

Ciclo de vida de los procesos

Modelo de cinco estados

- “En ejecución”: el proceso está usando el procesador
- “Listo”: el proceso está preparado para ejecutarse en el momento en que se le dé la oportunidad
- “Bloqueado”: el proceso no puede ejecutarse hasta que ocurre algún evento (por ejemplo, fin de una operación de E/S)

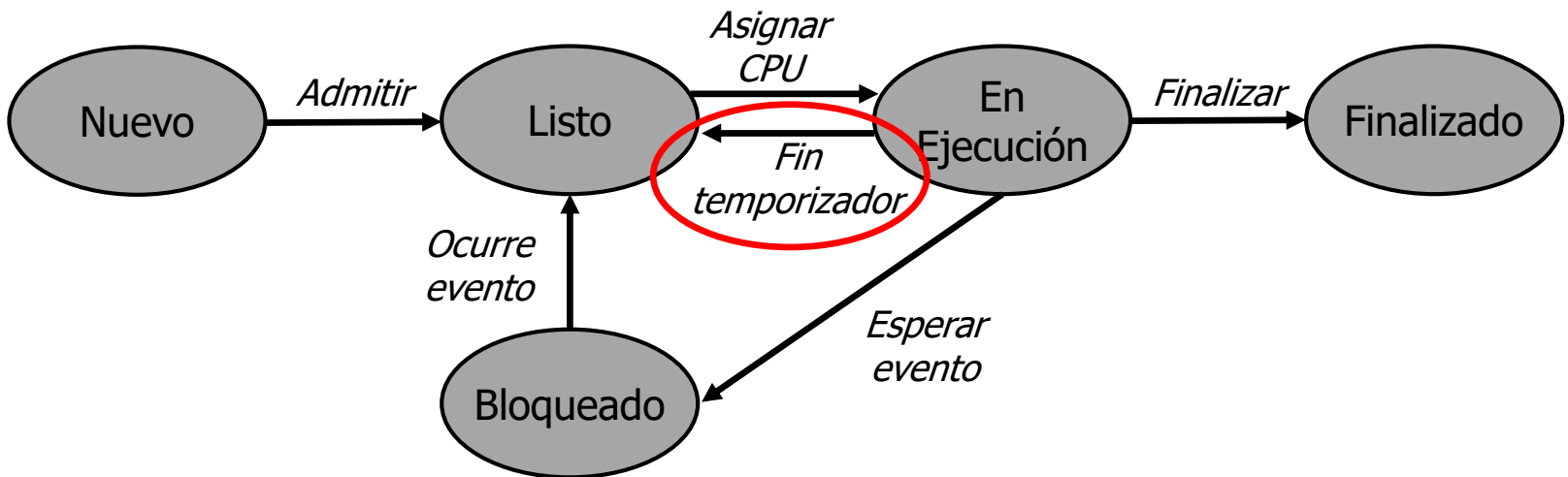


Grafo que implementa la MULTIPROGRAMACIÓN

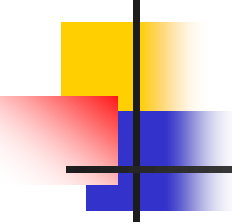
Ciclo de vida de los procesos

Modelo de cinco estados

- El grafo anterior NO SIRVE para tiempo compartido
- Faltaba una transición (marcada con un círculo rojo)



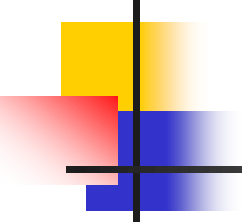
Grafo que implementa el Tiempo Compartido



Ciclo de vida de los procesos

Modelo de cinco estados

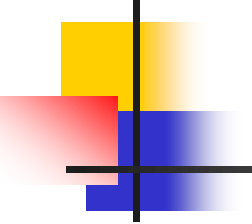
- Estado “Nuevo”
 - El proceso ha sido creado pero no ha sido admitido aún por el SO en el conjunto de procesos ejecutables
 - Habitualmente esto es debido a que el proceso aún no se ha terminado de cargar en memoria
- Estado “Finalizado”
 - El proceso ha sido eliminado por el SO del conjunto de procesos ejecutables (**pero sigue EXISTIENDO**)
 - Se conserva SOLO su PCB
 - Para obtener información de cómo finalizo y estadísticas



Ciclo de vida de los procesos

Modelo de cinco estados

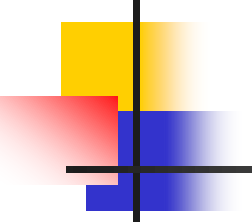
- Un proceso puede terminar por diversas razones
 - Ejecuta una instrucción especial de finalización
 - Invoca una llamada al sistema para indicar que ha terminado (*exit*)
 - Produce algún error de ejecución (excepción)
 - División por cero, instrucción inválida, instrucción privilegiada, acceso ilegal a memoria, etc.
 - Por intervención directa del operador o del SO
 - Su proceso padre lo “mata”
 - **Cada SO utiliza un subconjunto de las razones anteriores**



Ciclo de vida de los procesos

Modelo de cinco estados: Cuestiones de diseño

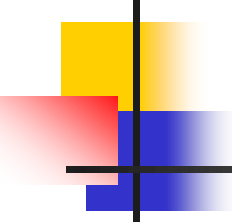
- El SO necesita saber rápidamente los procesos que están en un cierto estado
- Si hay más de uno → Necesito estructura de datos.
- ¿Qué se almacena en la estructura de datos?
 - Normalmente solo el PID → Mayor eficiencia espacial
- Estructuras de datos necesarias:
 - ¿Estructura de datos para procesos en estado “Nuevo”?
 - Sí
 - ¿Estructura de datos para procesos en estado “Listo”?
 - Sí y dependerá de algoritmo del PCP que se verá más adelante
 - ¿Estructura de datos para procesos en estado “Bloqueado”?
 - Sí y necesitaré asociar a cada proceso un evento (¿Diccionarios?)



Ciclo de vida de los procesos

Modelo de cinco estados: Cuestiones de diseño

- Estructuras de datos necesarias (continuación):
 - ¿Estructura de datos para procesos en estado “Finalizado”?
 - Sí
 - ¿Estructura de datos para procesos en estado “En Ejecución”?
 - Sí, si hay multiprocesamiento



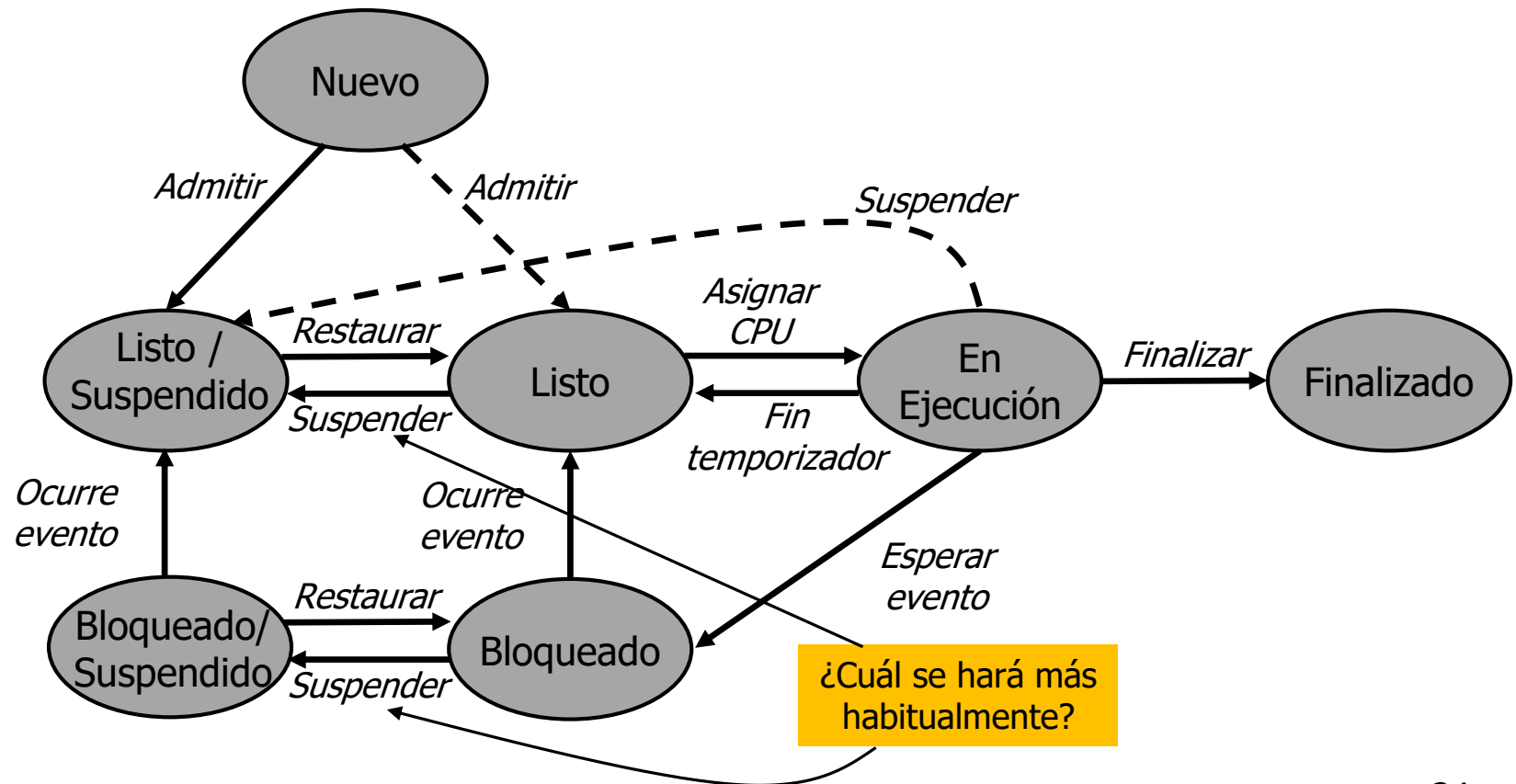
Ciclo de vida de los procesos

Modelo de siete estados

- Los 5 estados principales anteriores aparecen en buena parte de los sistemas operativos
- ¿Son necesarios más estados?
 - Si el sistema operativo se queda sin memoria es posible salvar procesos bloqueados o poco prioritarios a disco → **Suspender un proceso**
 - **Suspender proceso** → Llevar parte del proceso a memoria secundaria (el PCB NO)
 - Para implementar esta característica se necesita un modelo de 7 estados

Ciclo de vida de los procesos

Modelo de siete estados



- 2.4.1. Tipología de los procesos
- 2.4.2. Planificador a largo plazo
- 2.4.3. Planificador a medio plazo
- 2.4.4. Planificador a corto plazo
- 2.4.5. Cambio de proceso
- 2.4.6. Terminación de procesos



Control de procesos

Tipología de los procesos

- Intensivos en CPU
 - Predomina el uso de CPU frente a E/S
 - Pocas ráfagas de CPU, pero de larga duración
- Intensivos en E/S
 - Predomina la E/S frente al uso de CPU
 - Muchas ráfagas de CPU, pero de corta duración



Control de procesos

Tipología de los procesos

Intensivo en CPU

Intensivo en E/S

E/S

CPU



Control de procesos

Planificador a largo plazo

- Funciones

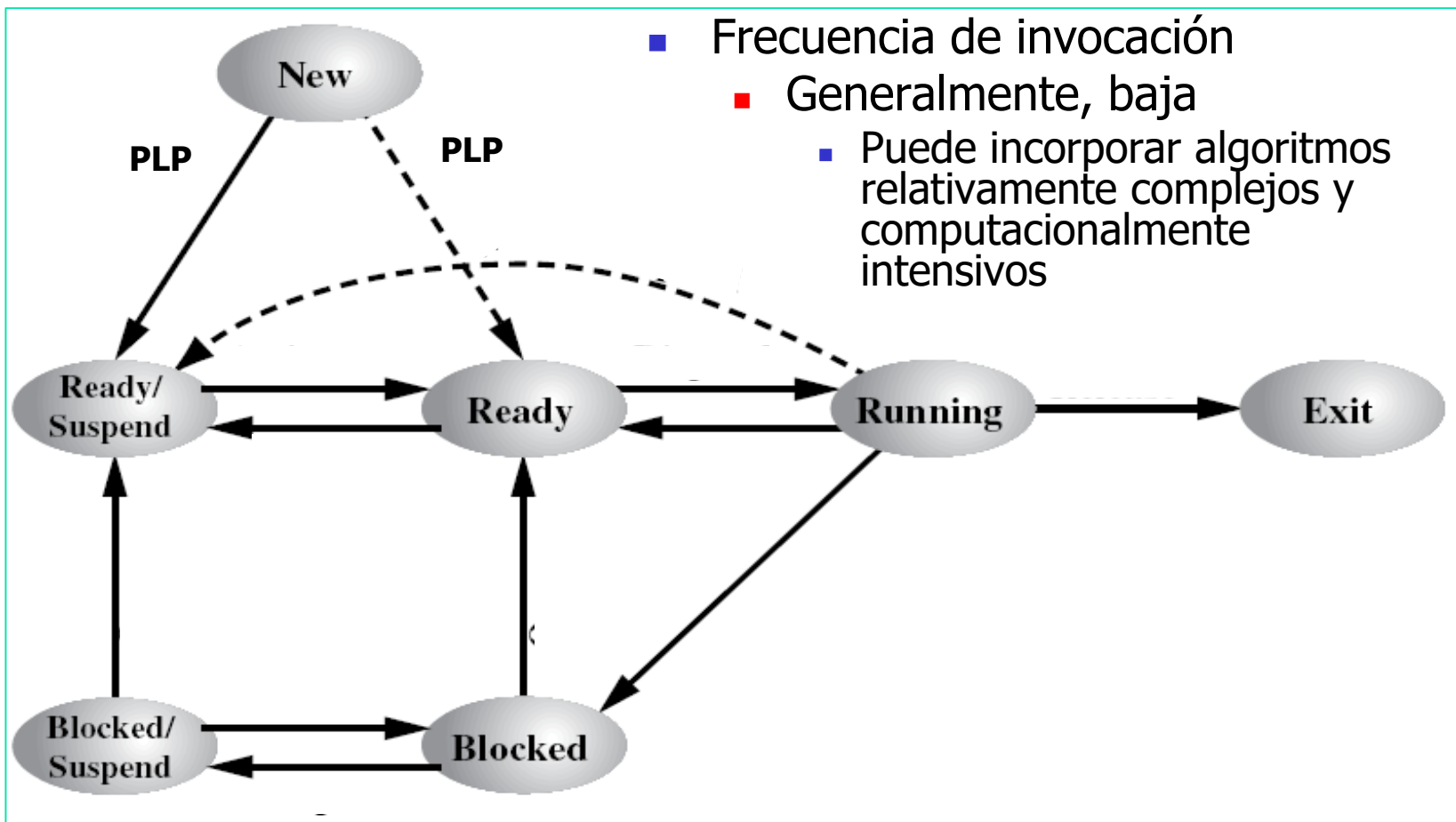
- Admite nuevos trabajos en el sistema, convirtiéndolos en procesos (ver transparencia 22)
 - Los añade primero a la cola de procesos nuevos y posteriormente a la de listos

- Decisiones

- Decide si el SO puede aceptar más procesos
 - Limita el ***grado de multiprogramación*** (nº de procesos en memoria)
- Decide qué trabajos se convierten, definitivamente, en procesos
 - Intenta proporcionar al PCP una mezcla equilibrada de trabajos intensivos en E/S y CPU

Control de procesos

Planificador a largo plazo





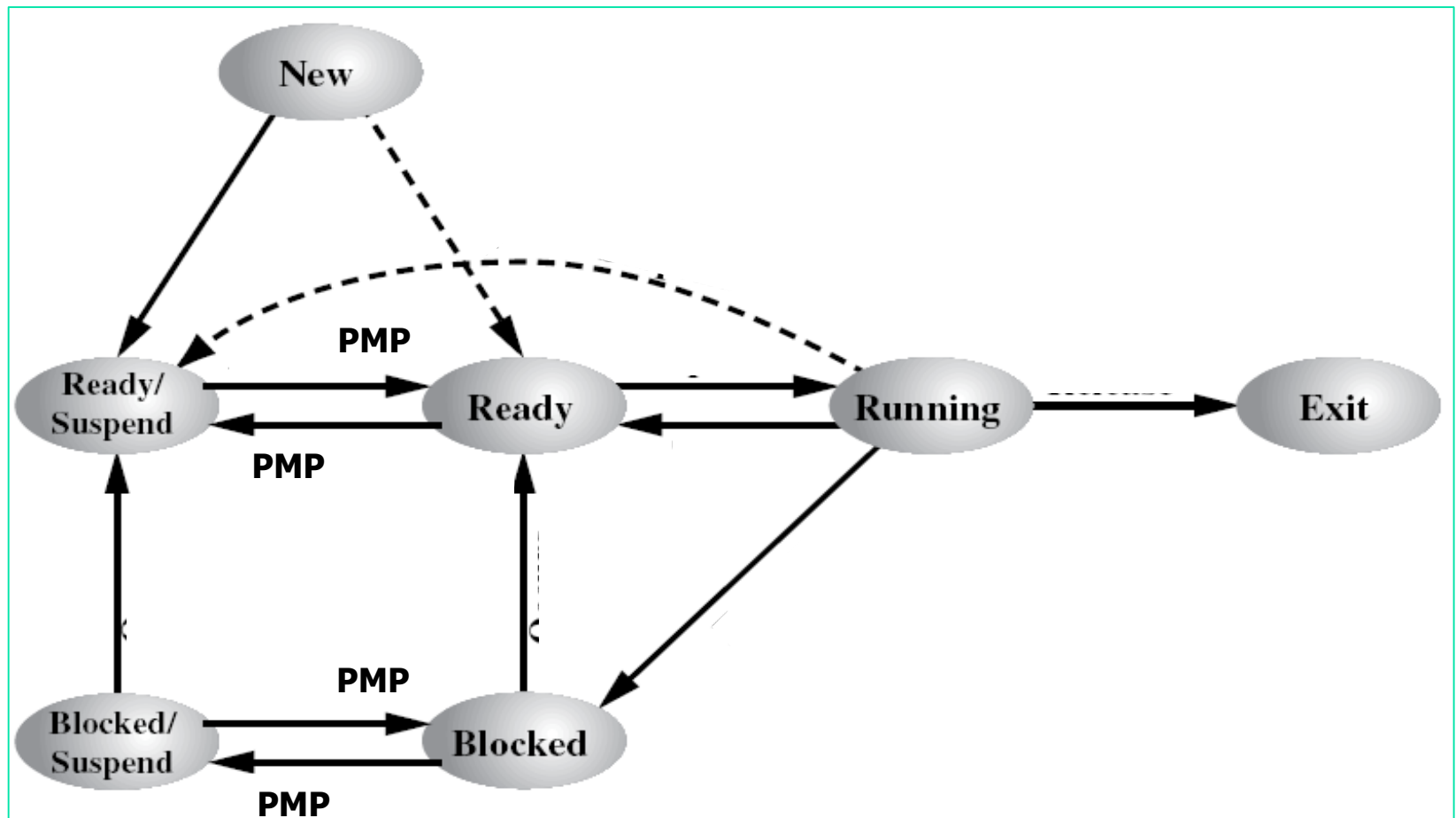
Control de procesos

Planificador a medio plazo

- Funciones
 - Se encarga de suspender y restaurar procesos
 - **Regula el grado de multiprogramación**
- Decisiones
 - Qué procesos de entre los “Listos” y “Bloqueados” (preferentemente éstos últimos) deben ir a memoria secundaria
 - Qué procesos deben volver a memoria principal
 - Cuándo se realizan las operaciones anteriores
- Frecuencia de invocación
 - Cuando queda espacio libre en MP o cuando el nº de procesos listos cae por debajo de un determinado nivel o cuando grado de multiprogramación es muy alto

Control de procesos

Planificador a medio plazo





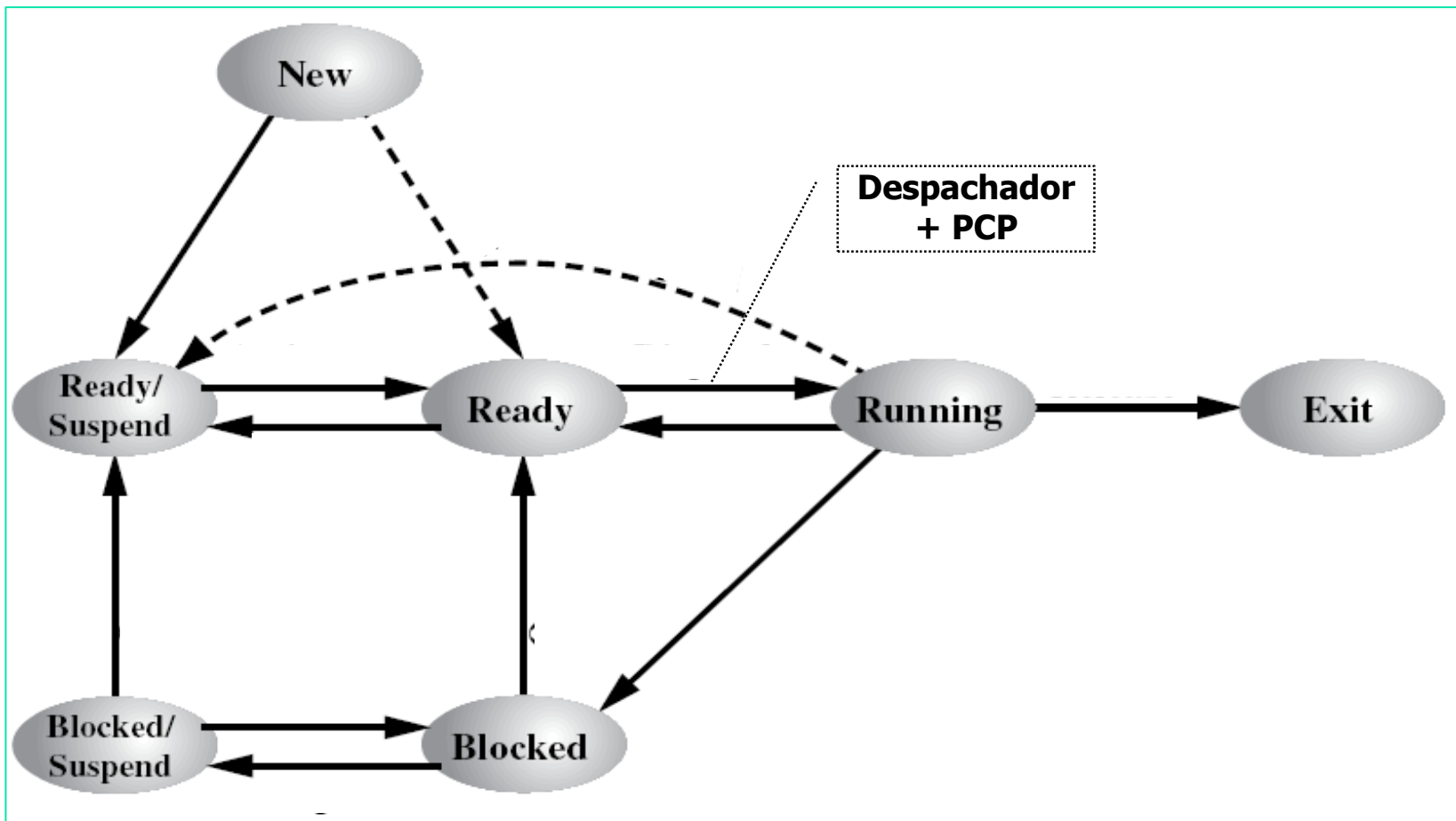
Control de procesos

Planificador a corto plazo

- Funciones
 - Asigna la CPU entre los procesos listos
- Decisiones
 - Qué proceso “Listo” va a tomar la CPU cuando ésta queda libre
 - Todos los procesos deben tener opción de ejecutarse
- Frecuencia de invocación
 - Mucha frecuencia
 - Espera por un evento o E/S
 - Interrupciones de reloj o de E/S
 - Algunas llamadas al sistema operativo
 - Proceso termina normalmente
 - Excepciones

Control de procesos

Planificador a corto plazo





Control de procesos

Cambio de proceso

- En cualquier momento, el proceso en ejecución puede ser interrumpido y el PCP puede decidir colocar en el estado “Ejecutando” a otro proceso
 - **Cambio de proceso**
- El **despachador** es el módulo del SO encargado del cambio de proceso (ver transparencia 11)
 - Actúa sobre el proceso que abandona la CPU
 - Salva su contexto (en su PCB)
 - Lo cambia de estado y lo mueve a la cola apropiada
 - Sobre el proceso que toma el control de la CPU
 - Restaura su contexto (desde su PCB)
 - Lo cambia de estado y lo elimina de la cola de “Listo”



Control de procesos

Cambio de proceso

- A tener en cuenta
 - El despachador no elige el proceso que va a tomar el control de la CPU
 - Lo hará el **planificador a corto plazo**
 - **Sobrecarga**
 - Dado que es consumo de tiempo de CPU por el SO
 - Es costoso por ser muy frecuente
 - Depende también de la velocidad de la memoria, el número de registros a copiar y de la existencia o no de instrucciones especiales para copiar todos los registros



Algoritmos de planificación a corto plazo

Índice

2.5.0. Observaciones

2.5.1. Parámetros estadísticos

2.5.2. FIFO

2.5.3. Prioridad estática sin requisamiento

2.5.4. Prioridad estática con requisamiento

2.5.5. Round-Robin + FIFO

2.5.6. Colas multinivel SIN realimentación

2.5.7. Colas multinivel CON realimentación



Algoritmos de planificación a corto plazo

Observaciones

- Para todos los algoritmos que siguen, se hacen las siguientes simplificaciones
 - Tiempo de admisión = 0
 - No hay suspensión de procesos
 - Tiempo de cambio de proceso = 0
 - Las operaciones de entrada/salida no interfieren entre ellas
 - Se utilizará como criterio definitivo de desempate (si no se indica otro) el menor valor de identificador de proceso (menor PID)



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- Además de realizar la planificación, se calcularán una serie de parámetros estadísticos.
- Su finalidad es comparar objetivamente unos algoritmos con otros y ver quien es mejor o peor para ciertas situaciones.
- Uno de estos parámetros estadísticos se considera idóneo para comparar algoritmos entre sí
 - ***Tiempo de retorno normalizado medio***



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- ***Instante de llegada (t_i)***
 - El instante de tiempo en que el proceso se crea (llega al sistema)
- ***Tiempo de servicio (t_s)***
 - El **tiempo mínimo** que necesita un proceso para ejecutarse.
 - En nuestro caso se corresponde con la suma de todas sus ráfagas de CPU y de E/S
- ***Tiempo de Fin (t_f)***
 - El tiempo en que el proceso pasa a estado "FINALIZADO"
 - $t_f = t_i + t_r$



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- ***Tiempo de retorno (t_r)***

- El tiempo total en que un proceso ha estado en el sistema (vida del proceso)
- $t_r = t_f - t_i$ ó $t_r = t_s + t_e$

- ***Tiempo de espera (t_e)***

- El tiempo total en que un proceso está en estado "LISTO" (esperando)
- $t_e = t_r - t_s$
- Entre más alto peor se ha comportado el algoritmo para el proceso
- Cuando es cero ($t_r = t_s$) se da la situación ideal



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- ***Tiempo de respuesta (t_{resp})***
 - El tiempo que transcurre desde que el proceso llega al sistema (t_i) hasta que usa POR PRIMERA VEZ la CPU
 - Entre más alto peor
 - Este parámetro es el que pretende optimizar el tiempo compartido



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- ***Tiempo de retorno normalizado (t_{rn})***
 - Establece la proporción entre lo que ha tardado el proceso en ejecutarse y el tiempo ideal en que se debería haber ejecutado
 - $t_{rn} = t_r / t_s$ (se cumple que $t_{rn} \geq 1$)
 - Si por ejemplo un proceso tiene $trn = 1.23$ significa que el algoritmo ha hecho que el proceso tarde un 23% más de lo ideal (23% más que si se hubiera ejecutado él solo)
 - Al estar normalizado se puede comparar el comportamiento de un proceso concreto con otros algoritmos



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- ***Tiempos medios***

- A todos los parámetros anteriores se les puede calcular la media → Sumar todos los tiempos de cada proceso y dividir por el número de procesos
- **El tiempo de retorno normalizado medio (trn_m)** permite comparar grupos de procesos con distintos algoritmos de planificación.



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- ***Porcentaje de uso/ocupación/actividad de CPU***
(%CPU_{activa} ó %CPU_{ocupada})
 - Porcentaje de tiempo en que la CPU ha estado en uso (algún proceso la estaba usando)
 - **%CPU_{activa} = (t_{uso} / T_{total}) * 100**
 - Donde **t_{uso}** es el tiempo total de uso de CPU y **T_{total}** es el tiempo total de la planificación (t_f del último proceso en finalizar)
 - Este es el parámetro que intenta MAXIMIZAR la multiprogramación
 - Este parámetro es lógicamente complementario al siguiente (**%CPU_{activa} = 100 - %CPU_{ociosa}**)



Algoritmos de planificación a corto plazo

Parámetros estadísticos

- **Porcentaje de ociosidad/inactividad de CPU ($\%CPU_{inactiva}$ ó $\%CPU_{libre}$ ó $\%CPU_{ociosa}$)**
 - Porcentaje de tiempo en que la CPU ha estado libre
 - Este parámetro es lógicamente complementario al anterior ($\%CPU_{ociosa} = 100 - \%CPU_{activa}$)
 - $\%CPU_{ociosa} = (t_{libre} / T_{total}) * 100$
 - Donde t_{libre} es el tiempo total en que la CPU estuvo libre y T_{total} es el tiempo total de la planificación (t_f del último proceso en finalizar)
 - Este es el parámetro que intenta MINIMIZAR la multiprogramación



Algoritmos de planificación a corto plazo

FIFO

- Ordena la cola de procesos listos (PCB) de acuerdo al instante en que los procesos pasan a dicho estado
- Algoritmo
 - Consiste en escoger para ejecutar al proceso de la cabecera de la cola, eliminando su PCB de la misma
 - Sin requisamiento (política no apropiativa)
- Implementación
 - La cola de "Listos" se implementa como una cola FIFO

Algoritmos de planificación a corto plazo

FIFO

- **Ejemplo:** Proceso (P), Instante Llegada (Ill), Tiempo Servicio (Ts), Tiempo Finalización (Tf), Tiempo Retorno (Tr), Tiempo Retorno Normalizado (Trn), Tiempo Espera (Te)

P	I.LI	CPU-(E/S)	Ts	Tf	Tr	Trn	Te
1	0	4,(1),8,(1),1	15	34	?	?	?
2	2	1,(5),3,(10),1	20	44	?	?	?
3	4	2,(2),5,(3),1	13	43	?	?	?
4	6	10,(1),8	19	42	?	?	?

Tpo. Medio de espera = 21



Algoritmos de planificación a corto plazo

Prioridades estáticas sin requisamiento

- Asocia a cada proceso un valor entero que no cambia a lo largo de su vida y que representa su prioridad (generalmente a menor valor mayor prioridad)
- La CPU se asigna al proceso con mayor prioridad y sin requisamientos (política no apropiativa)
 - Si hay empate se puede aplicar algún otro algoritmo
- Implementación
 - La cola de “Listos” se implementa como una lista ordenada por las prioridades de los procesos



Algoritmos de planificación a corto plazo

Prioridades estáticas sin requisamiento

- **Ejemplo:** Proceso (P), Instante Llegada (Ill), Tiempo Servicio (Ts), Tiempo Finalización (Tf), Tiempo Retorno (Tr), Tiempo Retorno Normalizado (Trn), Tiempo Espera (Te)

P	I.LI	Pri	CPU-(E/S)	Ts	Tf	Tr	Trn	Te
1	0	4	4,(1),8,(1),1	15	35	?	?	?
2	2	1	1,(5),3,(10),1	20	38	?	?	?
3	4	8	2,(2),5,(3),1	13	48	?	?	?
4	6	-3	10,(1),8	19	34	?	?	?

Tiempo medio de espera = 19



Algoritmos de planificación a corto plazo

Prioridades estáticas con requisamiento

- Asocia a cada proceso un valor entero que representa su prioridad (generalmente a menor valor mayor prioridad)
 - Este valor no cambia a lo largo de la vida del proceso
- La CPU se asigna al proceso con mayor prioridad y con requisamiento (versión apropiativa del anterior)
 - Llega un proceso a “LISTO” **más prioritario** que el que está en ejecución
 - Se requisa la CPU al proceso en ejecución



Algoritmos de planificación a corto plazo

Prioridades estáticas con requisamiento

- **Ejemplo:** Proceso (P), Instante Llegada (I_{ll}), Tiempo Servicio (T_s), Tiempo Finalización (T_f), Tiempo Retorno (T_r), Tiempo Retorno Normalizado (T_{rn}), Tiempo Espera (T_e)

P	I.LI	Pri	CPU-(E/S)	T _s	T _f	T _r	T _{rn}	T _e
1	0	4	4,(1),8,(1),1	15	37	?	?	?
2	2	1	1,(5),3,(10),1	20	38	?	?	?
3	4	8	2,(2),5,(3),1	13	47	?	?	?
4	6	-3	10,(1),8	19	25	?	?	?

Tiempo medio de espera = 17



Algoritmos de planificación a corto plazo

Round-Robin+FIFO

- Diseñado especialmente para sistemas de tiempo compartido
- División tiempo CPU en unidades de tiempo llamados **quantum** (cuantos)
- Se utiliza junto con otra política (auxiliar) de planificación para decidir qué proceso toma la CPU
 - Se suele utilizar la política FIFO
- Es siempre una política apropiativa
 - Cuando se produce el fin de quantum
 - Se requisa la CPU al proceso en ejecución y se elige a otro de la cola de "Listos" según política auxiliar



Algoritmos de planificación a corto plazo

Round-Robin+FIFO

- **Ejemplo:** Proceso (P), Instante Llegada (Il), Tiempo Servicio (Ts), Tiempo Finalización (Tf), Tiempo Retorno (Tr), Tiempo Retorno Normalizado (Trn), Tiempo Espera (Te)
- Cuanto= 3 unidades de tiempo

P	I.LI	CPU-(E/S)	Ts	Tf	Tr	Trn	Te
1	0	4,(1),8,(1),1	15	34	?	?	?
2	2	1,(5),3,(10),1	20	33	?	?	?
3	4	2,(2),5,(3),1	13	35	?	?	?
4	6	10,(1),8	19	45	?	?	?

Tiempo medio de espera = 17



Algoritmos de planificación a corto plazo

Colas multinivel sin realimentación

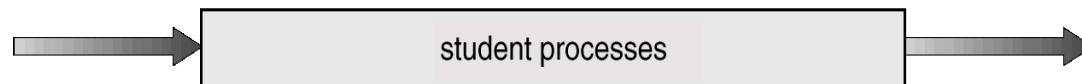
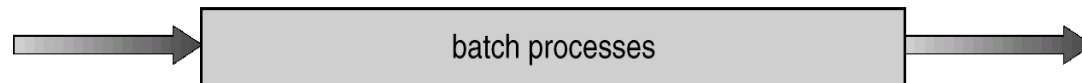
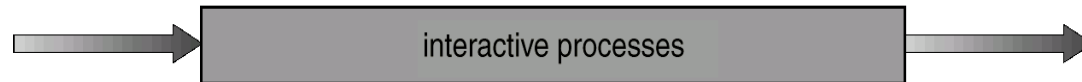
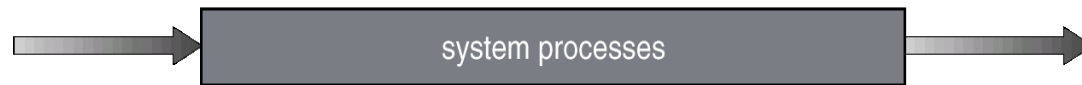
- Los procesos son fácilmente clasificables en grupos diferentes
 - Procesos del sistema, por lotes, interactivos
 - Tienen diferentes requerimientos
 - Por tanto, distintas necesidades de planificación
- Implementación
 - Particiona la cola de “Listos” en varias colas separadas
 - Cada cola tiene asociada una prioridad basada en la importancia de los procesos que están en ella (sistema, por lotes, interactivos, etc.)
 - Ejemplo: Los procesos interactivos deberían tener mayor prioridad que los procesos por lotes
 - **Los procesos no se pueden mover entre colas**



Algoritmos de planificación a corto plazo

Colas multinivel sin realimentación

highest priority



lowest priority



Algoritmos de planificación a corto plazo

Colas multinivel sin realimentación

- Planificación en cada cola
 - **Cada cola tiene su propio algoritmo de planificación**
- Planificación entre colas (dos posibilidades)
 - A. Prioridades con requisamiento
 - Cada cola tiene prioridad absoluta sobre las colas de niveles más bajos de prioridad.
 - Mientras haya procesos en colas de mayor prioridad NO se pueden planificar los de colas de menor prioridad
 - Si mientras se ejecuta un proceso de una cola llega un proceso a una de mayor prioridad se requisa la CPU y se planifica el proceso recién llegado



Algoritmos de planificación a corto plazo

Colas multinivel sin realimentación

- (...más planificación entre colas)
 - B. Tiempo compartido entre colas
 - Cada cola recibe cierta porción de tiempo de CPU
 - Se da más a las colas con procesos de mayor importancia
 - Ejemplo:
 - 45% para procesos del sistema
 - 35% para procesos interactivos
 - 20% para procesos por lotes
- EN LOS EJERCICIOS USAREMOS SIEMPRE LA OPCIÓN A, pero la opción B es más sensata



Algoritmos de planificación a corto plazo

Colas multinivel sin realimentación

■ Ejemplo

1. Sistema: RR con cuanto 5 (P1): cola de máxima prioridad
2. Interactivo: Round Robin con cuanto 3 (P2, P3)
3. FIFO (P4, P5): cola de mínima prioridad

P	I.LI	CPU-(E/S)	Ts	Tf	Tr	Trn	Te
1	0	4,(1),8,(1),1	15	15	15	1	0
2	2	1,(5),3,(10),1	20	29	27	1.35	7
3	4	2,(2),5,(3),1	13	30	26	2	13
4	6	10,(1),8	19	51	45	2.37	26
5	8	2,(8),5,(2),1	18	52	44	2.44	26



Algoritmos de planificación a corto plazo

Colas multinivel sin realimentación

- Ventajas
 - Gestión adecuada de cada proceso en función de su clasificación
- Inconvenientes
 - Inflexible
 - Los procesos no pueden cambiar de cola
 - Sería interesante que los procesos intensivos en CPU pasasen a colas menos prioritarias
 - Peligro de inanición de procesos de colas poco prioritarias



Algoritmos de planificación a corto plazo

Colas multinivel con realimentación

- Modificación del anterior **permitiendo que los procesos se muevan entre las colas**
 - Realimentación
- Objetivo
 - Separar los procesos según su tipología
 - Procesos intensivos en E/S quedan en colas más prioritarias
 - Procesos intensivos en CPU en colas menos prioritarias



Algoritmos de planificación a corto plazo

Colas multinivel con realimentación

- Implementación
 - Particiona la cola de “Listos” en varias
 - Cada cola tiene asociada una prioridad (mayor prioridad a colas más altas)
 - **Todos los procesos entran por la cola de mayor prioridad** la primera vez que llegan al estado “Listo”
 - Cada cola tiene su propio algoritmo de planificación
 - Generalmente RR para colas más prioritarias, con un quantum más pequeño cuanto más prioritaria sea, y FIFO en colas poco prioritarias
 - La planificación entre colas es requisable con prioridades (ver opción A política anterior)



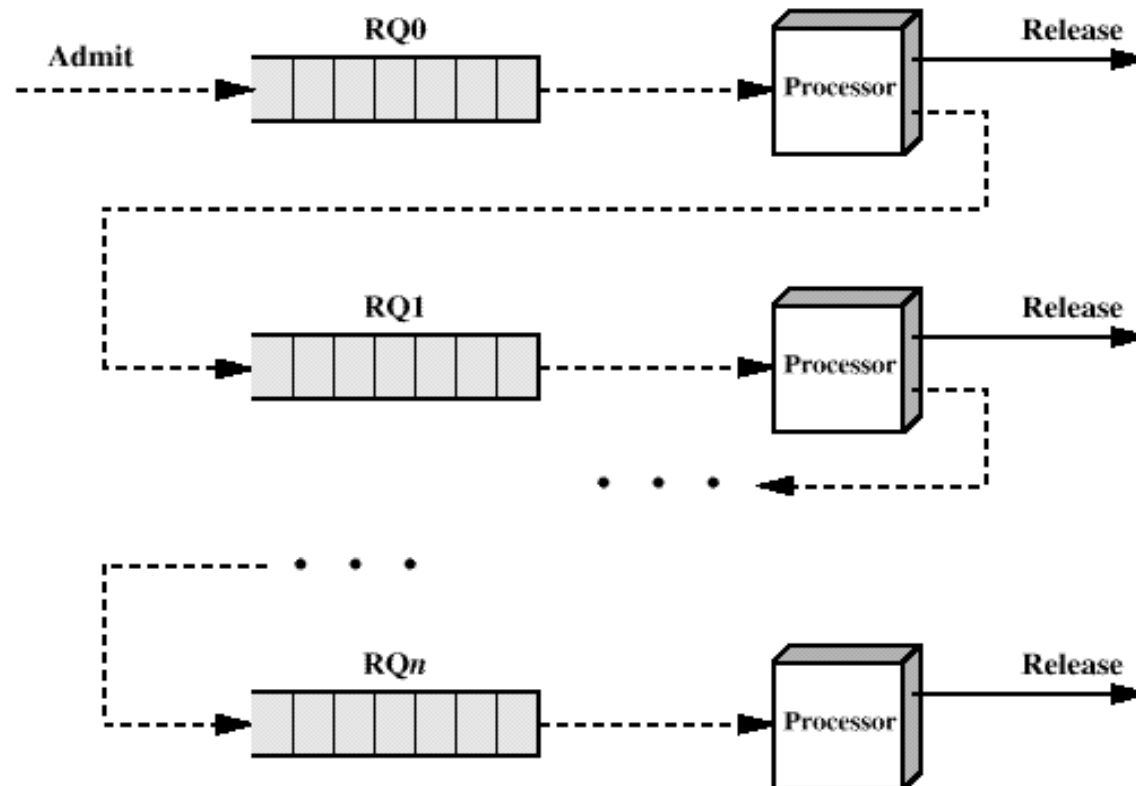
Algoritmos de planificación a corto plazo

Colas multinivel con realimentación

- Implementación (*continuación*)
 - Movimiento entre colas
 - Debe existir un criterio para decidir cuándo un proceso baja a la siguiente cola de menor prioridad
 - Por ejemplo cuando consuma un quantum completo →
Ráfaga CPU de $P \geq 1$ quantum de la cola actual
 - Si la condición no se cumple, el proceso que abandona la CPU volverá a la misma cola en que estuvo la última ocasión que visitó el estado "LISTO"

Algoritmos de planificación a corto plazo

Colas multinivel con realimentación





Algoritmos de planificación a corto plazo

Colas multinivel con realimentación

- **Ejemplo (con 3 colas)**

1. RR con cuanto 3
2. RR con cuanto 5
3. FIFO

P	I.LI	CPU-(E/S)	Ts	Tf	Tr	Trn	Te
1	0	4,(1),8,(1),1	15	52	52	3.47	37
2	2	1,(5),3,(10),1	20	38	36	1.8	16
3	4	2,(2),5,(3),1	13	39	35	2.69	22
4	6	10,(1),8	19	51	45	2.37	26
5	8	2,(8),5,(2),1	18	45	37	2.06	19



Algoritmos de planificación a corto plazo

Colas multinivel con realimentación

- Inconvenientes
 - Inanición de procesos intensivos en CPU
 - Se puede solucionar con una técnica de envejecimiento de los procesos
 - Procesos que lleven mucho tiempo en colas de poca prioridad se promocionan a colas de mayor prioridad

- 2.6.1. Introducción
- 2.6.2. Composición en memoria principal
- 2.6.3. Creación de procesos e hilos
- 2.6.4. Gestión de procesos e hilos
- 2.6.5. Finalización de procesos e hilos
- 2.6.6. Beneficios de los hilos
- 2.6.7. Ejemplos de uso



Hilos (*threads*)

Introducción

- Hasta ahora los procesos tenían 2 características
 1. Propietarios de recursos
 - Ficheros abiertos, dispositivos, etc
 2. Unidad mínima de planificación/ejecución
- En los sistemas operativos actuales esas dos características se tratan independientemente
 - Proceso
 1. Unidad de propiedad de recursos
 - Hilo (*thread*) o proceso ligero
 2. Unidad mínima de planificación/ejecución



Hilos (*threads*)

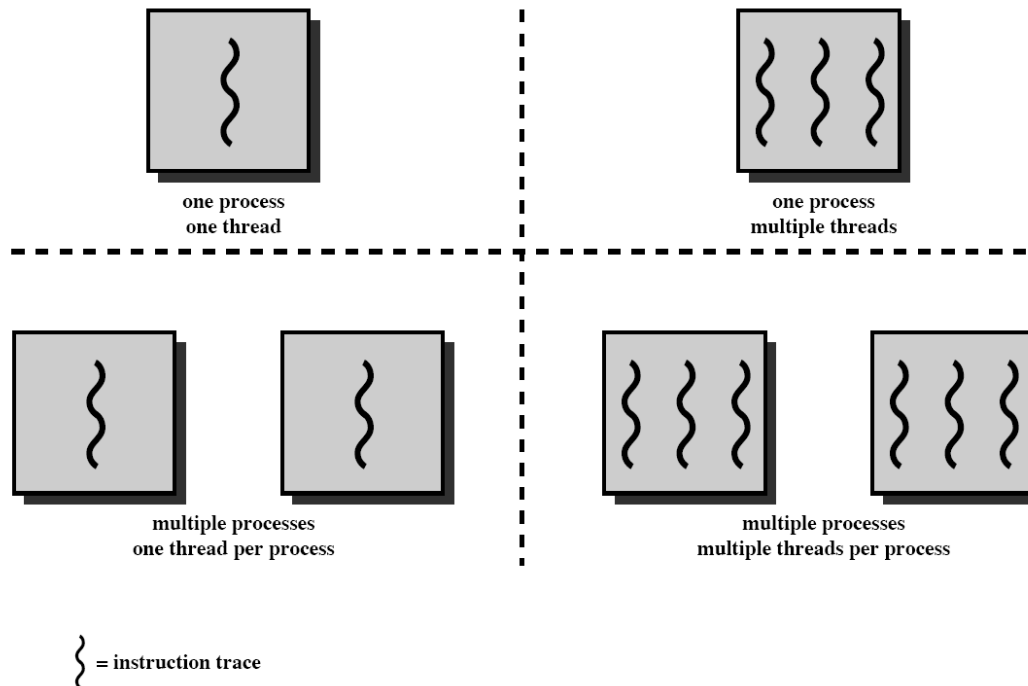
Introducción

- Los procesos son solo propietarios de los recursos → **No ejecutan código**
- Los procesos se componen de uno o más hilos
- Los hilos solo ejecutan código y sus recursos son los que tenga el proceso → No obstante son ellos los que los solicitan

Hilos (*threads*)

Introducción

- Sistemas operativos multihilo
 - El SO da soporte a múltiples hilos en un solo proceso





Hilos (*threads*)

Composición en memoria

- Los procesos tendrán:
 - Su imagen cargada en memoria principal (Igual que sin hilos)
 - Al menos un hilo de ejecución → Hilo principal
- Los procesos NO ejecutan código:
 - No se les asigna la CPU → No se les planifica
 - Carecen de estados ("Listo", "Suspendido", etc)
 - Carecen de pila
 - ¿Carecen de PCB?
 - **Sí tienen PCB** pero sin la sección 2 y parte de la sección 3 (Ver transparencias 11 y 12)



Hilos (*threads*)

Composición en memoria

- Los hilos ejecutan código, cada hilo tendrá:
 - Una pila de ejecución
 - Un “mini PCB” (llamado formalmente TCB → Thread Control Block) con:
 - Estado del hilo (“Listo”, “Ejecutando”, etc.)
 - Contexto de ejecución del hilo (sección 2 vista en transparencia 11)
 - Acceso a la memoria y recursos de su proceso, **compartido con el resto de hilos del mismo**



Hilos (*threads*)

Composición en memoria

Imagen del proceso



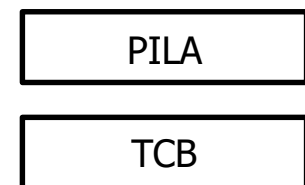
Hilo principal



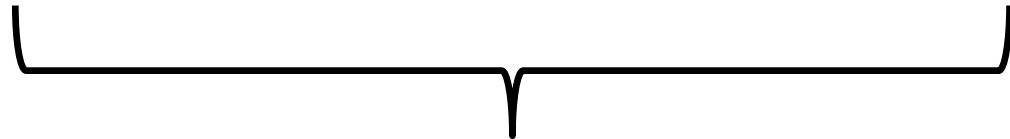
Hilo



Hilo



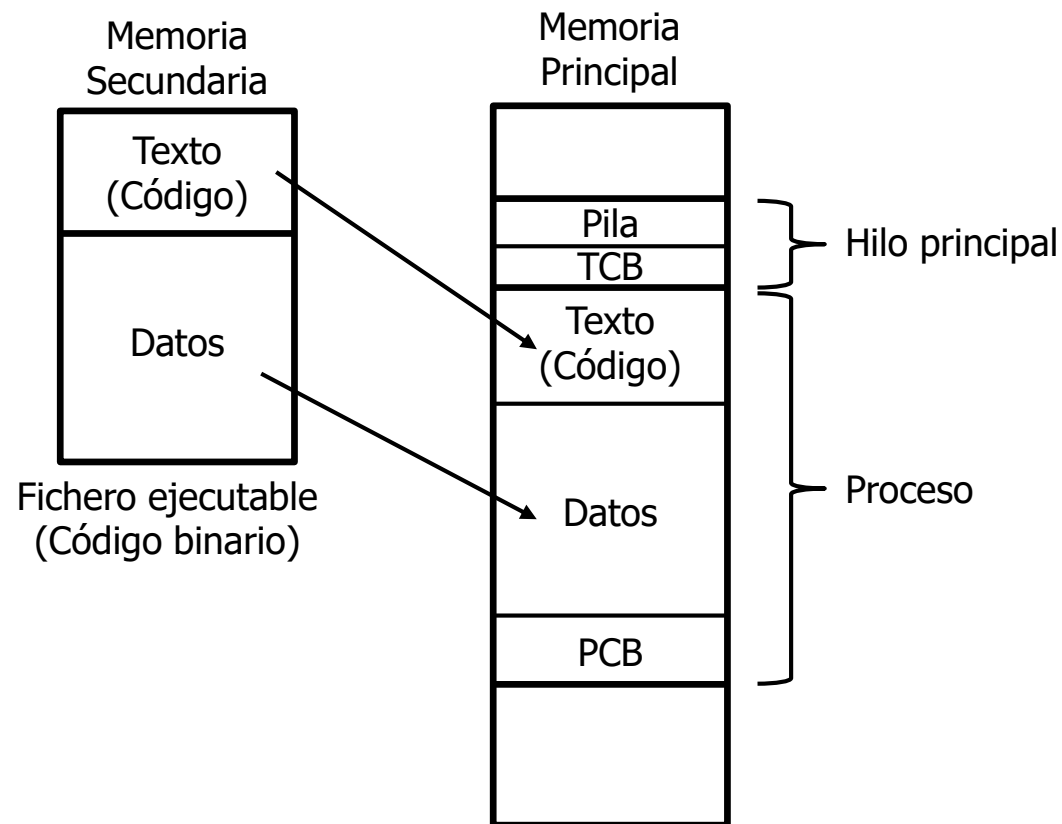
Hilos del proceso



Hilos (*threads*)

Creación de procesos e hilos

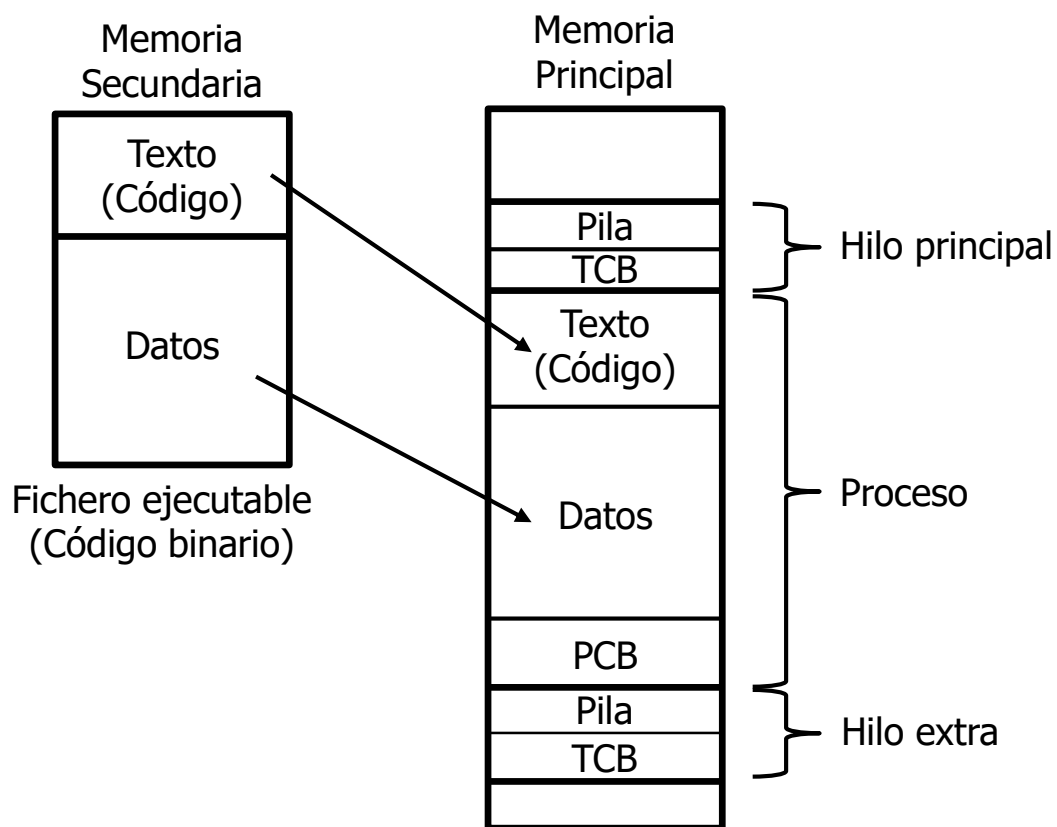
- El SO (PLP) carga el programa y **crea el proceso con UN hilo → Hilo principal**



Hilos (*threads*)

Creación de procesos e hilos

- El programador es responsable de crear si lo desea más hilos de ejecución → Con una llamada al sistema



- Crear un hilo es muy rápido y eficiente 😊
- Acceso a la memoria y recursos de su proceso, **compartido con el resto de hilos del mismo** 😊 😞
- **Ejecutan el mismo programa** 😞
 - **Solución:** Al crear un hilo se le asocia una función del programa 😊



Hilos (*threads*)

Gestión de procesos e hilos

- El SO gestiona y controla el proceso y todos sus hilos
 - Son los hilos los que tienen estados y no los procesos
 - El ciclo de vida (grafos de estados) son idénticos a los que ya vimos pero aplicados a los hilos.
 - Se asigna la CPU a un hilo, no a un proceso (PCP)
 - Se salva y restaura el contexto de los hilos, no de los procesos
 - Se suspenden y restauran procesos (PMP) → Todos sus hilos



Hilos (*threads*)

Finalización de procesos e hilos

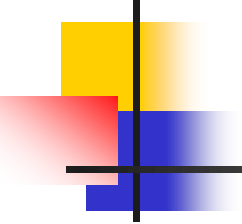
- Un hilo finaliza de manera independiente al resto de hilos del proceso (incluso con una excepción)
- El proceso finaliza cuando:
 - Finalizan todos sus hilos
 - En algunas implementaciones cuando finaliza su hilo principal → Se finalizan automáticamente el resto de hilos
- Solo se libera la memoria cuando finaliza EL PROCESO



Hilos (*threads*)

Beneficios de los hilos

- Menos costoso crear un hilo en un proceso existente que crear un proceso nuevo
- Menos costoso finalizar un hilo que un proceso
- Menos costoso hacer un “cambio de hilo” en un proceso que un “cambio de proceso”
- Los hilos del mismo proceso se comunican entre ellos compartiendo memoria



Hilos (*threads*)

Ejemplos de uso

- Trabajo en primer plano y segundo plano
 - Un hilo dedicado a la interfaz con el usuario
 - Otro hilo ejecuta los mandatos del usuario
- Procesamiento asíncrono (de eventos)
 - Un hilo en un procesador de textos que saca copia de seguridad periódica → Atender temporizador
- Velocidad de ejecución
 - Sacar partido de un sistema con varias CPUs y/o núcleos de CPU
- Servicios de Red
 - Cada hilo atiende a un cliente (conexión) distinto (Caso particular del segundo ejemplo de uso)



Lecturas recomendadas

- Stallings, “Sistemas Operativos”, 5ª edición
 - Capítulo 3, “Descripción y control de procesos”
 - Capítulo 4, “Hilos, SMP y micronúcleos”
- Silberschatz, “Fundamentos de Sistemas Operativos”, 7ª edición
 - Capítulo 3, “Procesos”
 - Capítulo 4, “Hebras”
- Nutt, “Sistemas Operativos”, 3ª edición
 - Capítulo 6, “Implementando procesos, hilos y recursos”