Sistemas Operativos 1

Procesos

Planificación, Creación, Estados e Hilos

Edwin Salvador

22 de octubre de 2015

Sesión 4

Contenido I

Deber

- Consultar sobre los diferentes tipos de sistemas operativos que existen o han existido y proporcionar una lista de aplicaciones donde podrían ser más útiles (un tipo de aplicación para cada tipo de SO.)
- Límite 29 octubre 2015 23:59 vía turnitin

Contenido I

Planificadores

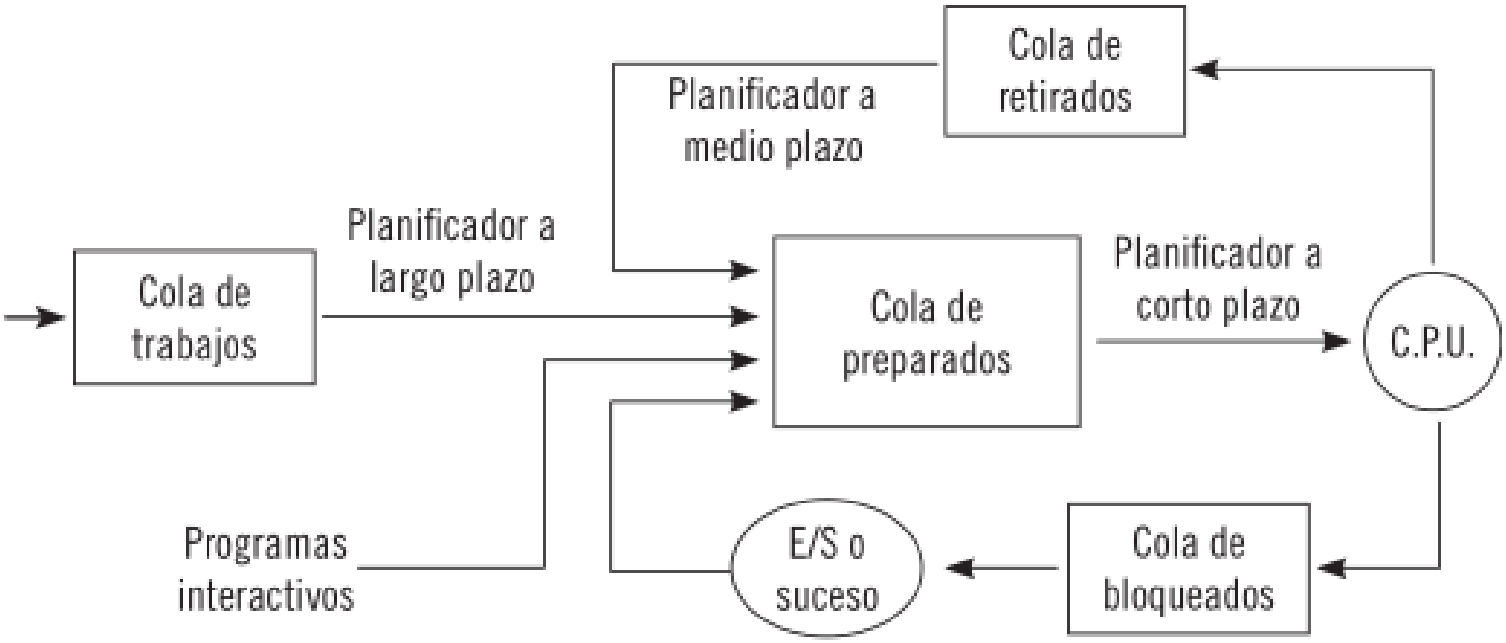
- El planificador se encarga de establecer el orden de ejecución de los procesos.
- Utiliza un algoritmo de planificación.
- Un proceso necesita recursos para ejecutarse (tiempo de CPU, memoria, archivos, dispositivos E/S).
- Estos recursos son asignados cuando se crea el proceso o durante su ejecución.

Tipos de planificadores

- **A largo plazo** entran en funcionamiento cuando un proceso finaliza o si no finaliza en un cierto tiempo. Trabajan con la cola de procesos por lote y admiten estos al sistema por lote.
- **A medio plazo** se encargan de ordenar los procesos que están bloqueados y de insertarlos y quitarlos de la memoria para ponerlos en la cola de ejecutables.
- **A corto plazo** eligen que proceso va a ser asignado a la cola de CPU.

Los SO antiguos no tenían planificadores a largo y medio plazo.
Los planificadores a largo y medio plazo regulan la carga del sistema.

Diagrama de planificadores



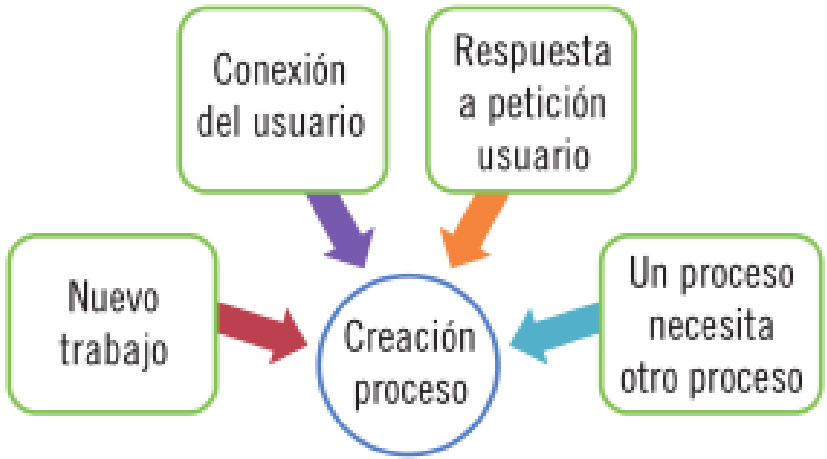
Algoritmos de planificación

- **FIFO** atiende al primer proceso que llega y el resto entra en cola de espera. **Desventajas** Penaliza procesos cortos, tiempo de respuesta impredecible.
- **Round-Robin** Se define un tiempo de ejecución para los procesos y se los atiende de manera circular por turnos. Si el proceso no ha finalizado en el intervalo se coloca al final de la lista.
- **Short job first** (*Trabajo más corto primero*) atiende al proceso más corto de la cola. La CPU no dejará al proceso hasta que finalice.
- **Prioridad multinivel** Los procesos se asignan por prioridad en diferentes colas. El SO aplicará un determinado algoritmo a cada cola.
- **High remainder next** la prioridad es marcada por el tiempo que está esperando en la cola y el tiempo necesario para su finalización.

Contenido I

Creación de procesos

- Cuando un nuevo proceso se va a añadir, el SO construye las estructuras de datos para manejar el proceso (BCP) y reserva el espacio de direcciones de memoria principal para el proceso.
- Existen cuatro eventos que llevan a la creación de un proceso:
  - Recepción y admisión de un nuevo trabajo.
  - La conexión del usuario provoca la creación de un proceso que ejecute el shell.
  - Respuesta a una petición del usuario.
  - Un proceso solicita la creación de otro proceso.



Terminación de procesos

De igual manera existen varias razones que pueden llevar a la terminación de un proceso.

- Finalización normal.
- Límite de tiempo excedido (en ejecución).
- Límite de tiempo (en espera)
- Memoria no disponible.
- Violaciones de frontera: acceder a direcciones de memoria no permitidas.
- Error de protección: acceder a recursos no permitidos o utilizarlos de maneras no apropiadas (archivos de solo lectura).
- Error aritmético: división por 0, representación de números mayores a la capacidad del hardware.
- Fallo de E/S: fichero no encontrado.
- Instrucción no válida.
- Instrucción privilegiada: ejecución de instrucciones exclusivas al SO.

Terminación de procesos

- Uso inapropiado de datos.
- Intervención del operador por el SO.
- Terminación del proceso padre.
- Solicitud del proceso padre.

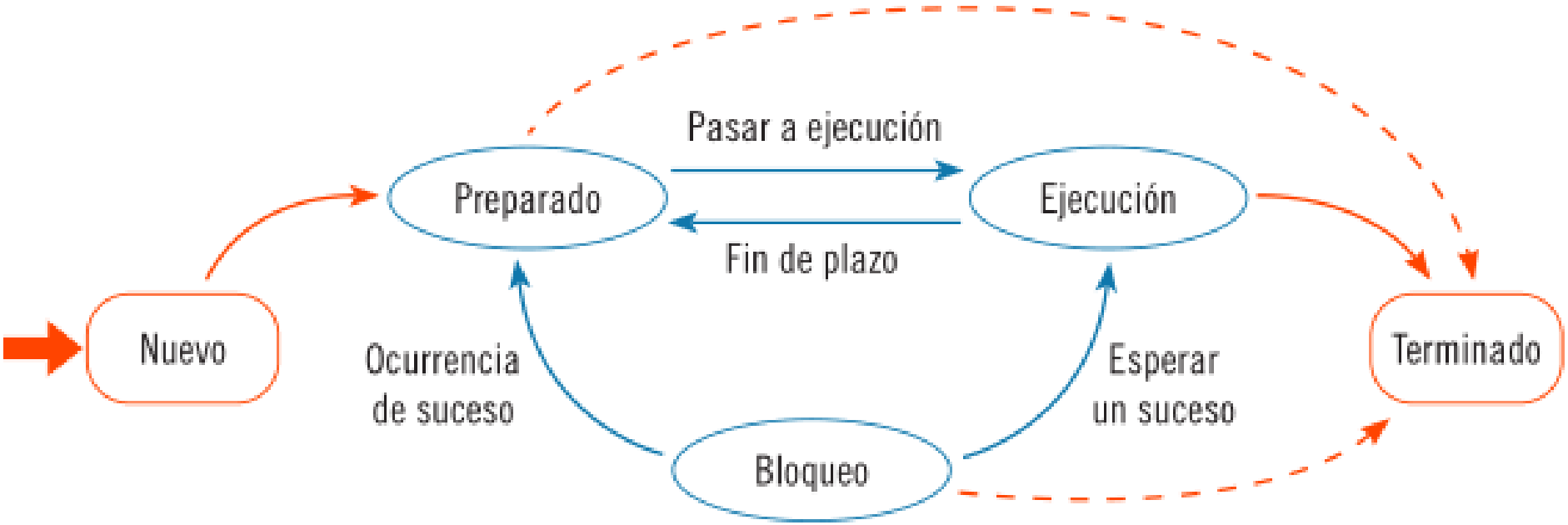
Contenido I

Estados de un proceso

Los procesos evolucionan de manera cíclica entre diferentes estados. Estos estados pueden ser (los más comunes):

- **Nuevo** procesos que acaban de ser incluidos y el SO aún no los admite para ser ejecutados.
- **Listo** procesos que cuentan con todos los recursos para comenzar o seguir su ejecución.
- **Ejecución** proceso que tiene el control del procesador. En sistemas con un solo procesador solo un proceso puede estar en este estado.
- **Bloqueado** procesos en espera de algún recurso o evento para continuar su ejecución.
- **Terminado** procesos excluidos por el SO del grupo de procesos ejecutables. Un proceso pasa a terminado:
  - terminación normal
  - error irrecuperable
  - terminación por otro proceso autorizado.

Estados de un proceso



Contenido I

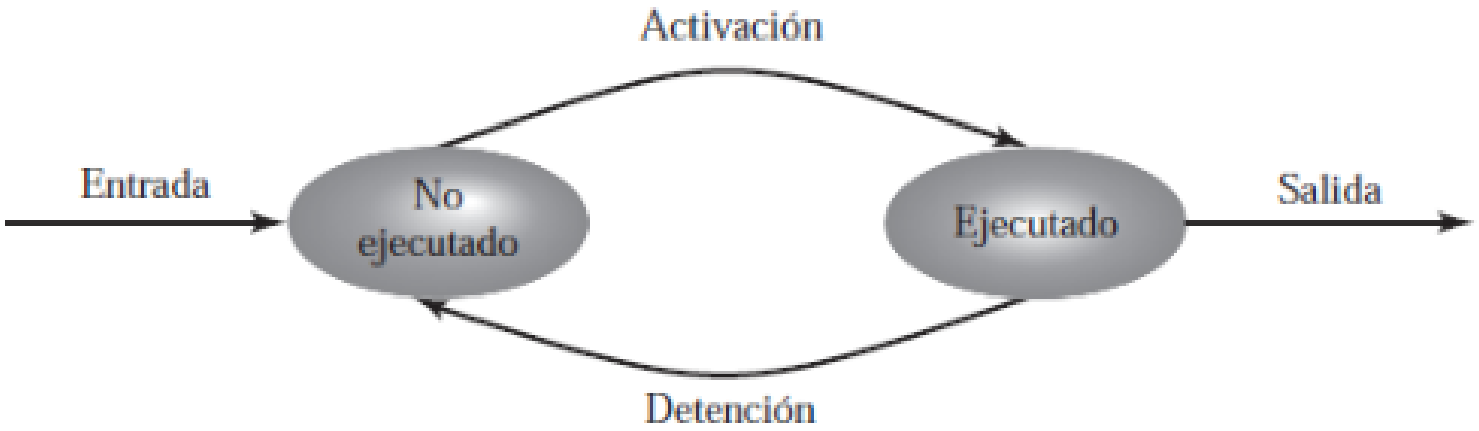
Modelos de estados de procesos

- La responsabilidad principal del SO es controlar la ejecución de los procesos, por lo tanto debe determinar el patrón de entrelazado para la ejecución y asignación de recursos a estos procesos.
- El primer paso para el diseño de un SO es describir el comportamiento que que se desea que tengan los procesos.
- A continuación veremos dos modelos de estados de procesos: un modelo de dos estados y uno de cinco estados.

Modelo de procesos de dos estados

- Si tenemos en cuenta que en un instante dado un proceso está siendo **ejecutado o no**, entonces podemos decir que un procesos estará en dos posibles estados **Ejecutando** y **No Ejecutando**.
- Cuando el SO, crea un proceso, crea un BCP para este nuevo proceso e inserta el proceso en el sistema en estado *No Ejecutando*. Entonces, en este punto el proceso ya existe, es conocido por el SO pero está esperando su oportunidad de ser ejecutado.
- Cada cierto tiempo el proceso actualmente en ejecución se interrumpirá y pasará al estado *No Ejecutando*.
- El **activador** seleccionará otro proceso para ejecutar y este pasará al estado *Ejecutando*.
- Los procesos en estado *No Ejecutando* deben ser almacenados en una especie de cola que contiene punteros a los BCP de cada proceso en este estado.

Modelo de procesos de dos estados



(a) Diagrama de transiciones de estados

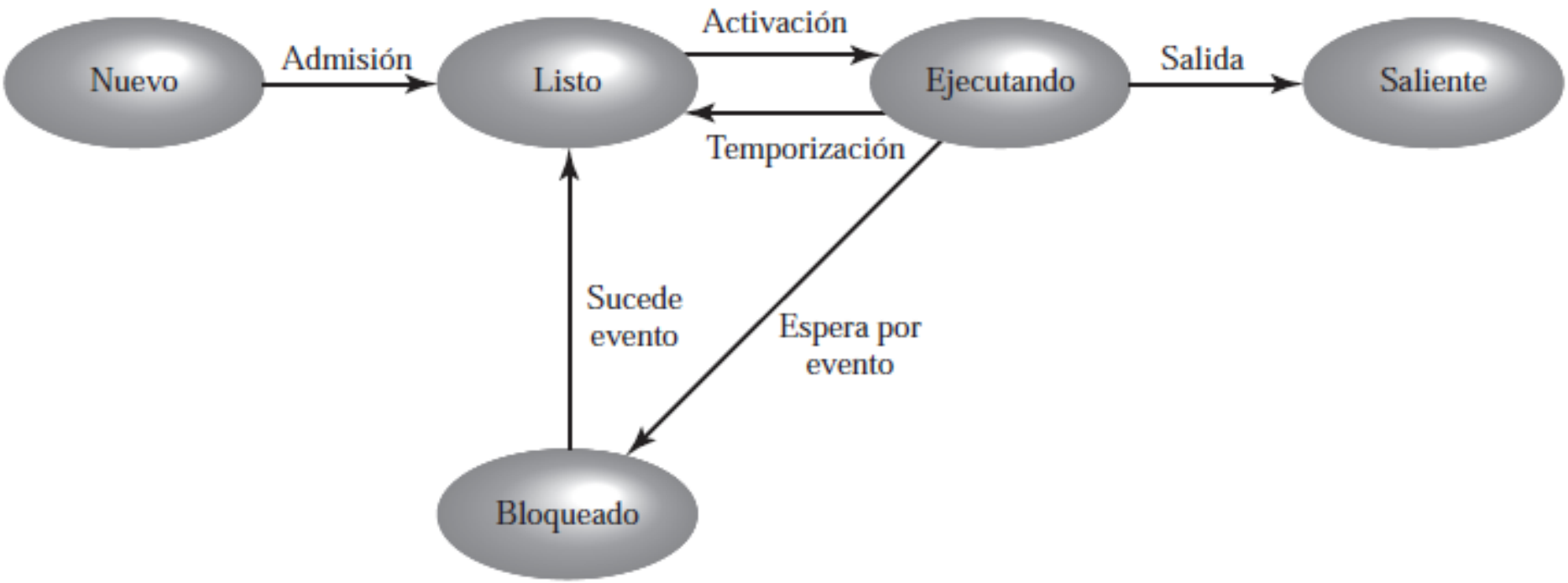
Modelo de procesos de dos estados

- Este modelo de gestión de colas sería efectiva si todos los procesos estuviesen siempre preparados para ejecutar.
- Esta cola es de tipo FIFO por lo tanto no toma en cuenta que procesos llevan más tiempo en la cola o que procesos están bloqueados esperando alguna operación de E/S.
- Por esta razón este modelo de dos estados no es eficiente.

Modelo de procesos de cinco estados

- Un modelo más eficiente podría dividir el estado *No Ejecutando* en dos: *Listo* y *Bloqueado*.
- Este modelo añade también un estado *Nuevo* y otro *Saliente*.
- Entonces los cinco estados son:
  - Ejecutando
  - Listo
  - Bloqueado (en espera de evento u operación E/S)
  - Nuevo (BCP creado pero aún no cargado en memoria principal.
  - Saliente (proceso detenido o abortado)

Modelo de procesos de cinco estados



Modelo de procesos de cinco estados

Posibles transiciones entre estados

Las posibles transiciones entre los 5 estados son:

- Null $\rightarrow$ Nuevo
- Nuevo $\rightarrow$ Listo
- Listo $\rightarrow$ Ejecutando
- Ejecutando $\rightarrow$ Saliente
- Ejecutando $\rightarrow$ Listo
- Ejecutando $\rightarrow$ Bloqueado
- Bloqueado Listo $\rightarrow$ Nuevo
- Listo $\rightarrow$ Saliente
- Bloqueado $\rightarrow$ Saliente

Modelo de procesos de cinco estados

- En este modelo los estados *Listo* y *Bloqueado* son manejados por colas independientes.
- Cuando el SO debe seleccionar el siguiente proceso a ejecutar busca en la cola de *Listos*. Podría existir un esquema de prioridades o simplemente un esquema FIFO.
- Si se maneja un esquema de prioridades, sería conveniente tener varias colas de procesos listos, una por cada prioridad. Así el SO podría seleccionar el procesos listo de mayor prioridad más rápidamente.

Modelo de procesos de cinco estados

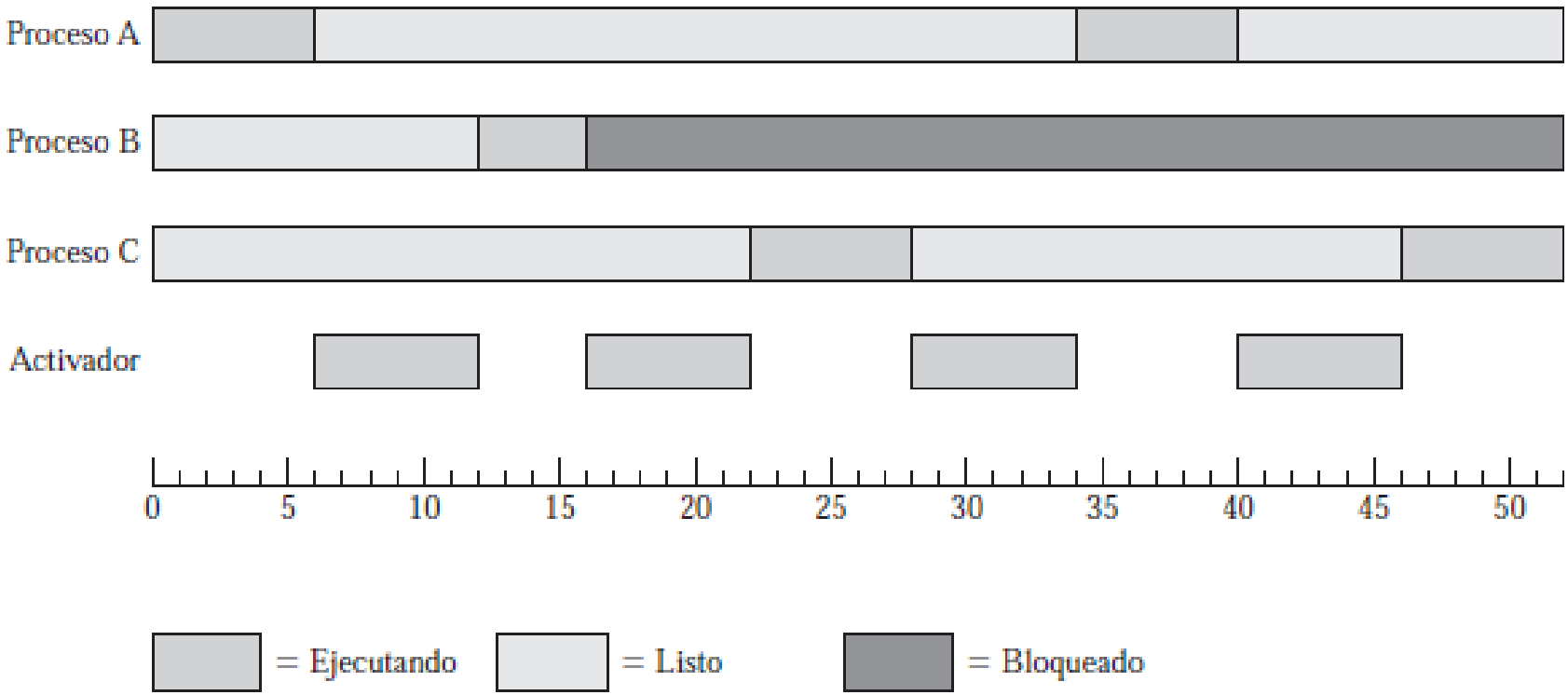
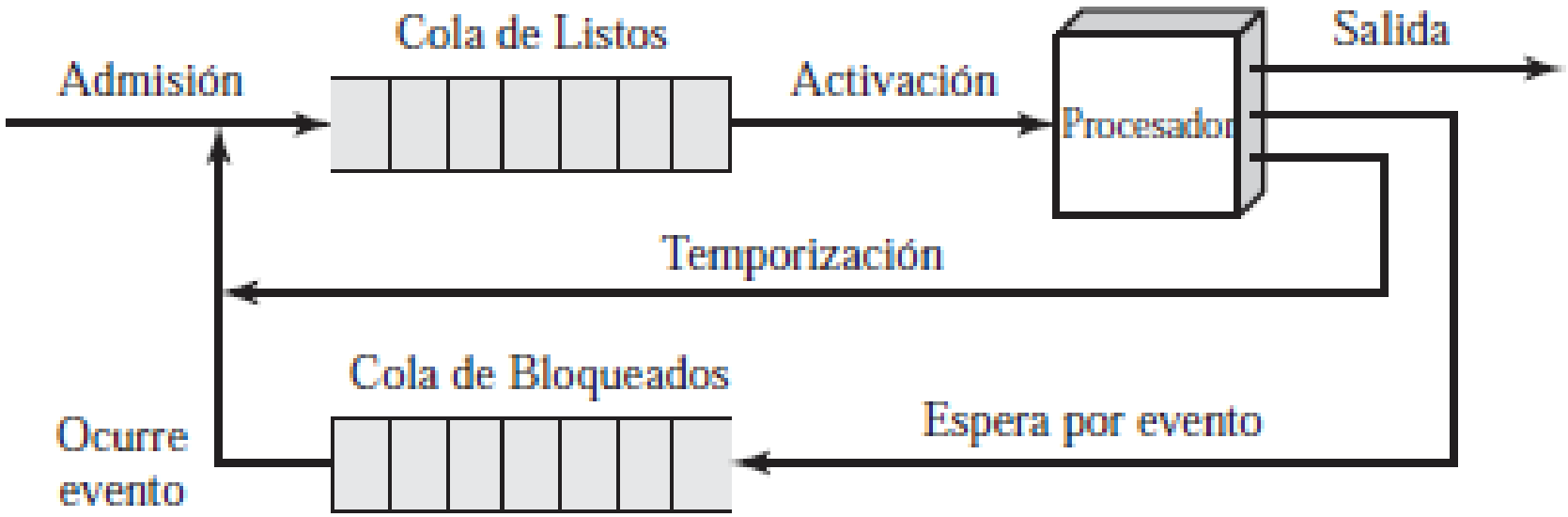


Figura: Estados de los procesos A, B y C

Modelos de procesos de cinco estados

Modelo de cola simple

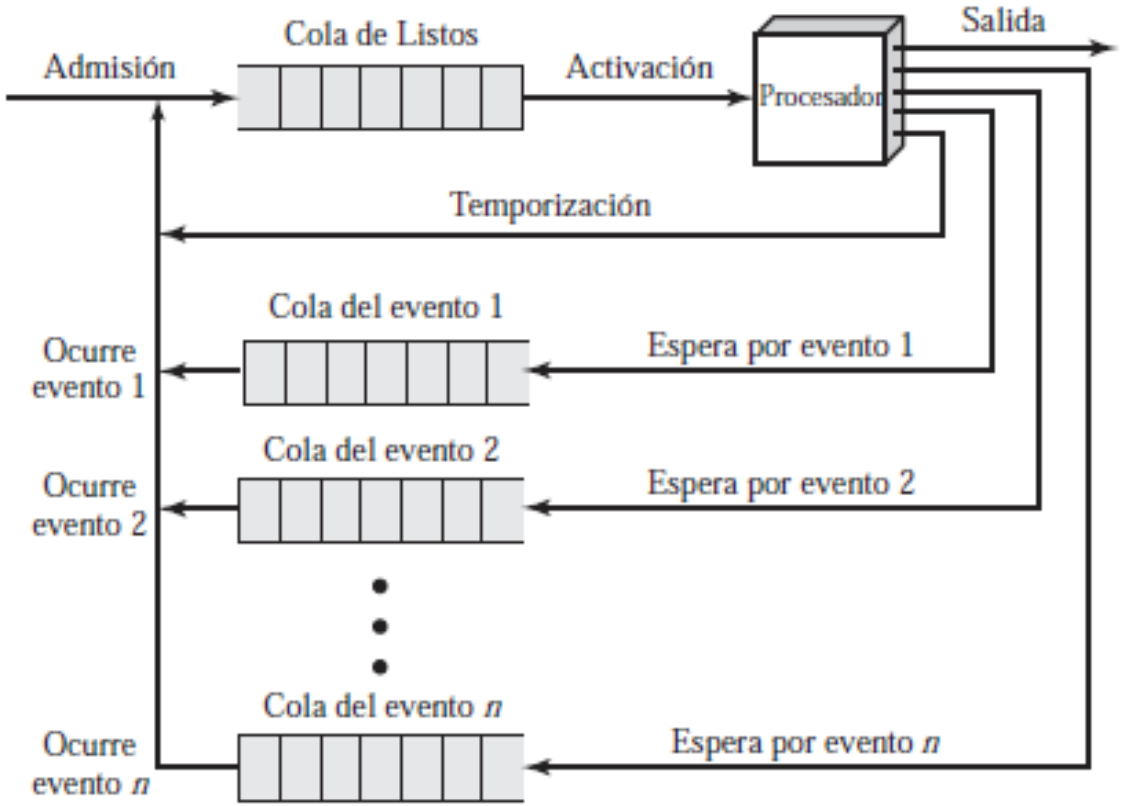


(a) Cola simple de Bloqueados

Cuando sucede algún evento el SO debe recorrer la cola de *Bloqueados* en busca de los procesos que estén esperando por este evento.

Modelos de procesos de cinco estados

Modelo de múltiples colas de bloqueados



(b) Múltiples colas de Bloqueados

En algunos SO las colas de procesos *Bloqueados* podría contener miles de procesos por lo que sería más eficiente manejar una cola por cada evento.

Contenido I

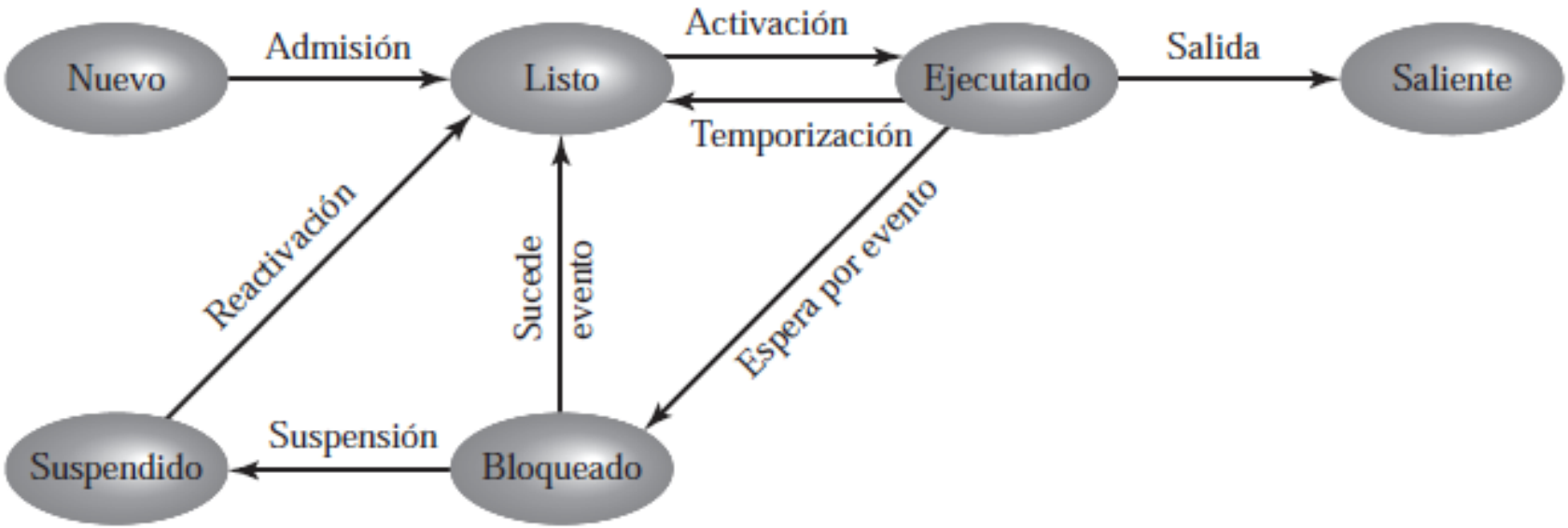
Swapping (procesos suspendidos)

- El modelo de múltiples colas de procesos *Bloqueados* aumenta la eficiencia del sistema, sin embargo la diferencia entre la velocidad del procesador y los dispositivos E/S es tan grande que es muy probable que todos los procesos lleguen a estar en un estado *Bloqueado*.
- Si tenemos miles de procesos en estado *Bloqueado*, estos podrían estar ocupando todo el espacio en memoria principal y el procesador estaría ocioso debido a que no tendríamos capacidad para aceptar nuevos procesos.
- Es cierto que podríamos aumentar la memoria pero esto también aumentaría el costo del sistema. Y es importante notar que debido a las grandes memorias disponibles, la tendencia es a ejecutar **procesos más grandes y no más procesos**.

Swapping (procesos suspendidos)

- Una solución para lograr aceptar más procesos y disminuir el tiempo ocioso del procesador es utilizar una técnica llamada *swapping* (memoria de intercambio).
- El swapping consiste en intercambiar parte o todo el proceso a de memoria principal a disco.
- Si no existen procesos en memoria principal en estado *Listo*, el SO mueve uno de estos procesos *Bloqueados* a disco (a la cola de procesos *Suspendidos*).
- El swapping añade el estado *Suspendido*
- Así el SO puede continuar con la creación de nuevos procesos o traer procesos de la cola de *Suspendidos* que puedan continuar siendo ejecutados.
- Esta técnica generalmente mejora el rendimiento del sistema.

Swapping (procesos suspendidos)



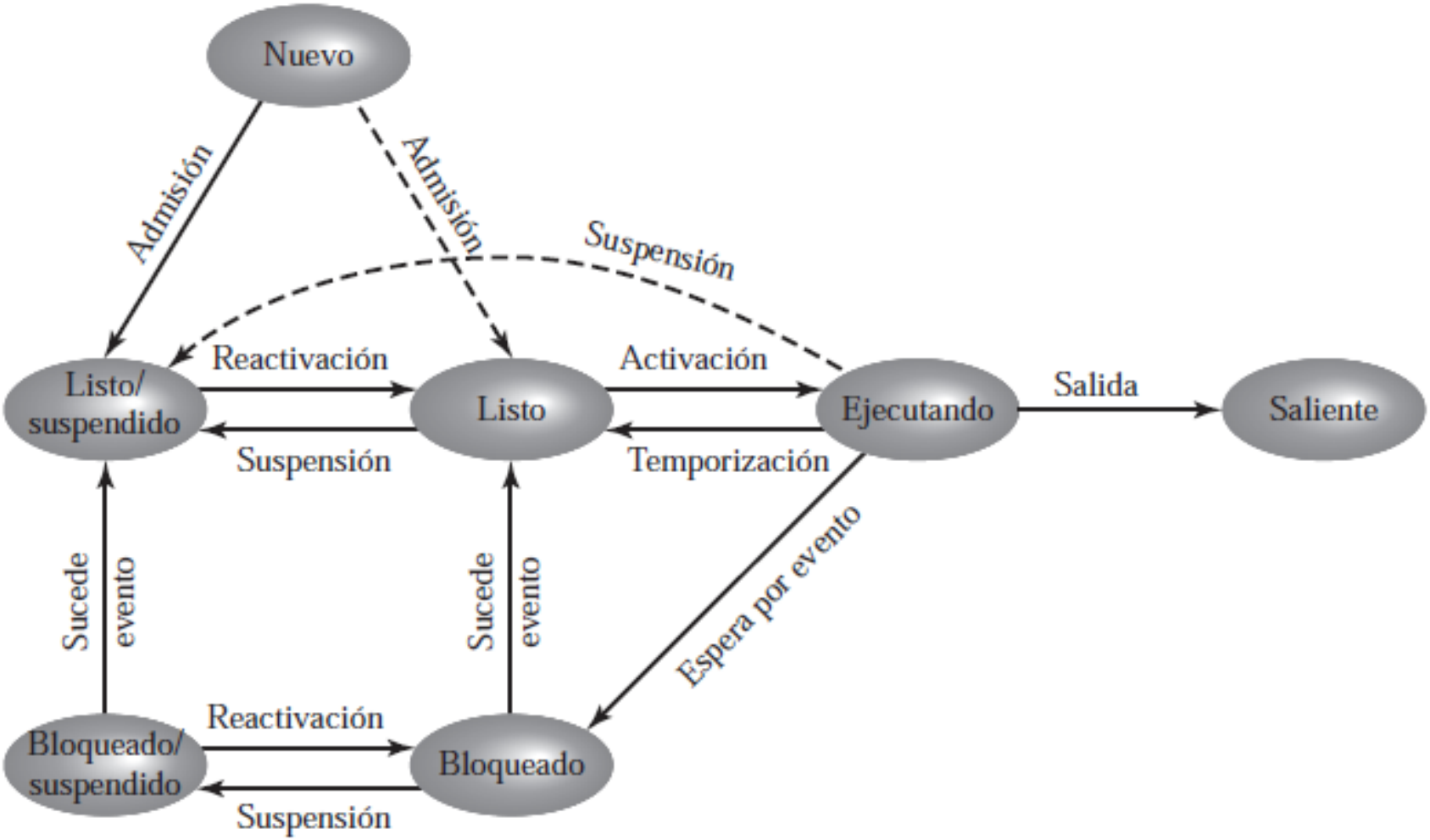
Modelo de estados con swapping

Para que el SO pueda determinar más fácilmente que proceso debe traer de la cola de *Suspendidos* se necesitan los siguientes estados:

- Listo
- Bloqueado
- Bloqueado/Suspendido
- Listo/Suspendido

Modelo de estados con swapping

Utilizando dos estados suspendido



Transiciones con swapping (dos estados suspendidos)

La figura anterior nos muestra las posibles transiciones entre estados cuando se utilizan dos estados suspendidos. Estas transiciones son:

- **Bloqueado** $\rightarrow$ **Bloqueado/Suspendido**
- **Bloqueado/Suspendido** $\rightarrow$ **Listo/Suspendido**
- **Listo/Suspendido** $\rightarrow$ **Listo**
- **Listo** $\rightarrow$ **Listo/Suspendido** (liberar memoria o prioridad baja)
- **Nuevo** $\rightarrow$ **Listo/Suspendido o Listo** (depende espacio en memoria)
- **Bloqueado/Suspendido** $\rightarrow$ **Bloqueado** (prioridad alta o evento pronto a ocurrir)
- **Ejecutando** $\rightarrow$ **Listo/Suspendido** (liberar memoria)
- **Cualquier estado** $\rightarrow$ **Saliente**

Contenido I

Procesos e Hilos

- Hasta ahora conocemos a un proceso como poseedor de tres características:
  - Es un conjunto de instrucciones.
  - Puede obtener el control de recursos (memoria principal, canales y dispositivos E/S y archivos)
  - Puede estar en diferentes estados de ejecución (ejecutando, listo, etc) puede ser planificada y activada por el SO.
- Estas características son la esencia de un proceso.
- Para distinguir estas hacer una distinción definimos a la unidad que se **activa** como **hilo (thread)**, mientras que la unidad de propiedad de **recursos** la definimos como **proceso** o **tarea**.

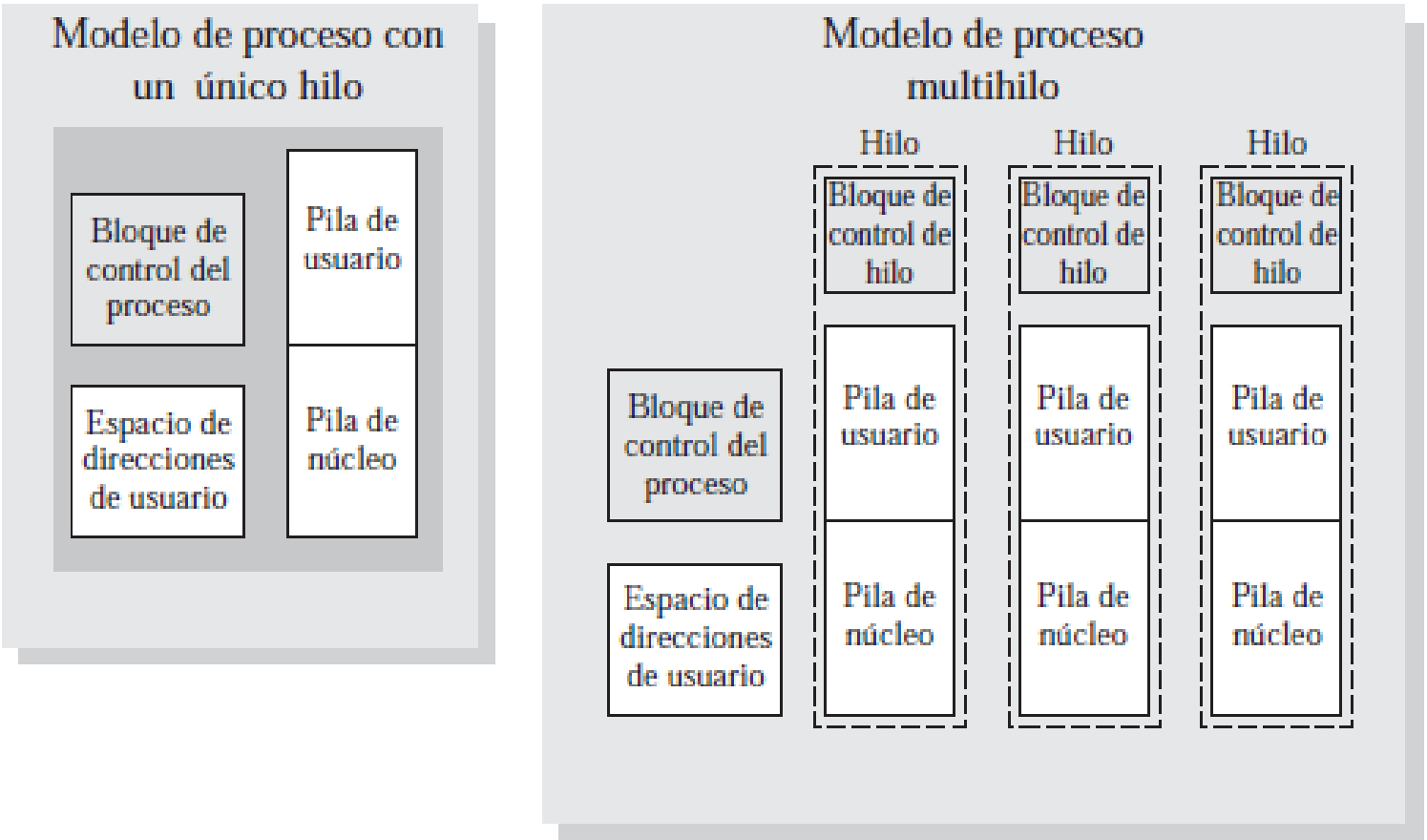
Multihilo

- Multihilo es la capacidad de un SO de dar soporte a múltiples hilos de ejecución en un solo proceso.
- No todos los SO soportan multihilo. Ejemplo:
  - MS-DOS = 1 proceso de usuario y 1 único hilo.
  - Otros SO y algunas versiones de UNIX = múltiples procesos de usuario, pero 1 solo hilo.
- ¿Cuál es la diferencia entre varios procesos de un solo hilo y un proceso de varios hilos?
- Windows, Solaris, Mac OS X, Linux entre otros soportan múltiples procesos y cada uno con múltiples hilos.

Multihilo

- En un entorno multihilo, un proceso es una unidad de asignación de recursos y una unidad de protección.
- Dentro de un proceso puede haber uno o más hilos, cada uno con:
  - Un estado de ejecución,
  - Un contexto de hilo (que se almacena cuando el hilo no está en ejecución).
  - Una pila de ejecución.
  - Espacio de almacenamiento para variables locales.
  - Acceso a la memoria y recursos de su proceso, compartido con todos los hilos de su mismo proceso.

Diferencia entre proceso e hilo



- Multihilo: existe un solo BCP y un espacio de direcciones de usuario para el proceso, pero hay pilas separadas para cada hilo, y un BCP para cada hilo (registros, prioridad, etc).

Multihilo

- Todos los hilos de un proceso:
  - comparten el estado y los recursos de ese proceso.
  - residen en el mismo espacio de direcciones
  - tienen acceso a los mismos datos.

Beneficios de los hilos

Los mayores beneficios provienen de las consecuencias del rendimiento:

- Es más rápido crear un nuevo hilo en un proceso que crear un nuevo proceso (10-100 veces más rápido).
- Un hilo se finaliza más rápido que un proceso.
- Es más rápido cambiar entre dos hilos del **mismo** proceso.
- Mejoran la eficiencia de la comunicación entre diferentes programas.

Hilos

- Entonces, si se desea crear una aplicación como un conjunto de unidades de ejecución relacionadas, es mucho más eficiente hacerlo con un conjunto de hilos que con un conjunto de procesos independientes.
- Un ejemplo de aplicación que puede hacer uso de hilos es un servidor de archivos. Cada vez que se realiza una petición de archivos, el gestor de archivos puede ejecutar un nuevo hilo. Así en un ambiente **multiprocesador** se pueden ejecutar simultáneamente múltiples hilos del mismo proceso en diferentes procesadores. Y los hilos pueden compartir archivos de datos y coordinar acciones de manera más rápida.
- ¿Ejemplos? Hacer grupos y escribir la mayor cantidad de ejemplos sobre programas que utilizan hilos durante su ejecución.

¿Cómo se utilizan los hilos en un sistema multiprocesador?

- **Trabajo en primer plano y en segundo plano**: Un programa de hojas de cálculo (Excel) puede utilizar un hilo para mostrar menús mientras otro hilo ejecuta los mandatos de usuario y actualiza la hoja de cálculo. Así se puede empezar un nuevo mandato antes de terminar el anterior y se percibe una mayor velocidad de la aplicación.
- **Procesamiento asíncrono:** Un procesador de texto (Word) utiliza **tareas asíncronas** que pueden implementarse como hilos. Se puede crear un hilo cuyo único trabajo sea crear una copia de seguridad cada minuto y guardarla en disco.
- **Velocidad de ejecución:** un proceso multihilo puede computar una serie de datos mientras lee los siguientes datos de un dispositivo. Así mientras un hilo está bloqueado por una operación de E/S, otro hilo puede estar ejecutando.
- **Estructura modular de programas:** Los programas con varias tareas o con varias fuentes y destinos de E/S se pueden implementar más fácilmente usando hilos.

Hilos

- Debido a que todos los hilos de un proceso comparten el mismo espacio de direcciones que su proceso, al suspender el proceso, todos los hilos se suspenden al mismo tiempo. De igual manera cuando se finaliza un proceso, se finalizan todos los hilos de ese proceso.

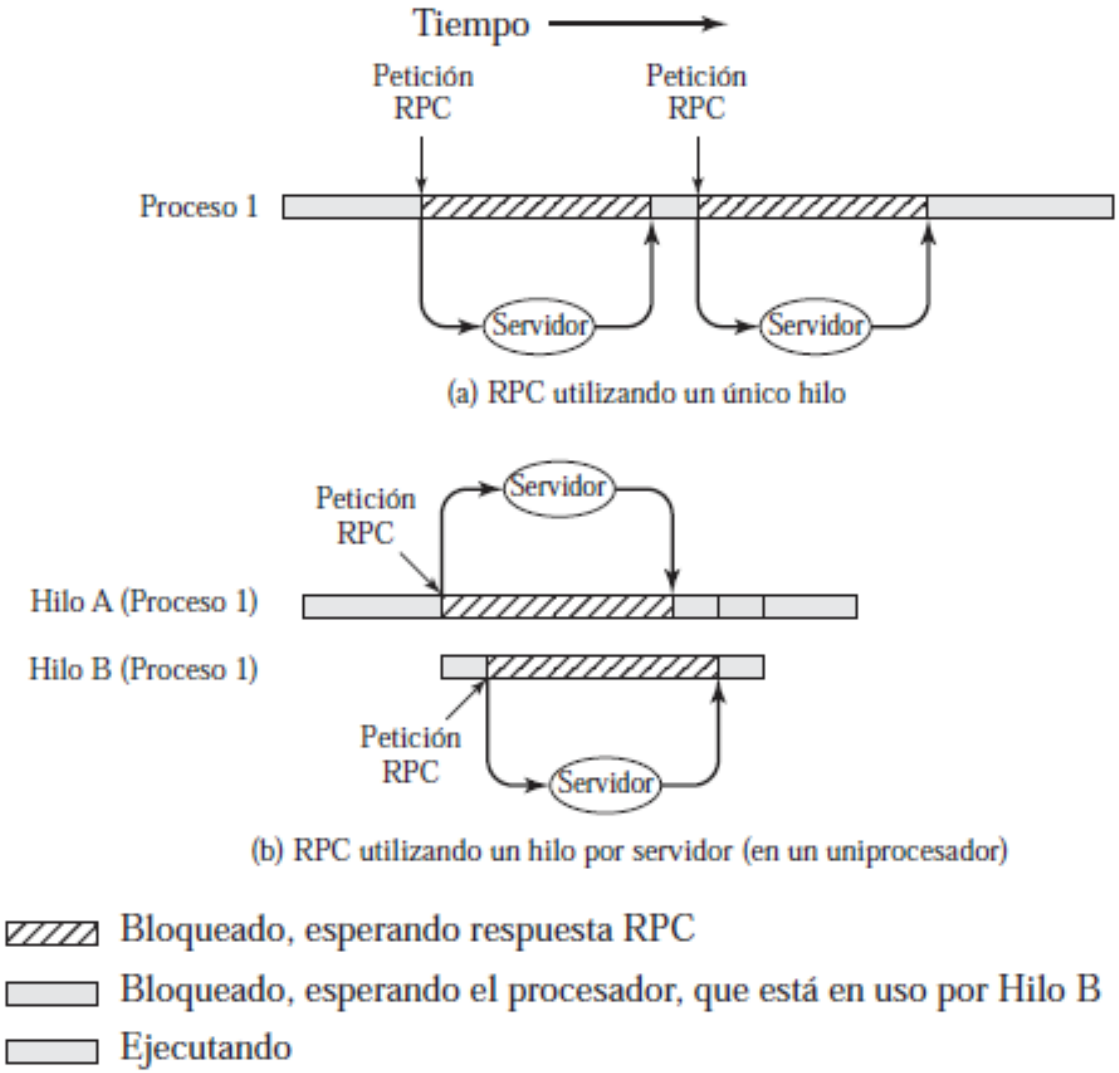
Contenido I

Funcionalidad de los Hilos

Estados de los hilos

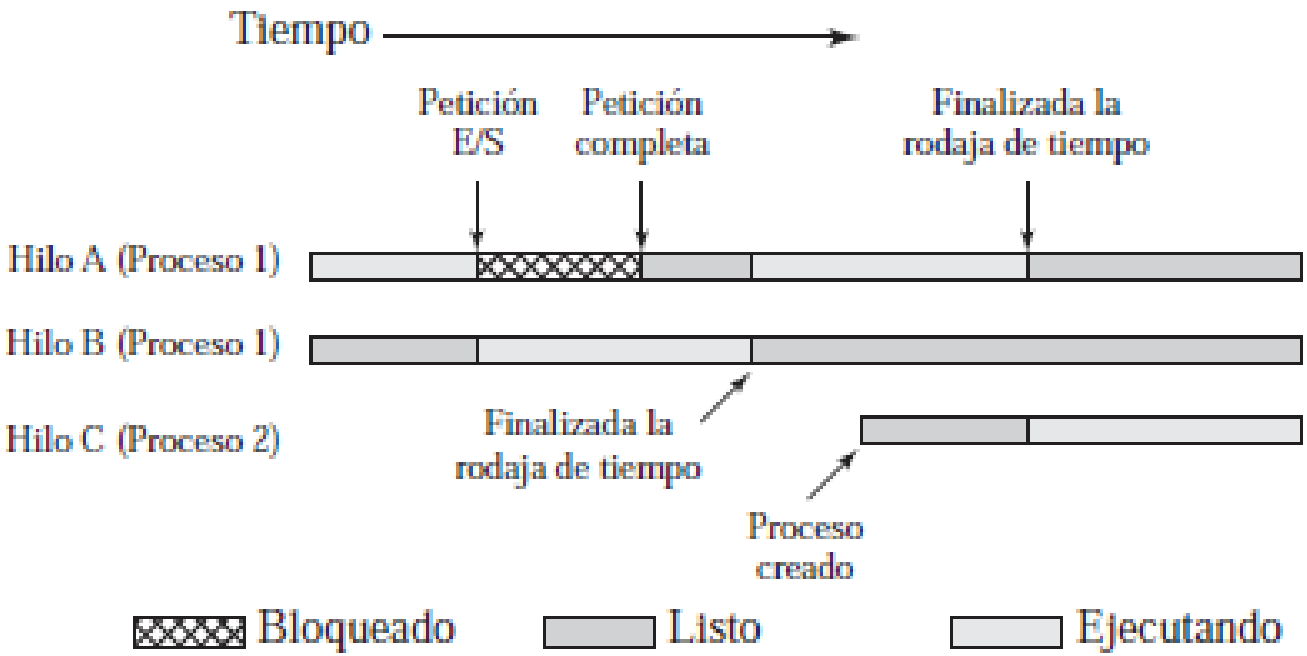
- Los principales estados de los hilos son Ejecutando, Listo y Bloqueado.
- El estado Suspendido se aplica solo a nivel proceso ya que si se expulsa un proceso de memoria, todos sus hilos deben también ser expulsados. Porqué? porque comparten el mismo espacio de direcciones.

Monohilo vs Multihilo



- Programa que realiza 2 RPC a dos máquinas diferentes y poder combinar resultados.
- en a los resultados se obtienen en secuencia.
- en b se mejora la velocidad de ejecución del programa.

Multihilo en uniprocesador con multiprogramación



- Multiprogramación permite **intercalado** de hilos de varios procesos pero nunca se llegan a ejecutar en paralelo.

Sincronización de hilos

- Debido a que todos los hilos de un proceso comparten recursos y espacio de direcciones, cualquier modificación de un recurso (archivo) por parte de un hilo, afecta el entorno de todos los hilos del mismo proceso.
- Por este motivo es necesario sincronizar las actividades de los hilos para que no interfieran entre ellos o corrompan estructuras de datos.
- Estudiaremos la sincronización de procesos e hilos más adelante.

Contenido I

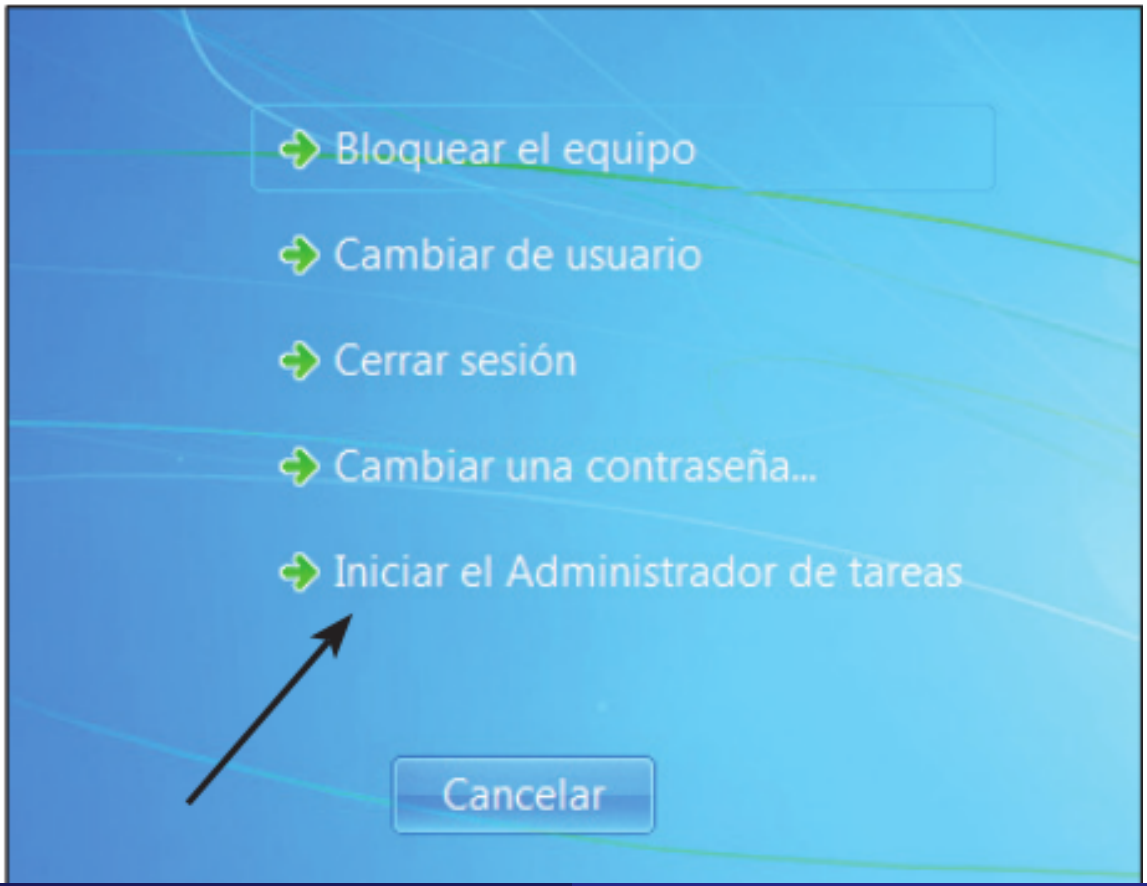
Hilos de nivel de usuario y de nivel núcleo

- **Hilos de nivel usuario (ULT):** la aplicación gestiona todo el trabajo de los hilos y el núcleo no es consciente de la existencia de los mismos. Se utilizan bibliotecas de hilos para gestionarlos (creación, suspensión, destrucción).
- **Hilos de nivel núcleo (KLT)**: El núcleo realiza todo el trabajo de gestión de hilos. No hay código de gestión de hilos en la aplicación, solamente una API para acceder a las utilidades de hilos del núcleo. Windows utiliza este enfoque.

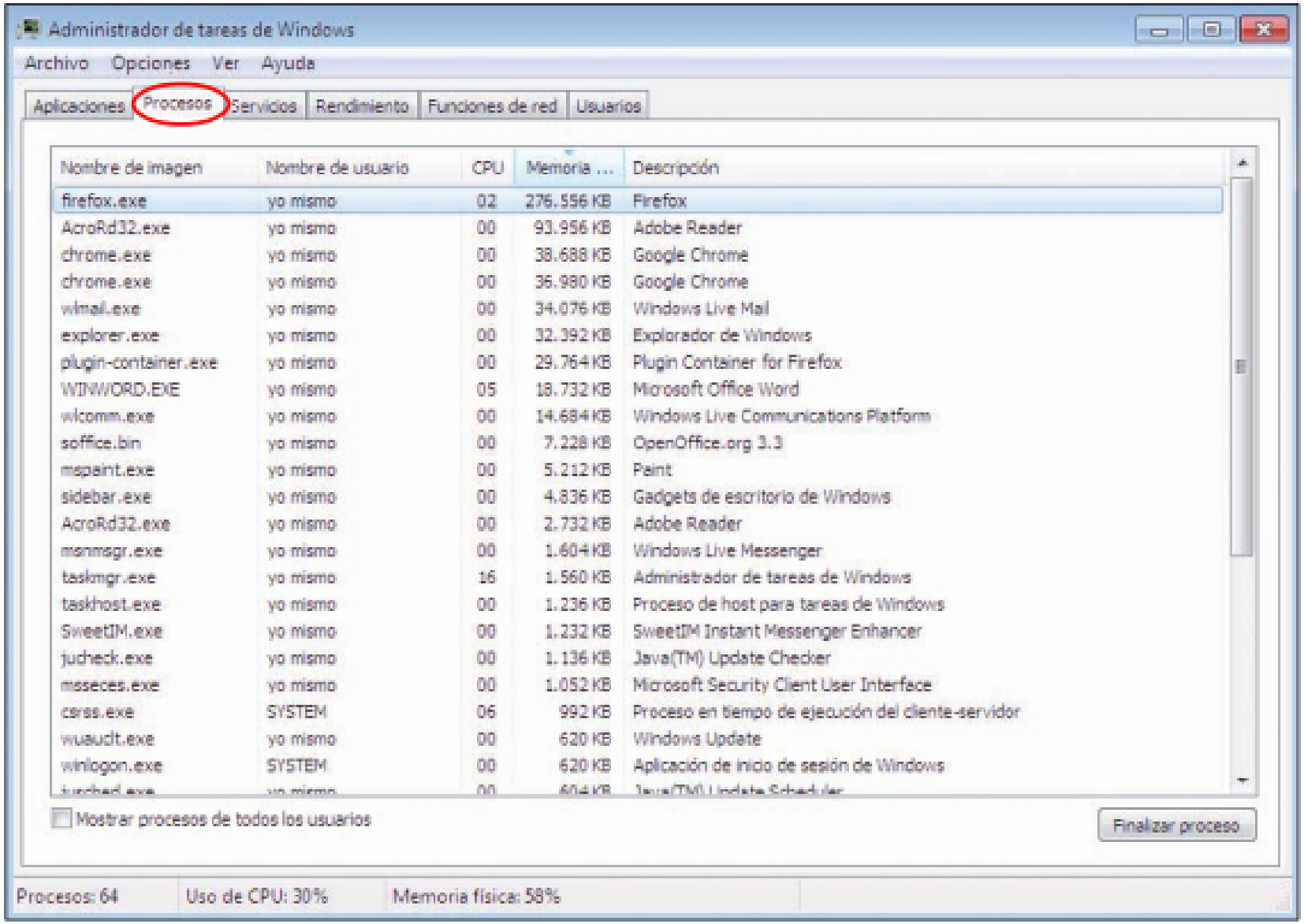
Contenido I

Acceder a procesos en Windows 7

- ¿Cómo accedemos a ver los procesos ejecutándose en Windows 7? `ctrl + alt + supr` **Administrador de tareas**.
- ¿Otras opciones en Windows 7?
- ¿Opciones en Windows 8?



Procesos en Windows



Procesos en Windows

- La pestaña de procesos nos mostrará:
  - los procesos en ejecución.
  - el usuario propietario
  - el trabajo del procesador
  - la cantidad de memoria
  - descripción
  - clic derecho en la cabecera permite seleccionar que aspectos mostrar.
  - Total de procesos activos (inferior izquierda)
  - clic derecho sobre proceso mostrará:
    - Terminar proceso
    - Establecer prioridad
    - Buscar en línea
    - Propiedades

Procesos en Ubuntu

- ¿Cómo abrimos el administrador de procesos en Ubuntu? Varias opciones: Información similar que en Win.
  - Buscar: Monitor del sistema
  - Buscar/Icono de aplicaciones/monitor del sistema
  - Instalar "indicador de carga del sistema"