PROCESOS Y HEBRAS

2.1 Generalidades sobre Procesos, Hilos y Planificación.

Ejecución del SO

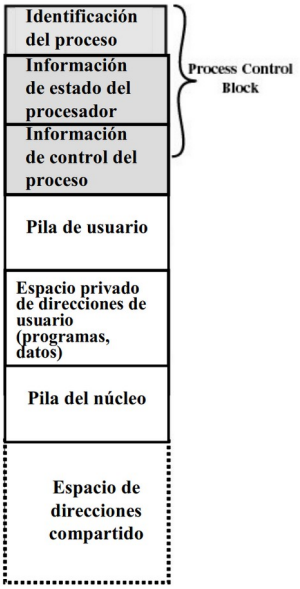
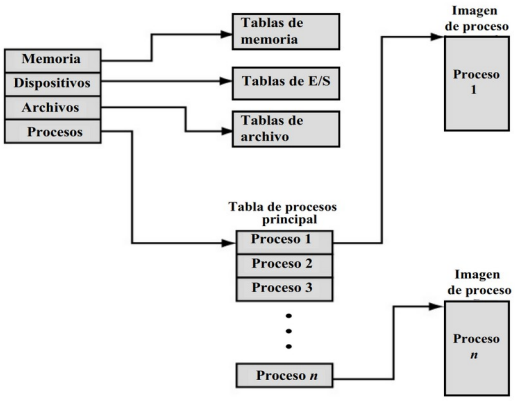


Imagen de un proceso: el sistema operativo se ejecuta dentro del proceso de usuario

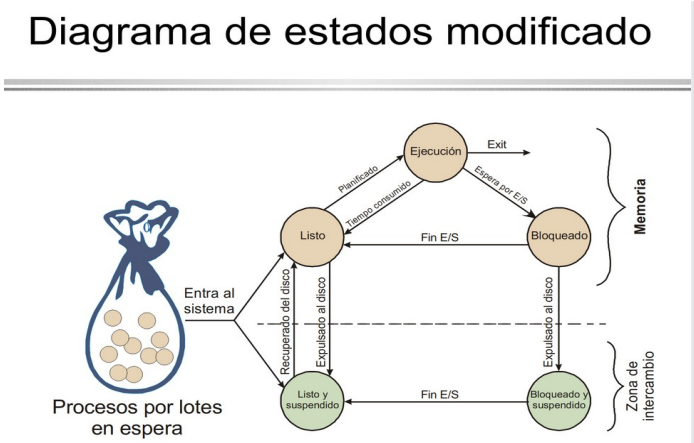
- Núcleo fuera de todo proceso: el núcleo del SO se ejecuta fuera de cualquier proceso. El código del mismo se ejecuta como una entidad separada en modo privilegiado.
- Ejecución dentro de los procesos de usuario: El software del SO se ejecuta en el contexto de un proceso de usuario. La ejecución de un proceso se lleva a cabo en modo privilegiado cuando se ejecuta el código del SO.

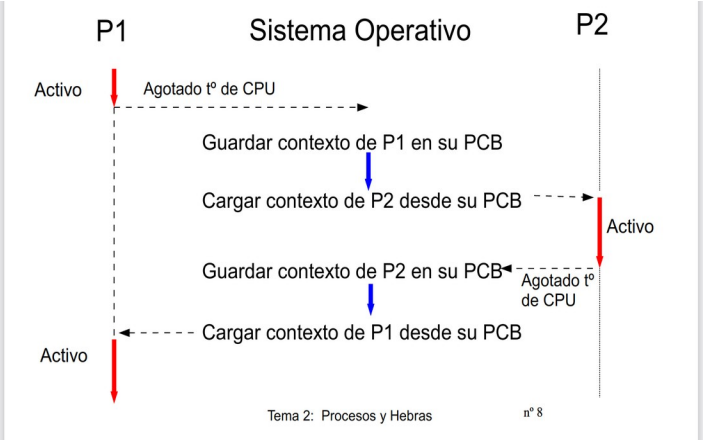


Estructura general de las tablas de control del sistema operativo.

Cambio de Contexto

Cuando un proceso está ejecutándose , el puntero de pila, los registros, el PC están cargados en la CPU; los registros hardware contienen los valores actuales. En el momento en el que el SO detiene un proceso en ejecución, salva los valores actuales de los registros en el PCB del proceso.
La acción de conmutar la CPU de un proceso a otro se denomina **cambio de contexto**. Los sistemas de tiempo compartido hacen entre 10-100 cambios de contexto / segundo.





Operaciones sobre procesos. Creación de procesos

- ¿Crear un proceso? → Asignar el espacio de direcciones que utilizará el proceso, y crear las estructuras de datos para su administración.
- ¿Cuándo se crean?
  - ➔ En sistemas batch (procesamiento por lotes), como respuesta a la recepción y admisión de un trabajo.
  - ➔ En sistemas interactivos, cuando el usuario se conecta, el SO crea un proceso que ejecuta el interprete de órdenes.
  - ➔ El SO puede crear un proceso para llevar a cabo un servicio solicitado por un proceso de usuario.
  - ➔ Un mismo proceso puede crear otros procesos formando un árbol de procesos (relación padre-hijo.
- Cuando un proceso crea a otro, el proceso hijo obtiene sus recursos directamente del SO, pues padre e hijo no comparten recursos o los comprate todos con el padre, o un subconjunto de ellos.
- Ejecución → tanto el padre como el hijo se ejecutan concurrentemente, y el padre espera a que termine el hijo.
- Espacio de direcciones → el hijo es duplicado del padre (Unix, Linux). El hijo tiene un programa que lo carga (VMS, W2K). Por ejemplo, en UNIX, la llamada al sistema **fork** crea un nuevo proceso, y, con la orden **exec** después de *fork* reemplaza el e.direcciones con el programa del proceso nuevo.

Pasos a seguir en la creación de procesos:
→ *Nombrar al proceso: asignarle un PID único*
→ *Asignarle espacio (en MP o en m. secundaria*
→ *Crear el PCB e inicializarlo*
→ *Insertarlo en la Tabla de procesos y ligarlo a la cola de planificación correspondiente*
→ *Determinar su prioridad inicial*

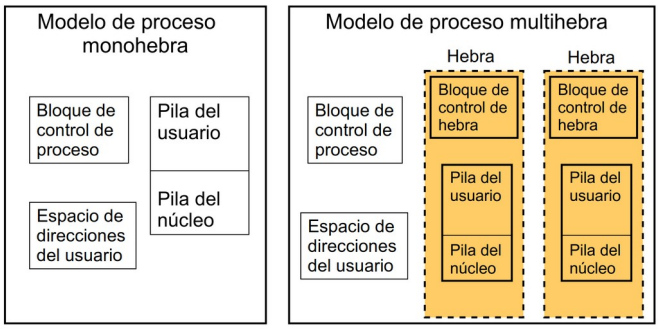
Terminación de procesos

Cuando un proceso ejecuta la última instrucción, solicita al SO su finalización (**exit)**: envío de datos del hijo al padre, y el SO libera los recursos del proceso.
El padre puede finalizar la ejecución de sus hijos (**abort** o **kill**):

- El hijo ha sobrepasado los recursos asignados.
- La tarea asignada al hijo ya no es necesaria.
- El padre va a finalizar: el SO no permite al hijo continuar (**terminación en cascada**).
- El SO puede terminar la ejecución de un proceso porque se hayan producido errores o condiciones de fallo.

Hebras

Una **hebra** o proceso ligero es la unidad básica de utilización de la CPU que consta de: un PC, un conjunto de registros, un espacio de pila y un estado. Una hebra comparte con sus hebras pares una *tarea* que consiste bien en sección de código, bien en sección de datos, o bien en recursos del SO (archivos abiertos, señales…).



Un *proceso pesado* o tradicional es igual a una tarea con hebra.

Ventajas de las hebras

Mayor rendimiento y mejor servicio debido a:

- Reducción del tiempo de cambio de contexto, el tiempo de creación y el de terminación.
- En una tarea con múltiples hebras, mientras una está bloqueada y esperando, otra hebra de la misma tarea puede ejecutarse (dependiendo del tipo de hebras).
- La comunicación entre hebras de una misma tarea se realiza a través de la memoria compartida (no se necesitan utilizar los mecanismos del núcleo).
- Las aplicaciones que necesitan compartir memoria se benefician de las hebras.

Funcionalidad de las hebras

Estados de las hebras: Ejecución, Lista o Preparada y Bloqueada.
Operaciones básicas relacionadas con el cambio de estado en hebras: creación, bloqueo, desbloqueo, terminación.
Sincronización entre hebras.

Tipos de hebras

- **Hebras de usuario:** Todo el trabajo de las hebras lo realiza la aplicación (el núcleo no es consciente de la existencia de las hebras). Se implementan a través de una biblioteca en el nivel usuario. La biblioteca contiene código para gestionar las hebras (crearlas, intercambiar datos entre hebras, planificar la ejecución de las hebra y salvar y restaurar el contexto de las mismas). La unidad de planificación para el núcleo es el proceso.
- **Hebras Kernel:** Toda gestión de hebras lo realiza el núcleo. El SO proporciona un conjunto de llamadas al sistema similares a las existentes para los procesos. El núcleo mantiene la información de contexto del proceso como un todo y de cada hebra. La unidad de planificación es la hebra y las propias funciones del núcleo pueden ser multihebras.

VENTAJAS HEBRAS USUARIO VS KERNEL

→ Se evita sobrecarga de cambios de modo, que sucede cada vez que se pasa el control de una hebra a otra en sistemas que usan hebras núcleo.
→ Se puede tener una planificación para las hebras distintas a la planificación subyacente del SO.
→ Se pueden ejecutar en cualquier SO. Su utilización no supone cambio en el núcleo.

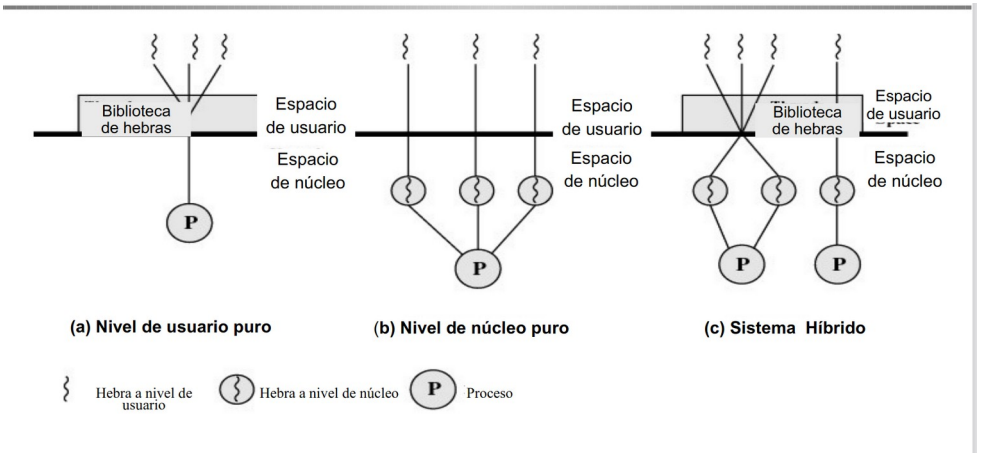
DESVENTAJAS HEBRAS USUARIO VS KERNEL

→ Cuando un proceso realiza una llamada al sistema, no sólo se bloquea la hebra que realizó la llamada, sino todas las hebras del proceso.
→ En un entorno multiprocesador, una aplicación multihebra no puede aprovechar las ventajas de dicho entorno ya que el núcleo asigna un procesador a un proceso.

Enfoques híbridos

Implementan tanto hebras a nivel kernel como usuario. La mayor parte de la planificación y sincronización, así como la creación de hebras, se realizan en el espacio de usuario. Las distintas hebras de una aplicación se asocian con varias hebras del núcleo (<= número). El programador

puede ajustar la asociación para obtener un mejor resultado. Las hebras de una aplicación se pueden paralelizar y las llamadas al sistema bloqueadoras no necesitan bloquear todo el proceso.



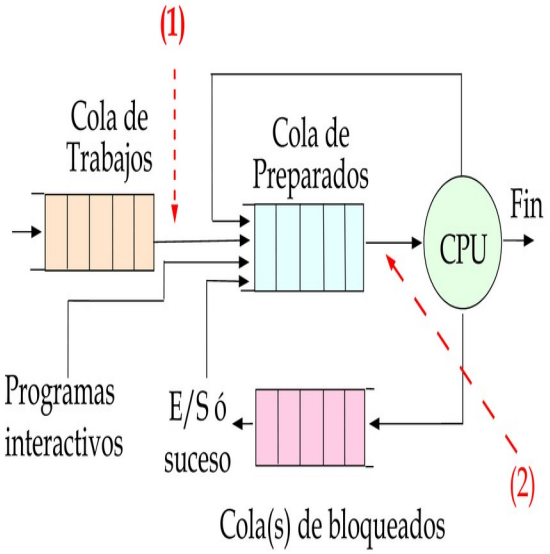
Planificación: PCB's y Colas de Estados

El sistema operativo mantiene una colección de colas que representan el estado de todos los procesos en el sistema. Suele haber una cola por estado. Cada uno de los PCB está encolado en una cola de estado acorde a su estado actual. Conforme un proceso cambia de estado, su PCB es retirado de una cola y encolado en otra.

Colas de estados

- **Cola de trabajos:** Conjunto de los trabajos pendientes de ser admitidos en el sistema (básicamente, los trabajos por lotes que no están en memoria).
- **Cola de preparados:** Conjunto de todos los procesos que residen en memoria principal, preparados y esperando para ejecutarse.
- **Cola(s) de bloqueados:** Conjunto de todos los procesos esperando por un dispositivo de E/S particular o por un suceso.

Planificadores



Un planificador es una parte del SO que controla la utilización de un recurso. Dentro de los diferentes planificadores de CPU encontramos:

- **A largo plazo** (planificador de trabajos): selecciona los procesos que deben llevarse a la cola de preparados. Permite controlar el grado de multiprogramación. Se invoca poco, por lo que puede ser más lento.
- **A corto plazo** (planificador de la CPU): selecciona al proceso que debe ejecutarse a continuación, y le asigna la CPU. Trabaja con la cola de preparados y se invocan muy frecuentemente por lo que debe ser rápido.
- A medio plazo

Clasificación de procesos

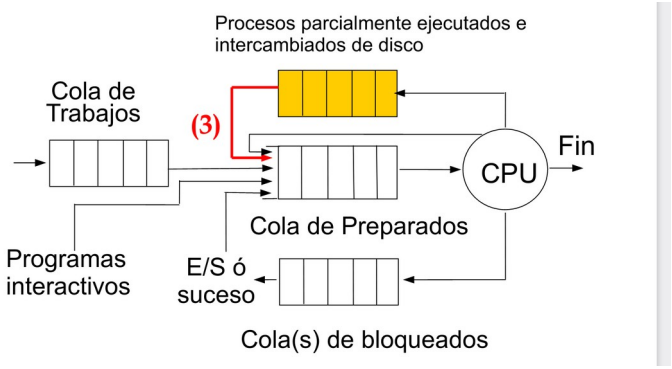
- **Limitados por E/S o procesos cortos:** dedican más tiempo a realizar E/S que cómputo; muchas ráfagas de CPU cortas y largo periodos de espera.
- **Limitados por CPU o procesos largos:** dedican más tiempo en computación que en realizar E/S; pocas ráfagas de CPU pero largas.

Mezcla de trabajos

Es importante que el planificador a largo plazo seleccione una buena mezcla de trabajos, porque si todos los trabajos están limitados por E/S, la cola de preparados estará casi siempre vacía y el planificador a corto plazo tendrá poco que hacer. Además, porque si todos los procesos están limitados por CPU, la cola de E/S estará casi siempre vacía y el sistema estará de nuevo desequilibrado.

Planificador a medio plazo

En algunos SO's, como por ejemplo uno de tiempo compartido, a veces es necesario sacar procesos de la memoria, es decir, reducir el grado de multiprogramación, bien para mejorar la mezcla de procesos, bien por cambio en los requisitos de memoria, y luego volverlos a introducir (**intercambio o swapping).** El planificador a **medio plazo** se encarga de devolver los procesos a memoria.



Despachador

Es un módulo del SO que da el control de la CPU al proceso seleccionado por el planificador a corto plazo, lo cual involucra un cambio de contexto (se realiza en modo kernel), la conmutación a modo usuario y un salto a la posición de memoria adecuada del programa para su reanudación.

Latencia de despacho: Tiempo que emplea el despachador en detener un proceso y comenzar a ejecutar otro.

Activación del Despachador: Puede actuar cuando:

- Un proceso no quiere seguir ejecutándose (finaliza o ejecuta una operación que lo bloquea).
- Un elemento del SO determina que el proceso no puede seguir ejecutándose (ej. E/S o retiro de memoria principal).
- El proceso agota el quantum de tiempo asignado.
- Un suceso cambia el estado de un proceso de bloqueado a ejecutable.

Políticas de planificación: Monoprocesadores

Sea t = tiempo de servicio (ráfaga) de un proceso P

- **Tiempo de respuesta (T):** tiempo transcurrido desde que se remite una solicitud (entra en cola de preparados) hasta que se produce la primera respuesta (no se considera el tiempo que tarda en dar la salida).
- **Tiempo de espera (M):** tiempo que un proceso ha estado esperando en la cola de preparados: T-t
- **Penalización (P)**: T/t
- **Índice de repuesta (R):** t/T: fracción de tiempo que P está recibiendo servicio.
- **Tiempo de núcleo:** tiempo perdido por el SO tomando decisiones que afectan a la planificación de proceso y haciendo los cambios de contexto necesarios. En un sistema eficiente debe representar entre el 10% y el 30% del total del tiempo del procesador.
- **Tiempo de inactividad:** tiempo en el que la cola de ejecutables está vacía y no se realiza ningún trabajo productiivo.
- **Tiempo de retorno:** cantidad de tiempo necesario para ejecutar un proceso completo.