PROCESOS

ESTRUCTURAS DE DATOS

BCP

PROCESOS

OPERACIONES SOBRE LOS PROCESOS

ESTADOS DE LOS PROCESOS

PLANIFICACIÓN DE LA CPU

ALGORITMOS NO APROPIATIVOS

ALGORITMOS APROPIATIVOS

EVALUACIÓN DE LAS POLÍTICAS

PLANIFICACIÓN EN TIEMPO REAL

ESTRUCTURAS BÁSICAS DE UN S.O.

Como dijimos en el tema introductorio, un S.O. es un programa que tiene como función la asignación de elementos físicos a los elementos lógicos. Para poder llevar a cabo esta función, el S.O. mantiene una serie de informaciones acerca de esos objetos lógicos y físicos. El soporte de esta información se conoce como ***Bloque de Control del Sistema*** (SCB).

- **Bloque de control del sistema**

  Estructura básica para almacenar los datos necesarios para poder ejecutar el sistema operativo:

  – Lista de descriptores de procesos.
  – Puntero al descriptor del proceso que está haciendo uso del procesador.
  – Puntero a la cola de descriptores de procesos que no hacen uso del ordenador, pero están en espera.
  – Punteros a colas de procesos que se encuentran en diferentes situaciones.
  – Identificadores de las rutinas necesarias para tratar las interrupciones producidas por el hardware, el software o errores indeseados.
  – Puntero a la cola de descriptores de recursos

BLOQUE DE CONTROL DE PROCESOS

- Cada proceso se representa en el **S.O.** por un conjunto de datos, que incluye toda la información necesaria para definirlo: el estado, recursos utilizados, registros.
- Este conjunto de datos se conoce como **bloque de control de procesos (PCB),** y es toda la información que el S.O. necesita para ejecutar el programa. Es el **proceso** para el **S.O**..
- Estos datos pueden estar en la memoria principal o en el disco, pero los que son necesarios para determinar en que situación se encuentra el proceso deben estar en memoria.
- Estas informaciones se almacenan en lo que se conoce como **Bloque de Control de Procesos (BCP).**
- La **Tabla de Procesos** es una estructura, generalmente estática por razones de eficacia, cuyas entradas son **bloques de control de procesos**.
- El **Bloque de Control de Procesos** contiene la información básica de cada proceso y que podemos definir aproximadamente:

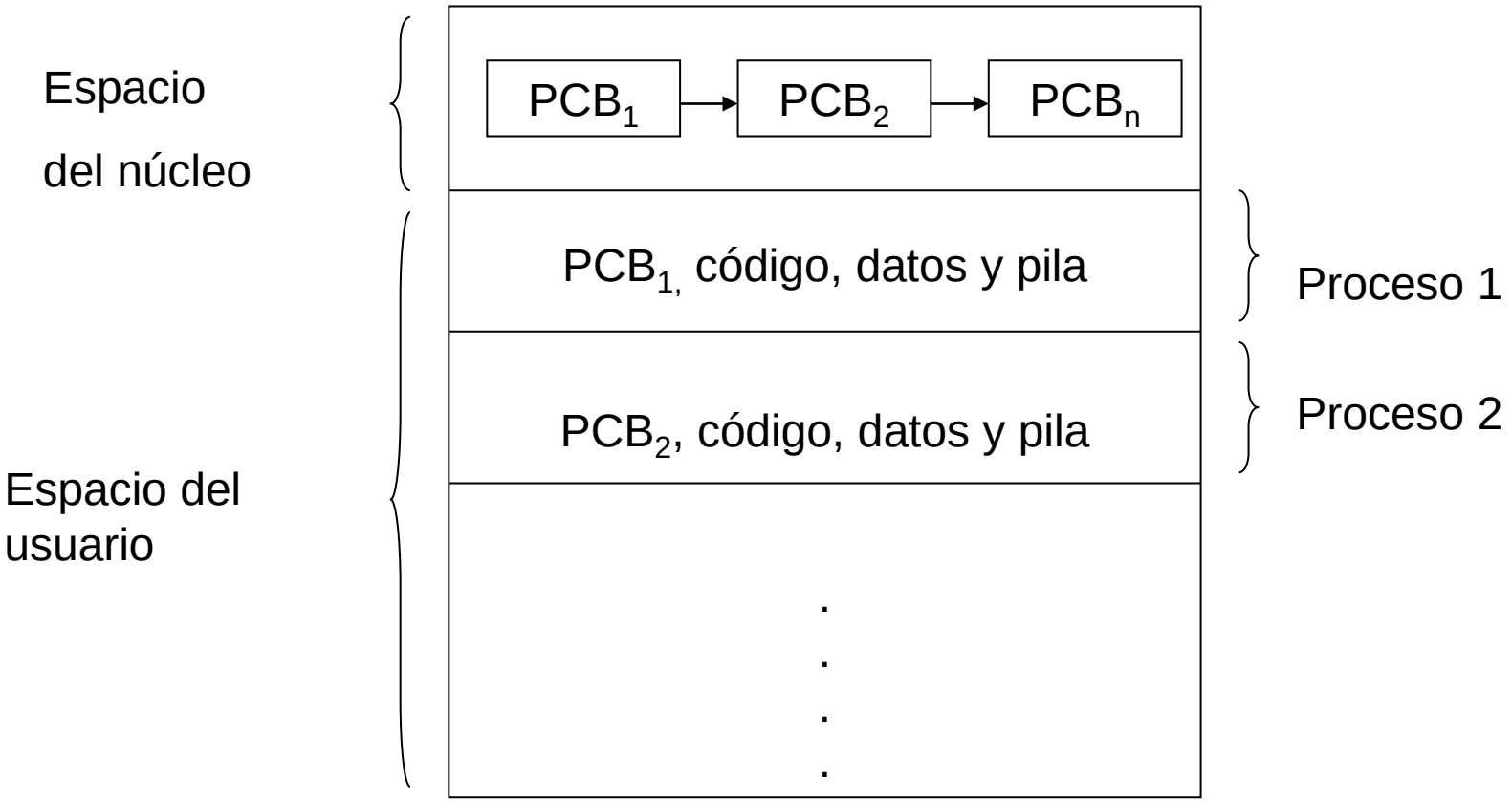
BLOQUE DE CONTROL DE PROCESOS

- Información de identificación
  - Identificador del proceso.
  - Identificador del proceso padre en caso de existir relaciones padre-hijo (UNIX).
  - Información sobre el usuario (identificador del usuario, grupo)
- Estado del procesador
- Información del control del proceso. Información diversa como:
  - Información de planificación y estado:
    - Estado del proceso
    - Evento por el que espera el proceso cuando está bloqueado.
    - Prioridad del proceso.
    - Información de planificación.
  - Descripción de los segmentos de memoria asignados al proceso.
    - Puntero al segmento de datos.
    - Puntero al segmento de código.
    - Puntero al segmento de pila.
  - Recursos asignados, tales como:
    - Archivos abiertos (tabla de descriptores o manejadores de archivo).
    - Puertos de comunicación asignados
- Punteros para estructurar los procesos en colas o anillos.
- Comunicación entre procesos. El BCP puede contener espacio para almacenar las señales y para algún mensaje enviado al proceso.

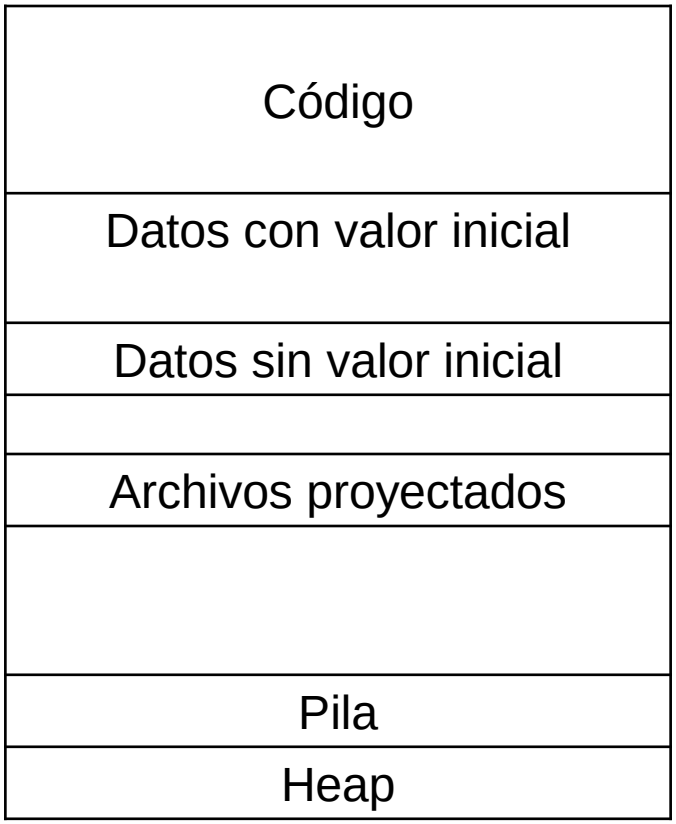
CONCEPTO DE PROCESO

- Todo lo que sucede dentro de un ordenador está determinado por la ejecución de procesos, quiere esto decir que se trata del objeto básico y fundamental. Podemos definir un proceso como un programa en ejecución o en disposición de ser ejecutado cuando las circunstancias que rodean el mismo así lo determinen.
- Un programa es un código objeto reposando en memoria, es decir, sin ser activado, pero un proceso es este objeto pero activado.
- La activación de este código implica la carga en memoria de parte o todo él para su ejecución y la creación por parte del S.O. de un conjunto de informaciones que permitan controlar la ejecución y desarrollo del mismo (**PCB**).
- El entorno definido por el contexto de hardware, de software y la imagen en ejecución representan un concepto dinámico que se conoce como **proceso**.
- Un **proceso** es pues un programa en ejecución, incluyendo el código o instrucciones que lo componen, el contador de programa, los registros y las variables, es decir, contiene toda la información relativa al entorno en donde se ejecuta.

UBICACIÓN EN MEMORIA



MAPA DE MEMORIA



OPERACIONES SOBRE LOS PROCESOS

- **Crear**: puede hacerse desde un proceso ya existente o a través del intérprete de comandos del S.O. en cualquier caso se considera hijo del proceso creador. El trabajo para el S.O. consiste en darle una entrada en el **PCB** y pasarlo a la cola de preparados. Hay varias formas de crear procesos:
    - **Inicialización del sistema.** Cuando se inicia el sistema se crean varios procesos, algunos interactúan con el usuario y otros no (demonios)
    - **Ejecución de una llamada al sistema para crear procesos por parte de un proceso en ejecución.** Mediante una llamada al sistema. Por ejemplo para bajar datos es conveniente crear un proceso para que los ponga en el buffer.
    - **Solicitud de un usuario para crear un proceso.** En modo interactivo es muy común por parte de los usuarios.
    - **Inicio de un trabajo por lotes.**

OPERACIONES SOBRE LOS PROCESOS (2)

– En cualquier caso la creación de procesos se hace mediante una llamada al sistema (**fork** en UNIX y **execve**). Los pasos de creación de procesos en UNIX es el siguiente:

  - Un proceso especial llamado ***init*** , está presente en la imagen de arranque. Cuando dicho proceso comienza a ejecutarse lee un archivo que indica cuantas terminales hay, y genera un proceso nuevo para cada una. Estos procesos esperan a que alguien inicie sesión. Si hay inicio de sesión ***login*** exitoso, el proceso ***login*** ejecuta un ***shell*** para aceptar comandos. Estos pueden generar más procesos, y así de forma sucesiva. Por lo tanto todos los procesos del sistema pertenecen a un árbol que tiene a ***init*** como raíz.

OPERACIONES SOBRE LOS PROCESOS (3)

EXEMPLO FORK()

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

main()
{
  int a,val=2;
  a=fork();
  switch(a)
  {
    case -1: break;          /*erro */
    case 0: val--;break;     /*proceso fillo */
    default: val++;          /*proceso pai */
  }
  printf("\n val= %d\n",val);
}
```

OPERACIONES SOBRE LOS PROCESOS (4)

EXEMPLO EXECVE()

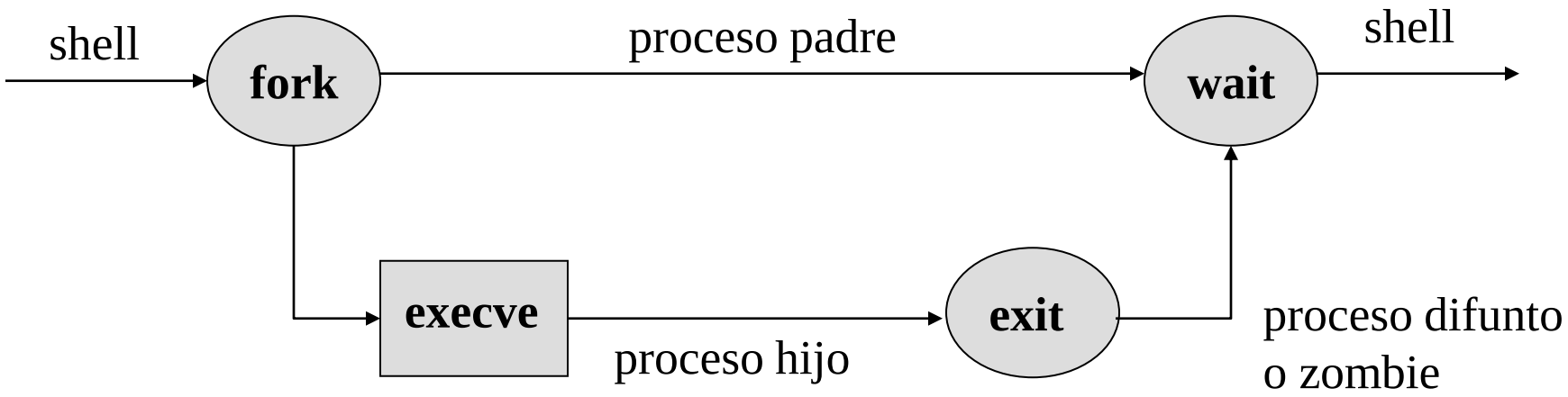
```
#include <stdio.h>
#include <unistd.h>

main()
{
  int error;
  char *arg[3],*argp[1];

  arg[0]="ls";    /* argumentos */
  arg[1]="-l";
  arg[2]=NULL;
  argp[0]=NULL;   /* entorno */
  execve("/bin/ls",arg,argp);
  printf("\n error \n");
}
```

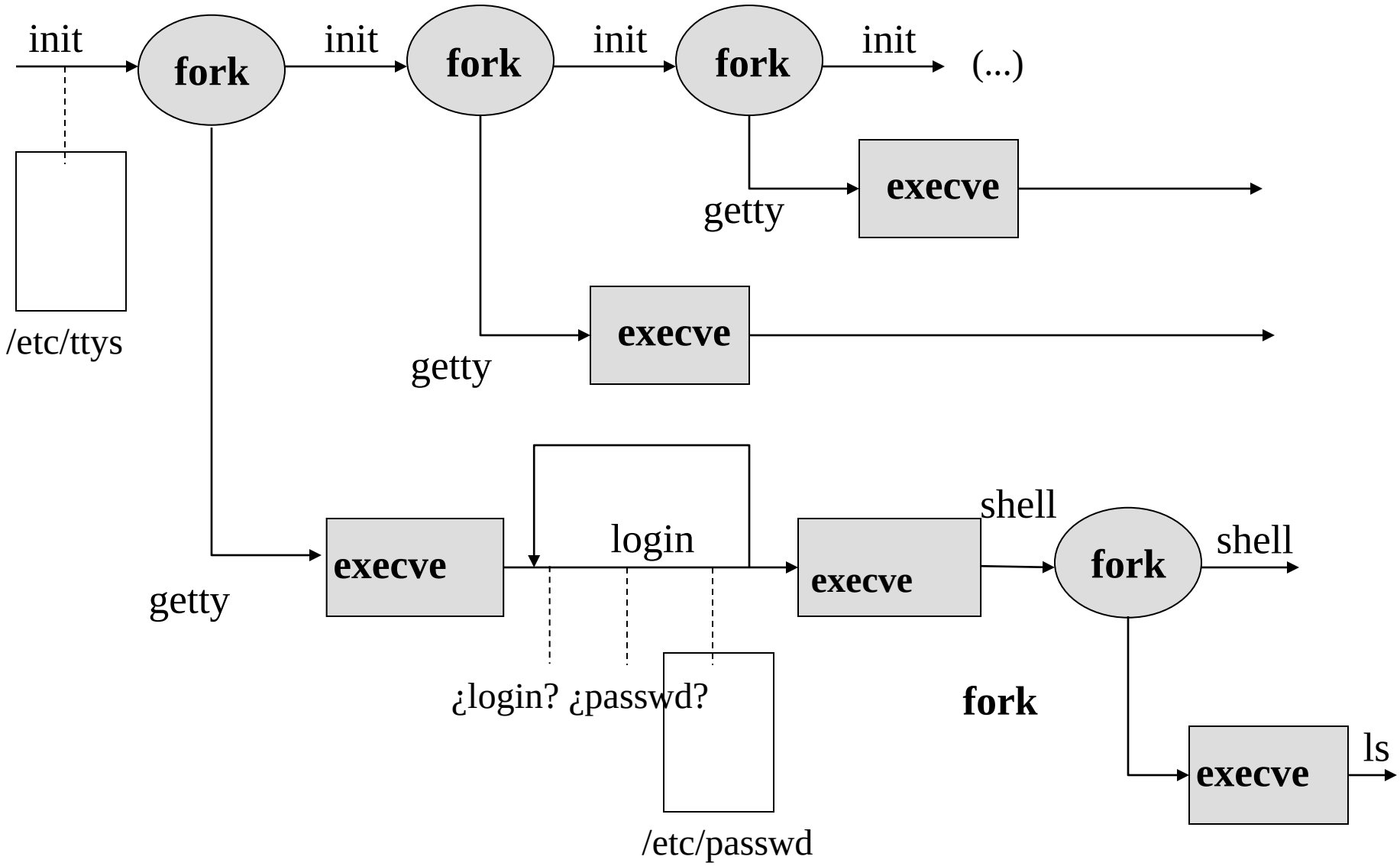
OPERACIONES SOBRE LOS PROCESOS (5)

- **execve** es usado por un proceso para cargar un nuevo ejecutable binario en el espacio virtual de memoria del proceso que hace la llamada
- **vfork** es igual que fork, pero **no copia** los **datos** ni la **pila**
- **Secuencia** 'normal' en la creación de un proceso:



OPERACIONES SOBRE LOS PROCESOS (5)

JERARQUÍA DE PROCESOS



OPERACIONES SOBRE LOS PROCESOS (6)

- **Destruir**: eliminar la entrada en la cola de **PCB**. Puede haber problemas en la gestión de las propiedades heredadas del proceso padre o, si tiene procesos hijo, tener que esperar a que finalicen estos o los finaliza forzosamente.
  - **Terminación normal.** Es la forma mas normal (exit en UNIX)
  - **Terminación por error.** Por ejemplo gcc uno.c y el fichero uno.c no existe.
  - **Error fatal.** Acceso a una posición no permitida, división por cero … etc.
  - **Terminado por otro proceso.** En UNIX es KILL.
- **Cambiar la prioridad del proceso.**
- **Dormir o bloquear la ejecución de un proceso**. Dormir un proceso un tiempo.
- **Despertar un proceso**. Una forma de desbloquear un proceso de forma artificial. Se suele emplear para procesos dormidos artificialmente.
- **Suspender un proceso**. Suele hacerse en situaciones de sobrecarga del S.O.
- **Reanudar un proceso**. Activar un proceso suspendido.

TIPOS DE PROCESOS

- **Según sea su diseño:**
  - **Reutilizables**: se pueden utilizar todas las veces que se desee. Cada vez que se ejecuta es necesario cargarlos en memoria. Los programas de usuario suelen ser de este tipo.
  - **Reentrantes**: no tienen asociados datos, sólo código. Sólo se carga una copia en memoria y esta es compartida por todos los usuarios que la precisen. No obstante, para cada usuario se crea un proceso, donde se recoge específicamente la zona de memoria que recoge los datos, diferente para cada proceso.
- **Según la capacidad de los procesos para acceder al procesador y a los recursos:**
  - **Apropiativos**: acceden a los recursos y sólo los abandona de forma voluntaria (CPU).
  - **No apropiativos**: permiten que otros procesos pueden apropiarse de los recursos que ahora poseen.
- **Desde el punto de vista de la ejecución:**
  - **Residentes**: permanecen íntegramente en memoria durante su ejecución.
  - **Intercambiables (swappables)**: pueden ser llevados a disco durante su ejecución a voluntad del S.O.
- **Otra clasificación que no siempre existe en todo S.O.:**
  - **Privilegiados**: se ejecutan de modo supervisor.
  - **No privilegiados**: son los que normalmente ejecuta el usuario.
- **Según los propietarios de los procesos:**
  - **Procesos de usuario**: son los diseñados por los usuarios. Se ejecutan en modo no protegido.
  - **Procesos del sistema**: el S.O. realiza la planificación de los procesos de usuario y ciertas operaciones para los mismos (E/S).

PRIORIDADES

Podemos clasificarlas del siguiente modo:

- Asignadas por el sistema operativo:
  - asignadas racionalmente: generalmente asignadas en función de los privilegios del propietario.
  - Asignadas arbitrariamente: a medida que llegan sin tener en cuenta ningún factor.
- Asignadas por el propietario: suele utilizarse en los sistemas en tiempo real

INTERRUPCIONES

completar con la materia del tema 1

- El **SCB** tiene las direcciones de las rutinas que se deben ejecutar ante la presencia de un evento.
- Utilidad:
  - En multiprogramación permite al sistema operativo tomar el control del procesador si se produce un error.
  - Notificar al procesador la finalización de una operación de E/S
  - Establecer periodos de tiempo a los procesos.
  - Reconocer eventos externos.

Nivel	Evento Software
0	Proceso de usuario
1	Planificación de procesos
2	Temporización
3 a 10	Drivers de entrada/salida
11 a 15	Otros

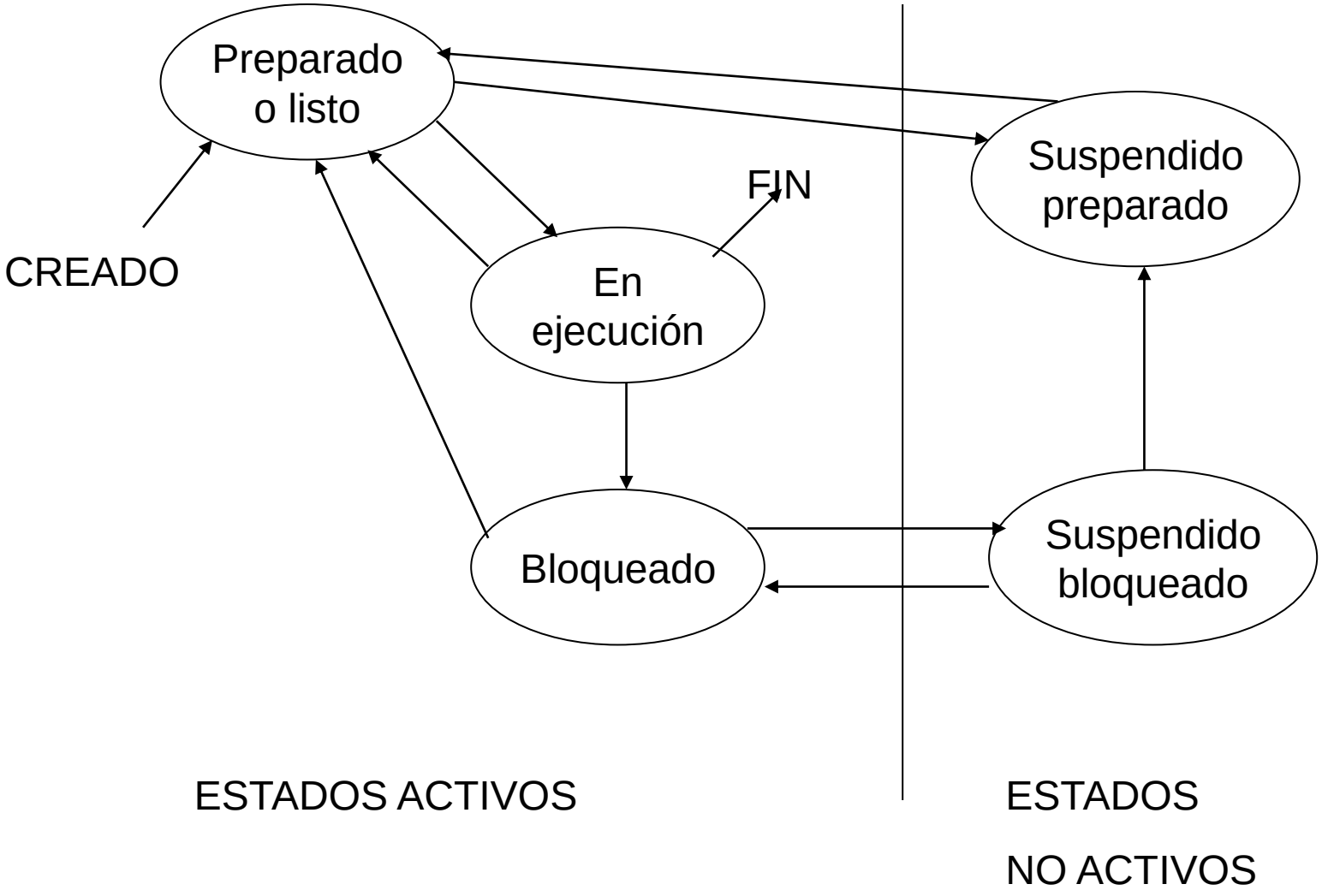
Nivel	Evento Hardware
16 a 23	Interrupciones de dispositivos
24	Reloj interno
25 a 29	Errores de: – Procesador – Memoria – Buses
30	Fallo de tensión
31	Pila (stack) errónea del núcleo

ESTADOS DE LOS PROCESOS

Podemos dividirlos en activos e inactivos

- **ACTIVOS:** compite por el procesador o está en condiciones de poder hacerlo
  - **Ejecución**: proceso que tiene el control de la CPU.
  - **Preparado**: está dispuesto para entrar en la CPU cuando el sistema así lo determine.
  - **Bloqueado**: no pueden ejecutarse porque han realizado una operación que exige una espera. Por ejemplo una operación de E/S.
- **INACTIVOS:** Estado en el que se sitúan los procesos que no pueden competir por el procesador.
  - **Suspendido bloqueado**: procesos suspendidos mientras se esperaba un evento, y las causas de su bloqueo no han desaparecido.
  - **Suspendido preparado**: procesos suspendidos pero que no tienen causas para estar bloqueados

TRANSICIONES DE ESTADO

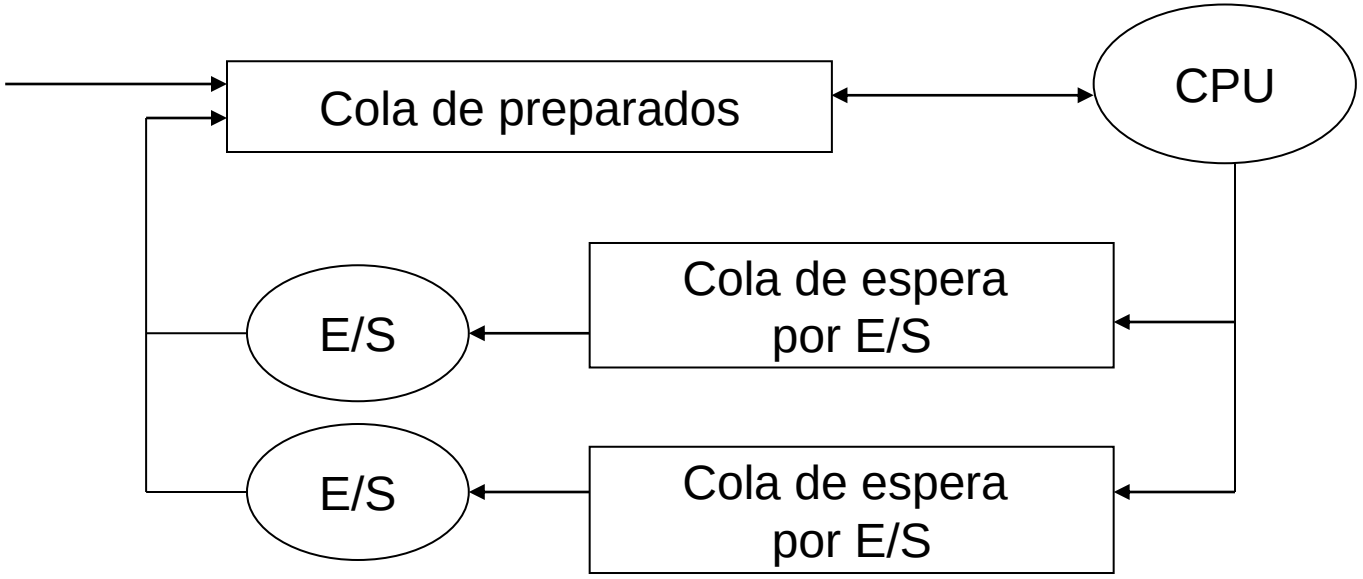


TRANSICIONES DE ESTADO

- **Comienzo de la ejecución**: se inserta en la cola de preparados.
- **Paso a estado de ejecución**: el que se encuentra primero en la cola pasa a ejecutarse cuando el reloj haya interrumpido el que estaba en ejecución.
- **Paso a estado bloqueado**: un proceso que está en ejecución y pasa a realizar una operación de E/S pasa a la cola de bloqueados. Una segunda posibilidad es que un proceso se haya suspendido estando bloqueado y en el momento de la reanudación no haya desaparecido la causa de la suspensión.
- **Paso a estado preparado**: puede ser por cuatro causas:
  - Ejecución de un programa.
  - Fin de una operación de E/S. Pasa de bloqueado a estar preparado porque finalizó la causa por la que estaba bloqueado.
  - Interrupción. Si se produce una interrupción que fuerza a que se corte un proceso, este pasa a preparado. Finalización del quantum.
  - Activación. Un proceso suspendido pero que no estaba bloqueado, en el momento de activarlo de nuevo pasa a la cola de preparados.
- **Paso a suspendido bloqueado**: si estando suspendido, el S.O. da la orden de activarlo.
- **Paso a suspendido preparado**: se puede producir por tres causas:
  - Suspender un proceso preparado.
  - Desbloqueo de un proceso suspendido bloqueado.

COLAS DE PROCESOS

- El S.O. organiza los PCB en colas de espera por el procesador o por los dispositivos de E/S. (colas de planificación: cola de procesos, colas de dispositivos)



PLANIFICACIÓN DE LA CPU

Se refiere a la gestión del procesador cuyo objetivo es proporcionar un buen servicio a los procesos que requieren su servicio.

- **PLANIFICADOR A LARGO (planificador de trabajos)**
  - Es el encargado de crear los procesos necesarios para realizar los trabajos, creándose una cola donde se colocan a medida que se solicita por el usuario el ordenador y se van sacando a medida que el sistema puede cargarlos en memoria. Decide cual es el siguiente trabajo que se va a ejecutar. Solo existe en los sistemas de trabajo por lotes, donde la decisión puede basarse en las necesidades de recursos y sus disponibilidades. En los sistemas de tiempo compartido, o no existen, o son muy elementales, limitándose su labor a cargar los programas en memoria o rechazar los mismos.
  - Su frecuencia de ejecución es de varios minutos, y es el que decide el grado de multiprogramación del sistema.
- **PLANIFICACIÓN A MEDIO PLAZO (planificador de procesos inactivos)**
  - Decide si se debe sacar temporalmente un proceso (suspenderlo) para reducir el grado de multiprogramación. Esta técnica se conoce como intercambio (swapping) y se estudia en gestión de memoria.
  - Solo existe en los sistemas de tiempo compartido y en aquellos en los que existe gestión de memoria virtual o en los que tienen procesos intercambiables . Este nivel es el que gestiona los procesos cuando se suspende su ejecución, o quedan esperando por una operación de E/S. Su frecuencia de ejecución suele ser de segundos y también tiene influencia directa en el grado de multiprogramación del sistema.

PLANIFICACIÓN DE LA CPU

- **PLANIFICACIÓN A CORTO PLAZO (planificador del corto plazo)**
  - Es el que decide acerca de los procesos que están en espera. Tiene la responsabilidad de la multiprogramación . Está siempre residente en memoria y suele ejecutarse varias veces por segundo (≈ 50). Es en este nivel donde se debe dar buen servicio a los procesos interactivos porque es donde el usuario valora el servicio.

OBJETIVOS

- **OBJETIVOS**
- Justicia
- Máxima capacidad de ejecución
- Máximo número de usuarios interactivos
- Predecible
- Minimización de sobrecarga
- Equilibrio del uso de recursos
- Seguridad de las prioridades

CRITERIOS

- **Productividad o rendimiento**: numero de trabajos que se completan por unidad de tiempo.
- **Tiempo de respuesta**: Hace referencia a la velocidad con la que el usuario da respuesta a una petición.
- **Tiempo de servicio o de retorno (ts):** es el tiempo que va desde que entra en el sistema hasta que sale del mismo. Incluye:
  - **Tiempo de carga en memoria.**
  - **Tiempo de espera como proceso preparado**
  - **Tiempo de ejecución en el procesador**
  - **Tiempo consumido en operaciones de E/S (bloqueado)**
- **Tiempo de ejecución:** idéntico al de servicio pero sin tener en cuenta el tiempo de espera en la cola de preparados. Sería el tiempo que el proceso necesita para ser ejecutado si fuese el único proceso existente.
- **Tiempo de procesador**: idéntico al de ejecución pero sin tener en cuenta el tiempo bloqueado.
- **Tiempo de espera**: tiempo que los procesos están activos sin ser ejecutados
- **Eficiencia**: se expresa como la relación entre el tiempo de procesador de cada proceso y la ocupación total del procesador. Medidas:

  Ti: momento de inicio del proceso

  Tf: momento de finalización del proceso

  T: tiempo de estado de ejecución.

  Tiempo de servicio: ts:tf-ti

  Tiempo de espera: te: ts – t

  Índice de servicio: i: t/ts

  Cuando ***i*** sea próximo a la unidad, el proceso se dice está **limitado por proceso**. Cuando ***i*** sea próximo a cero, el proceso se dice está **limitado por E/S**.

ALGORITMOS NO APROPIATIVOS

- First-Come-First-Served (**FCFS**)
  - Sencillo de entender y de implementar a través de una cola FIFO en su PCB.
  - A veces la productividad es muy baja.
  - Ej: un proceso de ráfaga de C.P.U. grande y muchos procesos de ráfaga de E/S grande pueden estar estorbándose. Efecto convoy. Un trabajo de mucha C.P.U. retiene a los de mucha E/S y estos pasan todos juntos a E/S sin apenas usar CPU.

ALGORITMOS NO APROPIATIVOS

SHORTEST JOB FIRST (SJF)

- Asigna la C.P.U. al que tenga la siguiente ráfaga de C.P.U. más corta. Si dos trabajos tienen la misma duración de ráfaga, se utiliza el FCFS
- Se pueden considerar el óptimo.
- La experiencia dice que dando paso a los trabajos limitados por E/S se mejora el tiempo de retorno de estos mas de lo que se empeora el de los limitados por C.P.U..
- No puede implementarse como planificador a C.P. debido a la imposibilidad de conocer la duración de la siguiente ráfaga. Un enfoque consiste en suponerla similar a las anteriores. Así asignamos la CPU al proceso que nos dé la predicción de ráfaga de CPU más corta mediante la fórmula:

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

t_n Contiene nuestra información más reciente.

τ_n Almacena la predicción pasada

PRIORIDAD

- El SJF es un caso particular de algoritmo por prioridades.
- La prioridad es inversa a la duración de la ráfaga.
- Las prioridades pueden asignarse internamente o externamente.
- Las prioridades internas utilizan información propia de los procesos:
  - Límites de tiempo
  - Requerimientos de memoria
  - Número de ficheros abiertos
  - Relación entre media de ráfagas de CPU y de E/S
- Las prioridades externas suelen ser criterios políticos: categoría del usuario, pagos...etc.
- Hay problema de **inanición** o **bloqueo indefinido** que se soluciona con el ***envejecimiento (aging)***.

ALGORITMOS DE APROPIACIÓN

- SRTF(***Shortest-Remaining-Time-First***)

  – El FCFS es intrínsecamente no apropiativo.
  – El SJF puede ser tanto apropiativo como no apropiativo. Si se interrumpe para analizar la situación cada vez que entra un trabajo, se conoce como SRTF

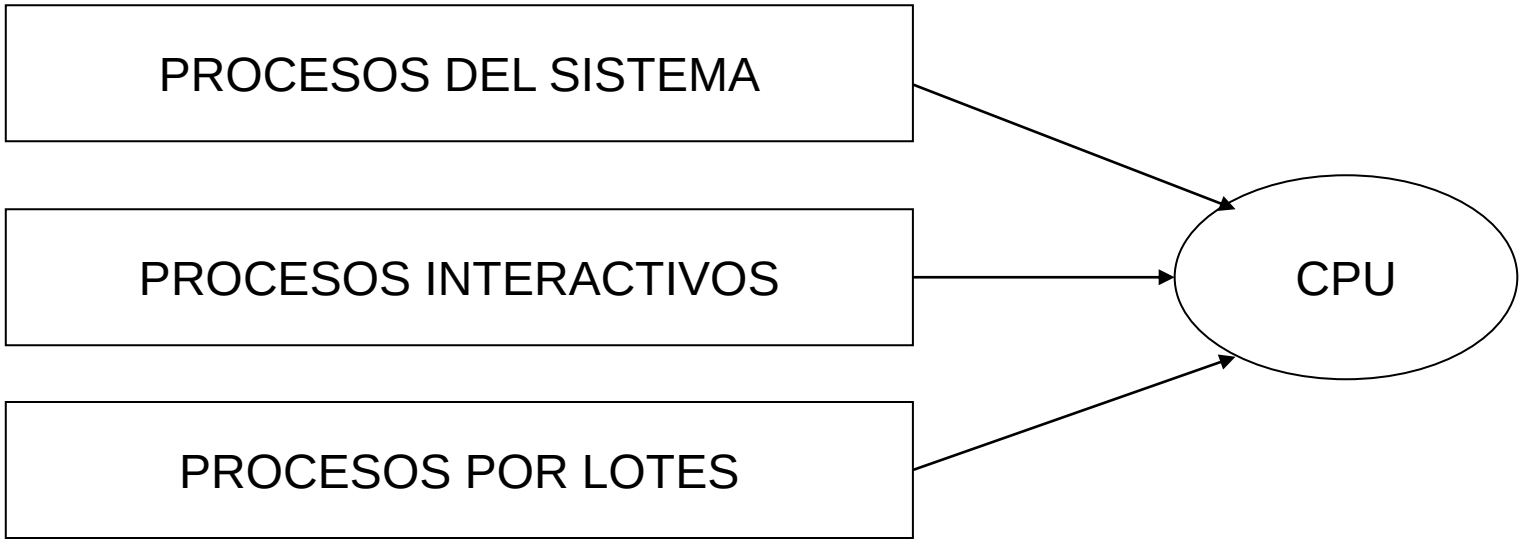
ALGORITMOS DE APROPIACIÓN

- **Asignación Round-Robin**
  - Especial para tiempo compartido.
  - ***Quantum de tiempo*** es un concepto importante.
  - Los procesos se mantienen en una cola FIFO.
  - Si el proceso sobrepasa el limite del quantum, el temporizador sobrepasa su limite y provoca un cambio de contexto. La duración de este cambio de contexto puede variar de 10 a 100 microsegundos.
  - Es importante analizar la duración del quantum pues determina la duración del tiempo medio de retorno.
  - Si la duración es demasiado grande, degenera en un FCFS.
  - Una regla empírica dice el 80% de las ráfagas de CPU deben ser menores que la duración del quantum.

COLAS MULTINIVEL

- Cuando los trabajos son fácilmente clasificables en diferentes grupos. Por ejemplo trabajos **foreground** (interactivos) y **background** (batch), la cola de preparados se divide en diversas colas
- Los trabajos se asignan a cada cola generalmente por tipo de trabajo o necesidades de memoria.
- Cada cola tiene su algoritmo de planificación.
- Es típico utilizar RR para la cola foreground y FCFS para la background.
- Además tiene que haber una planificación entre colas. Por ejemplo la foreground tiene prioridad sobre la background. Ejemplo de colas:
  - Trabajos del sistema
  - Programas interactivos
  - Edición interactiva
  - Trabajos batch
  - Trabajos de estudiantes
- Cada cola tiene prioridad absoluta sobre las de menor prioridad.
- Cuando entra un trabajo de mayor prioridad desbanca a los de menor.
- Otra posibilidad es fraccionar el tiempo entre las colas. P. Ej: 80% para cola foreground (RR)y 20% para background (FCFS).

COLAS MULTINIVEL
prioridades y partes equitativas



COLAS MULTINIVEL CON REALIMENTACIÓN

- Con dos colas típicas foreground y background no hay posibilidad de intercambio porque son procesos de diferente naturaleza pero es normal que en diferentes colas se permita el pase de una cola a otra de los procesos.
- La idea es separar los trabajos en función de las ráfagas de CPU.
- Los trabajos de poca CPU y mucha E/S así como los interactivos se les asignan las colas de mayor prioridad.
- Si un trabajo tarda mucho se le pasa a una de mayor prioridad. Ej:
    - Cola 0 quantum 8
    - Cola 1 quantum 16
    - Cola 2 FCFS
- Los trabajos se asignan primero a la cola 0, inmediatamente se pasan a la 1 y, si no finalizan, se pasan a la 2.
- En general un planificador de colas multinivel se define por:
  - Número de colas
  - Algoritmo de planificación de cada cola
  - Método para determinar cuando se pasa un trabajo a una cola de mayor prioridad
  - Método para determinar cuando se pasa un trabajo a una cola de menor prioridad
  - Método para determinar en que cola entra un trabajo cuando este precise un servicio
- Es el más complejo de todos

EVALUACIÓN DE POLÍTICAS

- ¿Cómo seleccionamos un algoritmo para un sistema determinado?
  - Lo primero es determinar el criterio a utilizar en la selección de un algoritmo.
  - Los criterios suelen ser:
    - Utilización de la CPU
    - Tiempo de respuesta
    - Productividad.
    - Podemos utilizar criterios como:
      - Maximizar la utilización de la CPU bajo la restricción de que el tiempo máximo de respuesta sea de un segundo.
      - Maximización de la productividad de modo que el tiempo de retorno sea (en promedio) linealmente proporcional al tiempo total de ejecución.
  - Una vez definido el criterio de selección tenemos que ***evaluar*** los distintos algoritmos bajo consideración

EVALUACIÓN DE POLÍTICAS (2)

- **Modelos deterministas**
  - Toma una carga de trabajo concreta predeterminada y obtiene las prestaciones de cada algoritmo para esa carga de trabajo. Hay que:
    - Definir los criterios de rendimiento
    - Establecer un conjunto de algoritmos candidatos
    - Establecer una carga de trabajo representativa del sistema
    - Para cada algoritmo:
      - Sometemos la carga de trabajo a su planificación
      - Evaluamos su rendimiento en función de los criterios definidos en el primer punto
    - Seleccionamos el que mejor se comporte
  - Características del modelo
    - Cómoda de realizar
    - Proporciona magnitudes exactas con las que comparar las estrategias
    - Una carga de trabajo puede no ser representativa

EVALUACIÓN DE POLÍTICAS (3)

- **Modelos de colas**
  - En muchos sistemas los trabajos son impredecibles (no hay un conjunto de trabajos estático) y no se puede utilizar un modelo deterministico.
  - Si se puede determinar la distribución de las ráfagas de CPU y la de E/S. El resultado es una fórmula matemática que describe la probabilidad de una ráfaga de CPU concreta. Análogamente podemos conocer la distribución de los tiempos de llegada de los trabajos al sistema (distribución del tiempo de llegada)
  - A partir de estas dos distribuciones, es posible calcular las medias de la productividad, utilización, tiempos de espera, etc., para la mayoría de los algoritmos.
  - El sistema informático se describe como una red de servidores. Cada servidor tiene una cola de trabajos en espera. La CPU es un servidor de su cola de preparados, así como el sistema de E/S lo es de su cola de dispositivo.
  - Si conocemos los ritmos de llegada y de servicio, podemos calcular la utilización, la longitud media de cola, el tiempo de espera medio, etc. Esto se conoce como ***análisis de redes de colas***.

EVALUACIÓN DE POLÍTICAS (4)

- Simulaciones
  - Se establece un modelo informático simulado
    - Principales componentes mediante tipos abstractos de datos:
      - Dispositivos
      - Planificadores
      - Estructuras de control
    - Los datos que conducen a la simulación se generan mediante números aleatorios o mediante trazas (simulación real)
      - Procesos
      - Ciclos de CPU y E/S
  - Las simulaciones permiten obtener una evaluación más cercana a la realidad
  - Presentan el problema de su alto coste

PLANIFICACIÓN EN TIEMPO REAL

- Un sistema en tiempo real es aquel en el que el tiempo desempeña un papel fundamental. Uno o más dispositivos externos generan estímulos y la computadora debe reaccionar en un tiempo limitado. Por ejemplo un reproductor de discos los bits que recibe del dispositivo debe convertirlos en música, en un intervalo de tiempo muy estricto. Si no lo hace, el sonido será raro.

- Los sistemas en tiempo real se clasifican en:
  - **Tiempo real estricto**: hay plazos absolutos para cumplir.
  - **Tiempo real no estricto**: admite incumplimientos ocasionales

- El comportamiento en tiempo real se logra dividiendo el programa en varios procesos cuyo comportamiento puede predecirse y se conoce de antemano. Por lo general son procesos cortos, cuando se detecta un suceso externo, el planificador debe programar los procesos de tal modo que se cumplan los plazos.

- Los sucesos a los que un sistema en tiempo real tiene que responder pueden ser **periódicos** o **aperiódicos** (ocurrencia impredecible).

- Para que un sistema en tiempo real sea planificable tiene que cumplir este criterio:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

PLANIFICACIÓN CON MULTIPROCESADORES

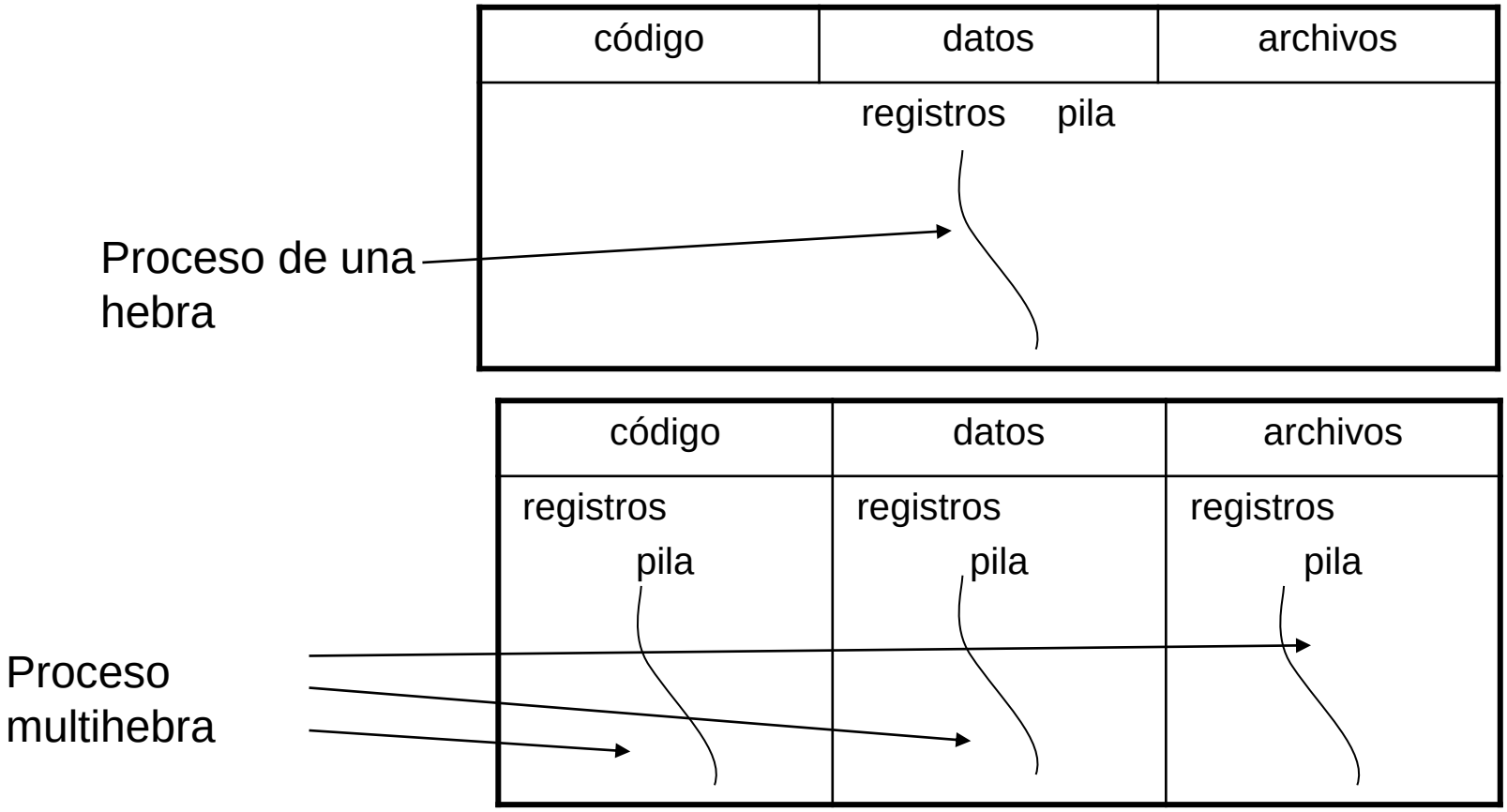
La carga de trabajo se comparte por varios procesadores, eso implica un sistema mas complejo. Puede ser que los procesadores sean homogéneos, en este caso es indiferente el uso del mismo

- Métodos de planificación
  - Multiprocesamiento simétrico. Todas las decisiones son tomadas por un procesador, los demás se limitan a ejecutar código.
  - Multiprocesamiento simétrico (SMP). Cada procesador tiene su propia planificación. Los procesos pueden estar en una cola de preparados común o haber colas privadas a cada procesador. Si la cola es común, debe tenerse cuidado de que cada proceso sea utilizado solo por un procesador.
- Afinidad al procesador
  - Los datos están en cada caché asociada a cada procesador.
  - puede ser:
    - Afinidad dura: solo lo hace en situaciones excepcionales
    - Afinidad suave : puede permitirlo por política
- Equilibrado de la carga
  - Busca mantener equilibrada la actividad de los procesadores
  - Migración comandada: cada determinado tiempo una tarea comprueba la carga de ambos procesadores y soluciona un posible desequilibrio
  - Migración solicitada: un procesador inactivo extrae un proceso de la cola activa.
  - Normalmente los S.O. admiten la coexistencia de ambos modelos

HEBRAS

- Es una unidad básica de utilización de la CPU. Un proceso se compone al menos de una hebra. Un proceso que tiene varias hebras puede realizar varias tareas a la vez. La hebra comprende:
  - Identificador de hebra
  - Contador de programa
  - Conjunto de registros
  - Pila
- Comparte con las otras hebras que pertenecen al mismo proceso:
  - Sección de código
  - Sección de datos
  - Otros posibles recursos: archivos abiertos, señales
- Un proceso que utiliza los mismos recursos puede desdoblarse. Por ejemplo un explorador Web puede disponer de una hebra para mostrar la información por la pantalla mientras que otra hebra está recibiendo información de la red.

HEBRAS


código
datos
archivos
registros
pila
Proceso de una hebra
código
datos
archivos
registros
pila
registros
pila
registros
pila
Proceso multihebra

HEBRAS

- VENTAJAS
  - **Capacidad de respuesta**: si un proceso se bloquea por alguna causa, el proceso puede continuar con otra parte del código.
  - **Compartición de recursos**: puede haber varias hebras accediendo a los mismos recursos (memoria, ficheros)
  - **Economía**: el trabajo de crear procesos es muy superior al de crear hebras. Los cambios de contexto también son mas latosos. (30 y 5 en Solaris)
  - **Utilización de arquitecturas multiprocesador**: la arquitectura multiprocesador puede ayudar en este sentido. Un proceso monohebra sólo se puede ejecutar sobre un procesador.