Tema

2.1

Sistema Operativos (SSOO)

Gestión de procesos

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Índice

- Creación de procesos.
- Terminación de procesos.
- Ciclo de vida de un proceso.
- Tipos de planificación.
- Algoritmos de planificación

Creación de procesos

Los SO proveen mecanismos para que los procesos puedan crear otros procesos → **Llamada al sistema**

El proceso de creación se puede repetir recursivamente creándose una "estructura familiar" → **Árbol de procesos**

Asignación de recursos al nuevo proceso:

- Los obtiene directamente del SO
- El padre debe repartir sus recursos con el proceso hijo o compartir todos o parte de ellos con él.

Esto no quiere decir que la comunicación entre padre e hijo sea natural y sencilla, NO.

Se evita así que un proceso bloquee el sistema multiplicándose indefinidamente.

Creación de procesos

Cuando se crea un proceso:

- En términos de ejecución
  - El padre continua ejecutándose en paralelo con su/s hijo/s.
  - El padre espera a que alguno o todos sus hijos hayan terminado.
- En términos del espacio en memoria
  - El proceso hijo es un clon del proceso padre.
  - El proceso hijo tiene ya un programa cargado en memoria.

Creación de procesos

¿Qué necesita saber el SO sobre cada proceso?

- Un proceso necesita la memoria suficiente para almacenar el programa y los datos que utiliza
- Atributos para que el SO pueda controlar el proceso. A estos atributos se les llama **Bloque de Control del Proceso (BCP)**
- A los programas, datos y atributos se le llama imagen del proceso

Creación de procesos

¿Qué tiene en concreto el **BCP**?

- Identificación del proceso: **PID, PPID y UID** del usuario que creó el proceso (ver orden top)
- Información del estado del procesador – Si se suspendió el proceso, se guardan aquellos registros del procesador que permitan reanudarlo más adelante.

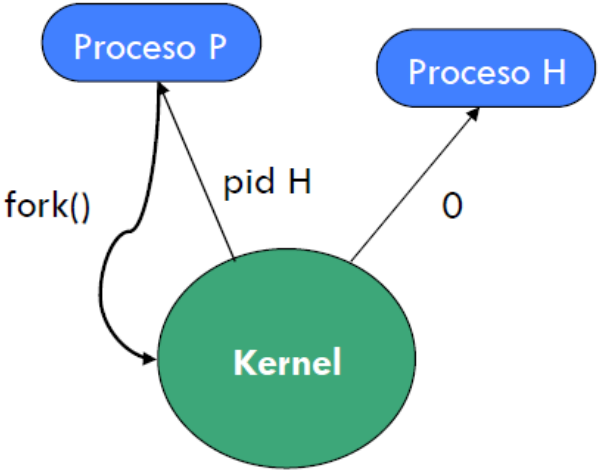
¿Dónde se guarda la imagen del proceso?

- En memoria principal (RAM) – si está en ejecución todo o parte tiene que estar aquí
- En memoria de disco (memoria virtual) – si está suspendido, todo. Si no, puede estar solo parte de la imagen.

Creación de procesos

En la familia Unix se distingue entre crear procesos y ejecutar nuevos programas.

- La llamada al sistema para crear un nuevo proceso se denomina ***fork()***.
- Esta llamada crea una copia casi idéntica del proceso padre.



- ✓ Ambos procesos, padre e hijo, continúan ejecutándose en paralelo.
- ✓ El padre obtiene como resultado de la llamada a ***fork()*** el pid del hijo y el hijo obtiene **0.**
- ✓ Algunos recursos no se heredan (p.ej. señales pendientes).

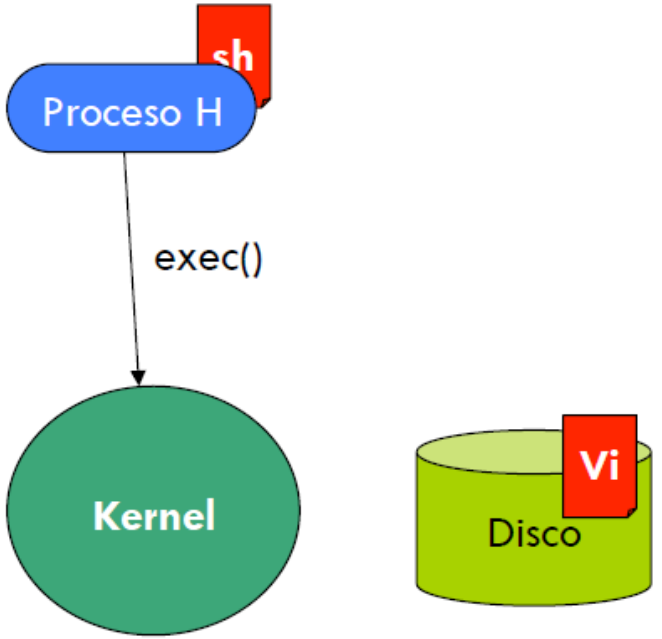
Creación de procesos

El proceso hijo puede invocar la llamada al sistema **exec***()

– sustituye su imagen en memoria por la de un programa diferente

El padre puede dedicarse a crear más hijos, o esperar a que termine el hijo

– **wait**() lo saca de la cola de "**listos**" hasta que el hijo termina



Creación de procesos

Ineficiencias del modelo ***fork()***

– Se copian muchos datos que podrían compartirse

– Si al final se carga otra imagen, todavía es peor porque todo lo copiado se deshecha.

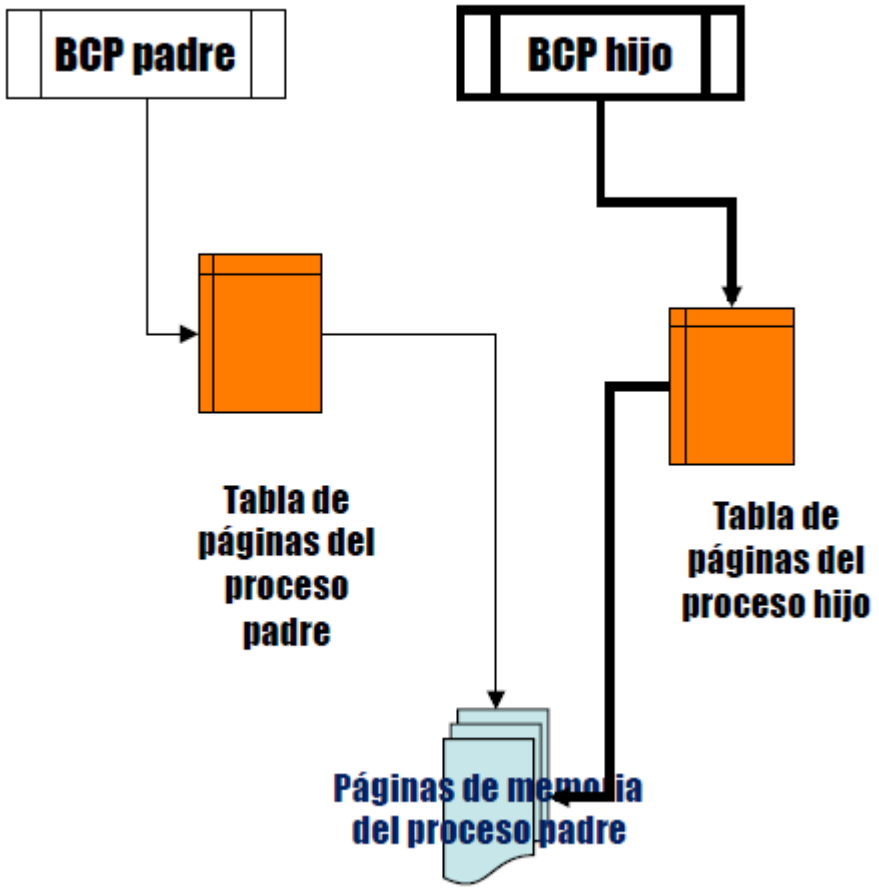
Muchos UNIX usan ***COW***

– ***Copy-on-Write*** es una técnica que retrasa o evita la copia de los datos al hacer el **fork.**

– Los datos se marcan de manera que si se intentan modificar se realiza una copia para cada proceso (padre e hijo).

– Ahora *fork()* sólo copia la tabla de páginas del padre (no las páginas) y crea un nuevo **BCP** para el hijo.

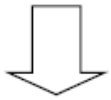
Creación de procesos



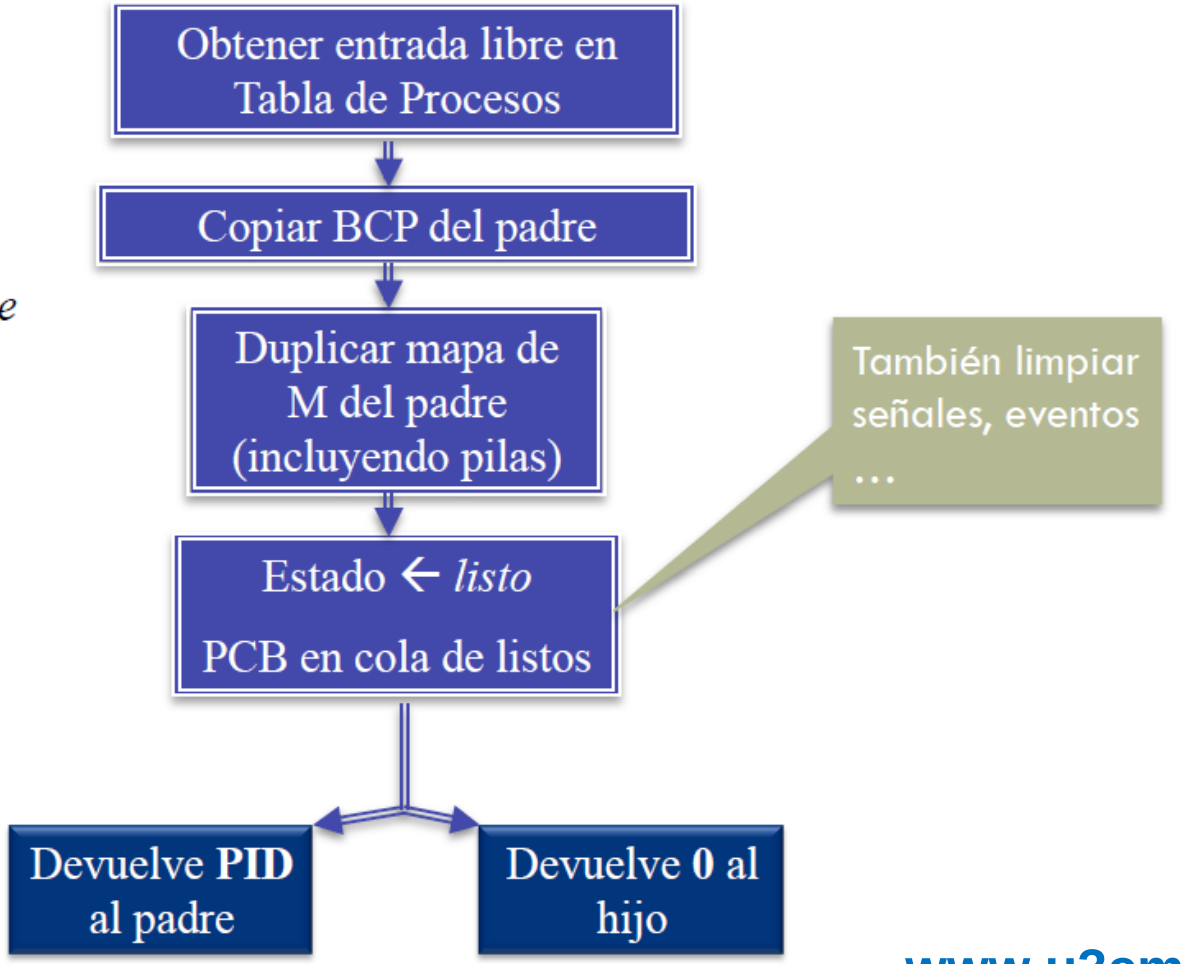
www.u3cm.es

Creación de procesos

`fork:`



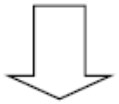
"Copia al proceso padre y le da una nueva identidad al hijo"



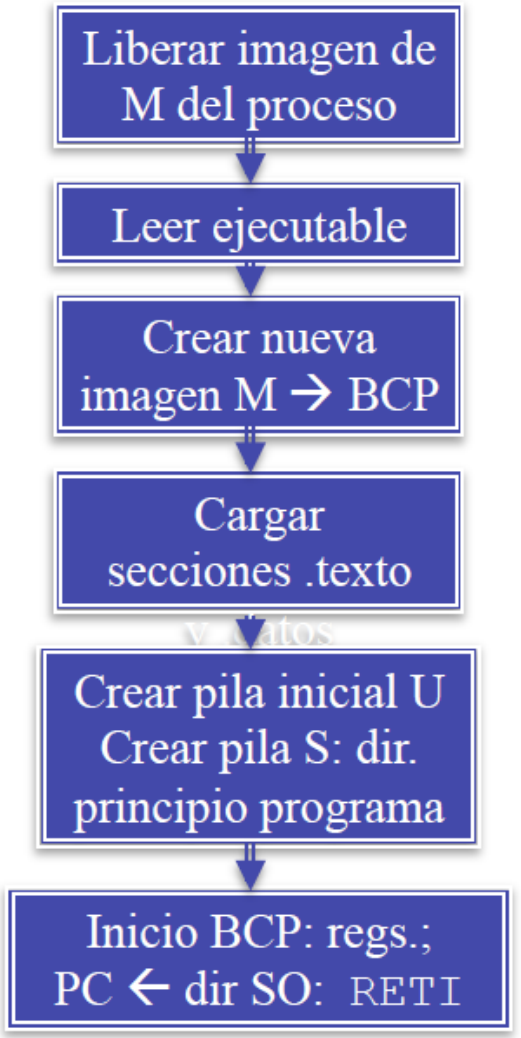
www.u3cm.es

Creación de procesos

`exec:`



"Cambia la imagen de M de un proceso usando como "recipiente" uno previo"



Terminación de procesos

Cuando un proceso termina todos los recursos asignados son liberados:

- memoria, ficheros abiertos, entradas en tablas,...

y el kernel notifica al proceso padre el evento.

Un proceso puede terminar de 2 formas:

- **Voluntariamente**: Llamada al sistema exit()
- **Involuntariamente**:
  - Excepciones: división por cero, violación de segmento
  - Abortado por el usuario (ctrl-c) u otro proceso (kill), es decir, señales que no puede manejar o ignorar.

Terminación de procesos

Cuando un proceso termina pueden suceder dos cosas:

- Sus hijos no se ven afectados
- Todos los hijos acaban también → **terminación en cascada (Ej. VMS)**

En Unix,

- Los hijos del proceso terminado pasan a depender del proceso ***init***
- El proceso finalizado pasa a estado Zombie hasta que el proceso padre recoge su código de finalización.

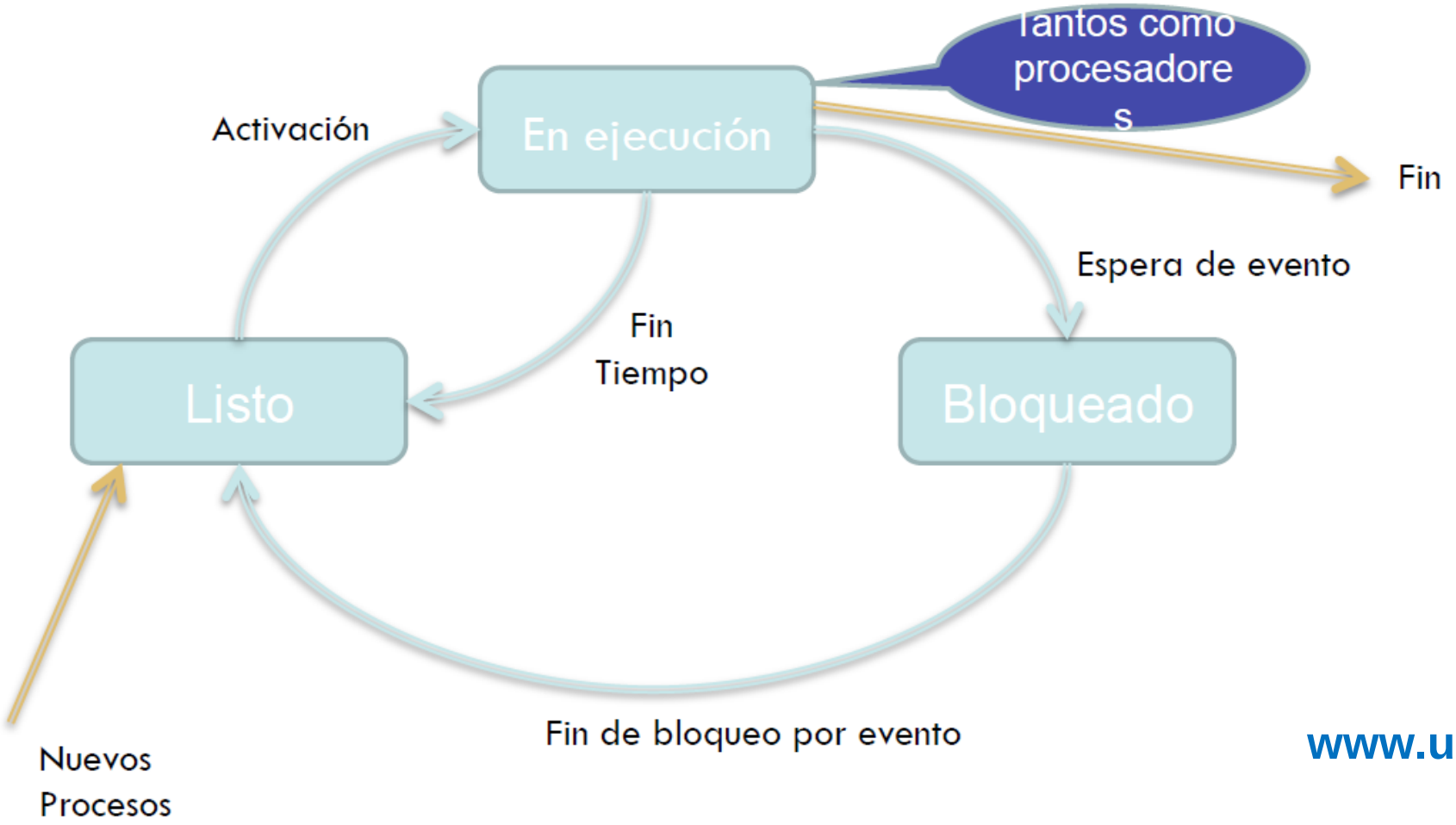
Terminación de procesos

Las terminación de un proceso y la eliminación de su BCP son tareas diferenciadas:

- Cuando el padre obtiene la información del hijo, se procede a eliminar las estructuras de datos
- Llamada al sistema *wait*()

- Bloquea al proceso hasta que termina el/un hijo
- Devuelve el PID del hijo finalizado

Ciclo de vida de un proceso

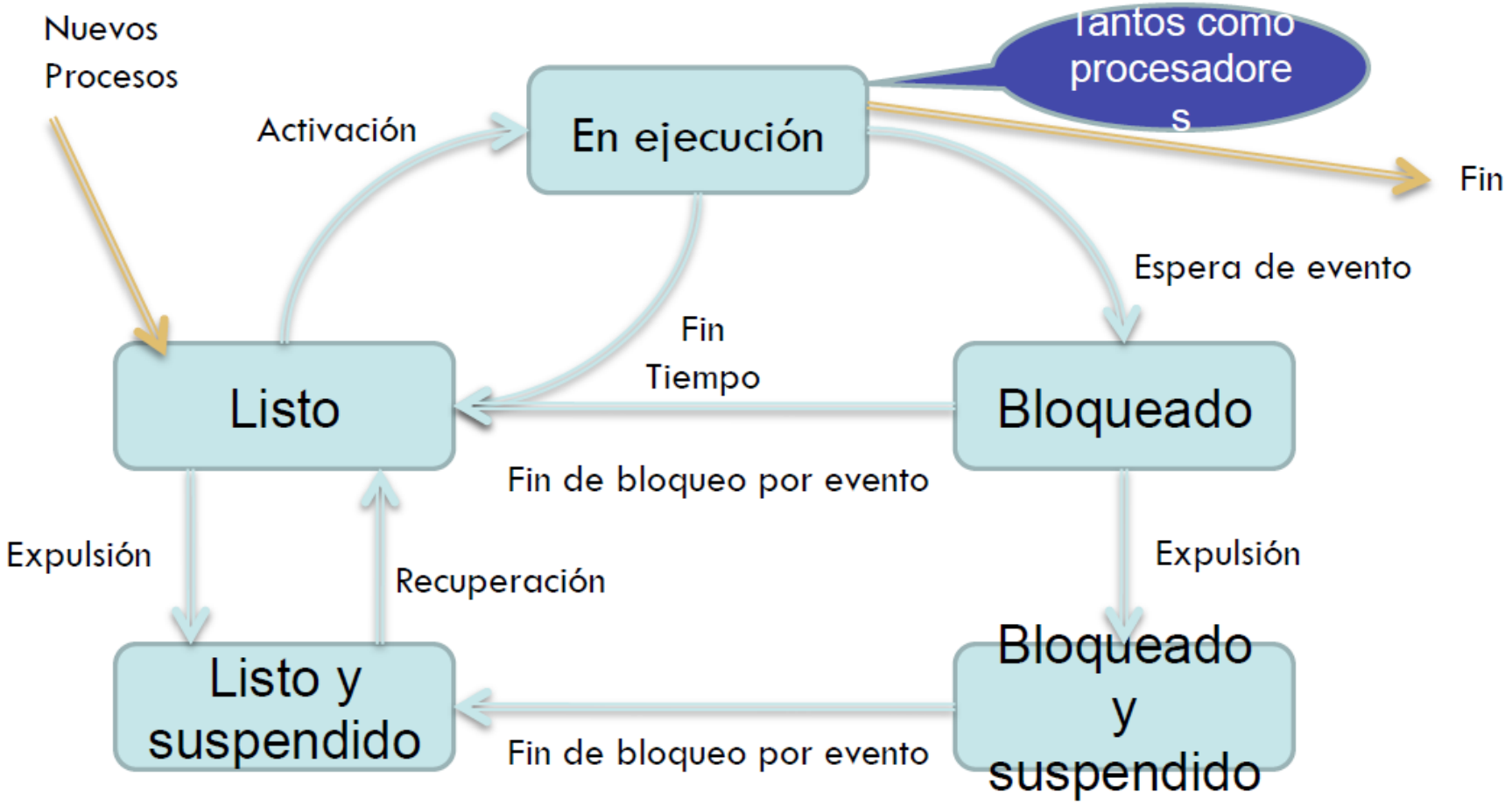


www.u3cm.es

Ciclo de vida de un proceso

- Cuando existen muchos procesos en ejecución el rendimiento puede bajar por excesiva paginación.
  - Solución: El Sistema Operativo puede expulsar totalmente procesos al área de intercambio del disco.
- Introduce nuevos estados de los procesos.
  - Bloqueado y suspendido.
  - Listo y suspendido.

Ciclo de vida de un proceso



Tipos de planificación (Niveles)

Planificación a corto plazo

– Selecciona el siguiente proceso a ejecutar.

Planificación a medio plazo

– Selecciona qué procesos se añaden o se retiran (expulsión a swap) de memoria principal.

Planificación a largo plazo

– Realiza el control de admisión de procesos a ejecutar.

– Muy usada en sistemas batch.

Tipos de planificación

- No apropiativa.
  - El proceso en ejecución conserva el uso de la CPU mientras lo desee.
- Apropiativa.
  - El sistema operativo puede expulsar a un proceso de la CPU.

Puntos de decisión de planificación

Momentos en los que se puede decidir la planificación de un proceso:

1. Cuando un proceso se bloquea en espera de un evento
   - Realización de una llamada al sistema.
2. Cuando se produce una interrupción.
   - Interrupción del reloj.
   - Interrupción de fin de E/S.
3. Fin de proceso.

Planificación no apropiativa: 1 y 3.

- Windows95, MacOS anteriores a versión 8.

Planificación apropiativa: 1, 2 y 3.

Pilas de procesos

Los procesos listos para ejecutar se mantienen en una cola.

Alternativas:

- Cola única.
- Colas por tipos de procesos.
- Colas por prioridades.

Algoritmos de planificación (Medidas)

- **Utilización de CPU**:
  - Porcentaje de tiempo que se usa la CPU.
  - Objetivo: Maximizar.
- **Productividad**:
  - Número de trabajos terminados por unidad de tiempo.
  - Objetivo: Maximizar.
- **Tiempo de retorno** (Tq)
  - Tiempo que está un proceso en el sistema. Instante final (Tf) menos instante inicial (Ti).
  - Objetivo: Minimizar.

Algoritmos de planificación (Medidas)

- Tiempo de servicio (Ts):
  - Tiempo dedicado a tareas productivas (cpu, entrada/salida). $Ts = T_{CPU} + T_{E/S}$
- Tiempo de espera (Te):
  - Tiempo que un proceso pasa en colas de espera.

    $Te = Tq - Ts$
- Tiempo de retorno normalizado (Tn):
  - Razón entre tiempo de retorno y tiempo de servicio.

    $Tn = Tq/Ts$
  - Indica el retardo experimentado.

Algoritmos de planificación

Grupo 1: Asignación FCFS

First to Come First to Serve: Primer en llegar primero en servir.

- Algoritmo no apropiativo.
- Penaliza a los procesos cortos.

Grupo 2: Asignación SJF

***Shortest Job First*:** Primero el trabajo más corto.

- Algortimo no apropiativo.
- Selecciona el trabajo más corto.
- Solamente se puede aplicar si se conoce de antemano la duración de cada trabajo.
- Posibilidad de inanición:
  - Si continuamente llegan trabajos cortos, los trabajos largos nunca llegan a ejecutarse.

Algoritmos de planificación

Grupo 3: Cíclico o Round-Robin

– Mantiene una cola FIFO con los procesos listos para ser ejecutados.

– Un proceso recibe el procesador durante un cuanto o rodaja de tiempo.

– Algoritmo apropiativo.

Grupo 4: Planificación en Windows

Principales características:

– Basado en prioridades y uso de cuantos de tiempo.

– Planificación apropiativa.

– Planificación con afinidad de procesador.

No olvidemos que…

- La creación de un proceso implica la creación de su imagen de memoria y de su BCP.
- Un proceso pasa por distintos estados durante su ejecución.
- El sistema operativo realiza la planificación de los procesos.
- La planificación puede ser apropiativa y no apropiativa.
- Los distintos algoritmos de planificación de procesos pueden favorecer más o menos a un tipo de procesos.
- Los sistemas operativos modernos usan planificación apropiativa.

Bibliografía

- **U3CM**, www.u3cm.es, Material de la asignatura Sistemas Operativos
- **CARRETERO**, Jesús, GARCÍA, Félix, DE MIGUEL, Pedro, PÉREZ, Fernando. Sistemas Operativos: una visión aplicada. McGraw-Hill, 2001.
- **STALLINGS**, William. Sistemas operativos: aspectos internos y principios de diseño. 5ª Edición. Editorial Pearson Educación. 2005. ISBN: 978-84-205-4462-5.
- **TANENBAUM**, Andrew S. Sistemas operativos modernos. 3ª Edición. Editorial Prentice Hall. 2009. ISBN: 978-607- 442-046-3.

marlon.cardenas@ufv.es

Universidad
Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior