Servicios del Sistema Operativo

1. Servicios del Sistema Operativo

Un sistema operativo proporciona un entorno para la ejecución del programa. Proporciona algunos servicios a los programas. Los distintos servicios que proporciona un sistema operativo son los siguientes:

- **Ejecución de programas**: El sistema debe ser capaz de cargar un programa en memoria y ejecutarlo. El programa debe ser capaz de terminar esta ejecución de forma normal o anormal.
- **Operación de E/S**: Un programa en ejecución puede requerir E/S. Esta E/S puede involucrar un archivo o un dispositivo de E/S para un dispositivo específico. Alguna función especial puede ser deseada. Por lo tanto el sistema operativo debe proveer un medio para hacer E/S.
- **Manipulación del Sistema de Archivos**: Los programas necesitan crear y borrar archivos por nombre y leer y escribir archivos. Por lo tanto el sistema operativo debe mantener todos y cada uno de los archivos correctamente.
- **Comunicación**: La comunicación se implementa a través de memoria compartida o mediante la técnica de paso de mensajes en la que paquetes de información son movidos entre los procesos por el sistema operativo.
- **Detección de errores**: El sistema operativo debe tomar las medidas adecuadas en caso de que se produzcan errores de cualquier tipo, como desbordamiento aritmético, acceso a una ubicación de memoria ilegal o un tiempo de CPU de usuario demasiado largo.
- **Asignación de recursos**: Cuando hay varios usuarios conectados al sistema, los recursos deben asignarse a cada uno de ellos. Para la distribución actual del recurso entre los distintos procesos, el sistema operativo utiliza los tiempos de ejecución de programación de la CPU, que determinan a qué proceso se asignará el recurso.
- **Contabilidad**: El sistema operativo lleva la cuenta de qué usuarios utilizan cuántos y qué tipo de recursos del ordenador.
- **Protección**: El sistema operativo es responsable de la protección tanto del hardware como del software. El sistema operativo protege la información almacenada en un sistema informático multiusuario.

1.1. Gestión de procesos

Proceso: Un proceso o tarea es una *instancia* de un programa en ejecución. La ejecución de un proceso debe programarse de forma secuencial. En cualquier momento se ejecuta como máximo una instrucción. **El proceso incluye la actividad actual representada por el valor del contador de programa y el contenido de los registros del procesador**. También incluye **la pila de proceso que contiene datos temporales** (como parámetros de método, dirección de retorno y variables locales) **y una sección de datos que contiene variables globales**.

Diferencia entre proceso y programa

> **Un programa por sí mismo no es un proceso**. Un programa en ejecución se conoce como proceso. **Un programa es una entidad pasiva**, como el contenido de un archivo almacenado en disco, **mientras que un proceso es una entidad activa** con un contador de programa que especifica la siguiente instrucción a ejecutar y un conjunto de recursos asociados que pueden ser compartidos entre varios procesos con algún algoritmo de programación que se utiliza para determinar cuándo dejar de trabajar en un proceso y dar servicio a otro diferente.

Estado del proceso: A medida que un proceso se ejecuta, cambia de estado. El estado de un proceso viene definido por la actividad correcta de dicho proceso. Cada proceso puede estar en uno de los siguientes estados.

- **Nuevo / New**: El proceso se está creando.

- **Listo / Ready**: El proceso está esperando a ser asignado a un procesador.
- **En ejecución / Running**: Se están ejecutando instrucciones.
- **En espera / Waiting**: El proceso está esperando que ocurra algún evento.
- **Terminado / Terminated**: El proceso ha finalizado su ejecución.

Muchos procesos pueden estar en estado ready y waiting al mismo tiempo. Pero **sólo un proceso** puede estar ejecutándose en cualquier procesador en cualquier instante.

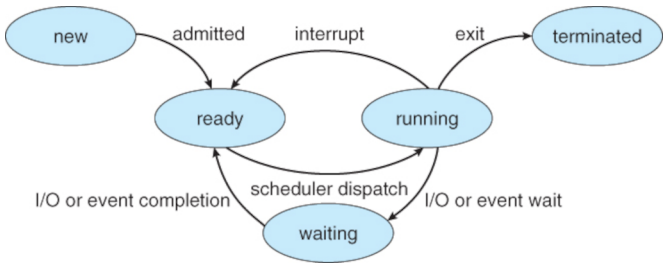


Fig. 01: Estado de los procesos

1.2. Programación de procesos / Process scheduling

La programación es una función fundamental del sistema operativo. Cuando un ordenador está multiprogramado, tiene múltiples procesos completándose para la CPU al mismo tiempo. Si sólo hay una CPU disponible, hay que decidir qué proceso se ejecutará a continuación. Este proceso de toma de decisiones se conoce como **programación / scheduling** y la parte del sistema operativo que hace esta elección se llama **programador / scheduler**. El algoritmo que utiliza para hacer esta elección se llama **algoritmo de programación / scheduling algorithm**.

Colas de programación / Scheduling queues A medida que los procesos entran en el sistema, se colocan en una cola de trabajos (*job queue*). Esta cola está formada por todos los procesos del sistema. Los procesos que residen en la memoria principal y están listos y esperando para ejecutarse se mantienen en una lista llamada cola lista (*ready queue*).

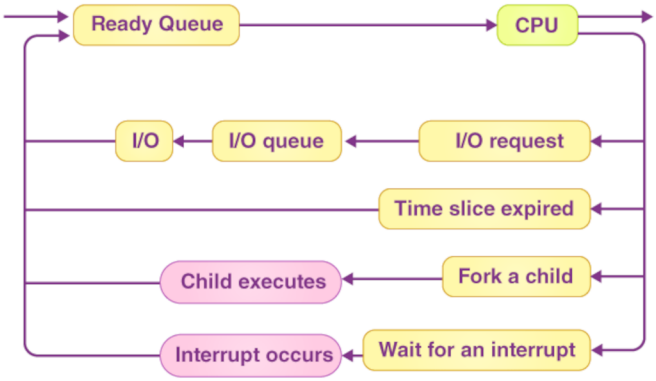


Fig. 02: Colas de programación

Esta cola se almacena generalmente como una lista enlazada. La cabecera de una cola de espera contiene punteros al primer y último PCB (*Process Control Block*) de la lista. El PCB incluye un campo puntero que apunta al siguiente PCB en la cola de listos. Las listas de procesos en espera de un determinado dispositivo de E/S se mantienen en una lista denominada cola de dispositivos. Cada dispositivo tiene su propia cola de dispositivos. Un nuevo proceso se coloca inicialmente en la cola de listos. Espera en la cola de listos hasta que es seleccionado para la ejecución y se le da la CPU.

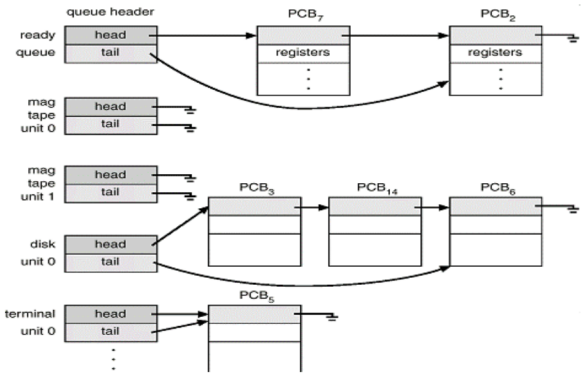


Fig. 03: Process Control Block

1.3. Programadores / Schedulers

Un proceso migra entre las distintas colas de programación a lo largo de su vida útil. El sistema operativo debe seleccionar de alguna manera los procesos de estas colas para programarlos. Este proceso de selección lo lleva a cabo el planificador apropiado. En un sistema por lotes, se envían más procesos que se ejecutan inmediatamente. Por lo tanto, estos procesos se almacenan en un dispositivo de almacenamiento masivo como el disco, donde se guardan para su posterior ejecución.

Tipos de programadores:

Existen 3 tipos de planificadores utilizados principalmente:

1. **Planificador a largo plazo /** *Long term scheduler*: El planificador a largo plazo selecciona procesos del disco y los carga en memoria para su ejecución. Controla el grado de multiprogramación, es decir, el número de procesos en memoria. Se ejecuta con menos frecuencia que otros planificadores. Si el grado de multiprogramación es estable, la tasa media de creación de procesos es igual a la tasa media de salida de procesos del sistema. Por tanto, el planificador a largo plazo sólo debe invocarse cuando un proceso abandona el sistema. Debido a los intervalos más largos entre ejecuciones puede permitirse tomarse más tiempo para decidir qué proceso debe seleccionarse para su ejecución.

   La mayoría de los procesos en la CPU están ligados a la E/S o a la CPU. Un proceso ligado a la E/S (un programa 'C' interactivo es aquel que gasta la mayor parte de su tiempo en operaciones de E/S que en hacer operaciones de E/S. Un proceso ligado a la CPU es aquel que gasta más de su tiempo en hacer cálculos que en operaciones de E/S (programa de ordenación complejo). Es importante que el planificador a largo plazo seleccione una buena combinación de procesos ligados a la E/S y a la CPU.

2. **Planificador a corto plazo /** *Short-term scheduler*: El planificador a corto plazo selecciona entre los procesos que están listos para ejecutarse y asigna la CPU a uno de ellos. La principal diferencia entre estos dos planificadores es la frecuencia de su ejecución. El planificador a corto plazo debe seleccionar un nuevo proceso para la CPU con bastante frecuencia. Debe ejecutar al menos uno cada 100 ms. Debido a la corta duración del tiempo entre ejecuciones, debe ser muy rápido.

3. **Planificador a medio plazo /** *Medium-term scheduler*: algunos sistemas operativos introducen un nivel intermedio adicional de planificación conocido como planificador a medio plazo. La idea principal detrás de este planificador es que a veces es ventajoso eliminar procesos de la memoria y reducir así el grado de multiprogramación. En algún momento posterior, el proceso puede ser reintroducido en la memoria y su ejecución puede continuar desde donde se había quedado. Esto se denomina swapping. El proceso se intercambia y se vuelve a intercambiar más tarde por el planificador a medio plazo. El intercambio es necesario para mejorar la pérdida de procesos o debido a algún cambio en los requisitos de memoria, se excede el límite de memoria disponible, lo que requiere liberar algo de memoria.

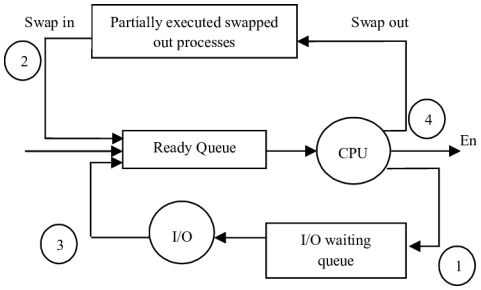


Fig. 04: **Proceso de Programadores** Muestra los siguientes estados que se han ejecutado en el Programador de CPU. 1. Cuando un proceso pasa del estado de ejecución al estado de espera. 2. Cuando un proceso pasa del estado en ejecución al estado listo. 3. Cuando un proceso pasa del estado de espera al estado de listo. 4. Cuando un proceso termina.

El éxito de un programador de CPU depende del diseño de un algoritmo de programación de alta calidad. Los algoritmos de programación de CPU de alta calidad se basan principalmente en criterios como la tasa de utilización de la CPU, el rendimiento, el tiempo de respuesta, el tiempo de espera y el tiempo de ejecución.

1.4. Bloque de control de procesos

Cada proceso está representado en el SO por un bloque de control de proceso (PCB). También se conoce como bloque de control de tareas.

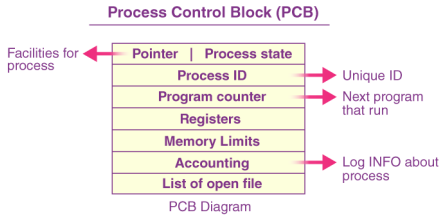


Fig. 03: Process Control Block o Task Control Block

Un bloque de control de procesos contiene muchas informaciones asociadas a un proceso específico. Incluye la siguiente información

- **Estado del proceso / Process state**: El estado puede ser nuevo, preparado, en ejecución, en espera o finalizado.
- **Contador de programa** / *Program counter*: indica la dirección de la siguiente instrucción que se va a ejecutar del proceso.
- **Registros de la CPU** / *CPU registers*: Los registros varían en número y tipo dependiendo de la arquitectura del ordenador. Incluye acumuladores, registros de índice, puntero de pila y registros de propósito general, además de cualquier información de código de condición que deba guardarse cuando se produce una interrupción para permitir que el proceso continúe correctamente después.
- **Información de programación de la CPU** / *CPU scheduling information*: Esta información incluye los punteros de prioridad del proceso a las colas de programación y cualquier otro parámetro de programación.
- **Información sobre la gestión de la memoria** / *Memory management information*: Esta información puede incluir datos como el valor de los registros de barras y límites, las tablas de páginas o las tablas de segmentos, dependiendo del sistema de memoria utilizado por el sistema operativo.
- **Información contable** / *Accounting information*: Esta información incluye la cantidad de CPU y tiempo real utilizado, límites de tiempo, número de cuenta, números de trabajo o proceso, etc.
- **Información de estado de E/S** / *I/O Status Information*: Esta información incluye la lista de dispositivos de E/S asignados a este proceso, una lista de archivos abiertos, etc. La PCB sirve simplemente como repositorio de cualquier información que pueda variar de un proceso a otro.

2. Algoritmos de programación de la CPU

La programación de la CPU se ocupa del problema de decidir cuál de los procesos en la cola de espera debe ser asignado primero a la CPU. Existen cuatro tipos de programación de CPU.

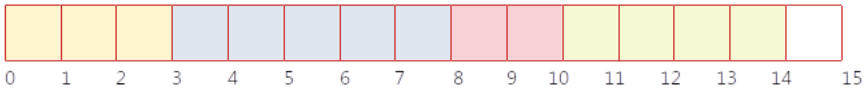
2.1. Algoritmo Por Orden de Llegada o FCFS (*First Come, First Served*)

Es el algoritmo de programación de CPU más sencillo. En este esquema, el proceso que solicita la CPU en primer lugar, que se asigna a la CPU en primer lugar. La implementación del algoritmo FCFS se gestiona fácilmente con una cola FIFO. Cuando un proceso entra en la cola de listos, su PCB se enlaza a la parte posterior de la cola. El tiempo medio de espera bajo la política FCFS es *bastante largo*. Considere el siguiente ejemplo:

Proceso	Tiempo CPU
P1	3
P2	5
P3	2
P4	4

Utilizando el algoritmo FCFS encuentre el tiempo medio de espera y el tiempo medio de entrega si el pedido es P1, P2, P3, P4.

Solución: Si el proceso llegó en el orden P1, P2, P3, P4 entonces según el FCFS el diagrama de Gantt será:



El tiempo de espera para el proceso

P1 = 0, P2 = 3, P3 = 8, P4 = 10

entonces el tiempo de respuesta para el proceso

P1 = 0 + 3 = 3, P2 = 3 + 5 = 8, P3 = 8 + 2 = 10, P4 = 10 + 4 =14.

Entonces,

Tiempo medio de espera = (0 + 3 + 8 + 10)/4 = 21/4 = **5,25**

Tiempo medio de respuesta = (3 + 8 + 10 + 14)/4 = 35/4 = **8,75**

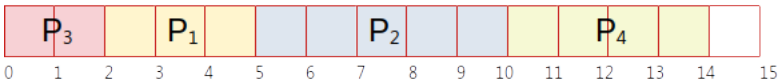
El algoritmo FCFS es no preventivo, lo que significa que una vez que la CPU ha sido asignada a un proceso, el proceso mantiene la CPU hasta que la libera, ya sea terminando o solicitando E/S.

2.2. Algoritmo Programación del trabajo más corto primero / *Shortest Job First Scheduling* (SJF)

: Este algoritmo asocia a cada proceso si la CPU está disponible. Esta programación también se conoce como "la siguiente ráfaga de CPU más corta" (*shortest next CPU burst*), porque la programación se realiza examinando la longitud de la siguiente ráfaga de CPU del proceso en lugar de su longitud total. Considere el siguiente ejemplo:

Proceso	Tiempo CPU
P1	3
P2	5
P3	2
P4	4

Solución: el SJF el diagrama de Gantt será



El tiempo de espera para el proceso

P1 = 0, P2 = 2, P3 = 5, P4 = 9

entonces

el Tiempo de respuesta para el proceso P3 = 0 + 2 = 2, P1 = 2 + 3 = 5, P4 = 5 + 4 = 9, P2 = 9 + 5 = 14.

Tiempo medio de espera = (0 + 2 + 5 + 9)/4 = 16/4 = **4**

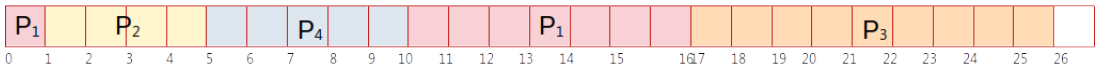
Tiempo medio de respuesta = (2 + 5 + 9 + 14)/4 = 30/4 = **7,5**

El algoritmo SJF puede ser preferente o no preferente. El algoritmo SJF preferente también se conoce como **Algoritmo del menor tiempo restante**.

Considere el siguiente ejemplo.

Proceso	Tiempo de llegada	Tiempo CPU
P1	0	8
P2	1	4
P3	2	9
P4	3	5

En este caso, el diagrama de Gantt será



El tiempo de espera del proceso

P1 = 10 1 = 9

P2 = 1 – 1 = 0

P3 = 17 – 2 = 15

P4 = 5 – 3 = 2

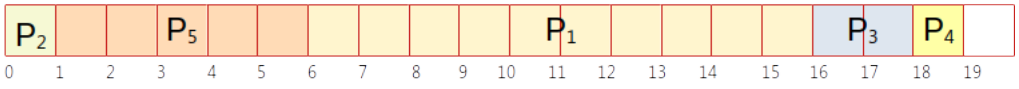
El **tiempo medio de espera** = (9 + 0 + 15 + 2)/4 = 26/4 = **6,5**

2.3. Algoritmo de programación por prioridades / *Priority Scheduling Algorithm*

En esta programación se asocia una prioridad a cada proceso y la CPU se asigna al proceso con mayor prioridad. Los procesos de igual prioridad se programan de forma FCFS. Considere el siguiente ejemplo:

Proceso	Tiempo de llegada	Tiempo CPU
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Según la programación de prioridades, el diagrama de Gantt será



El tiempo de espera del proceso

P1 = 6

P2 = 0

P3 = 16

P4 = 18

P4 = 1

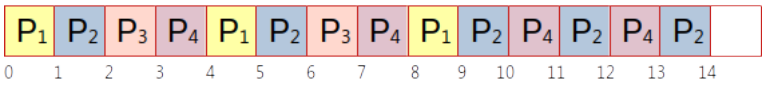
El **Tiempo medio de espera** = (0 + 1 + 6 + 16 + 18)/5 = 41/5 = **8,2**

2.4. Algoritmo de programación Round Robin / *Round Robin Scheduling*

: Este tipo de algoritmo está diseñado sólo para el sistema de tiempo compartido. Es similar a la programación FCFS con condición de preferencia para cambiar entre procesos. Una pequeña unidad de tiempo llamada *quantum time* o *time slice* se utiliza para cambiar entre los procesos. El tiempo medio de espera bajo la política *round robin* es bastante largo. Considere el siguiente ejemplo:

Proceso	Tiempo CPU
P1	3
P2	5
P3	2
P4	4

Time Slice = 1 millisecond.



El tiempo de espera del proceso

P1 = 0 + (4 – 1) + (8 – 5) = 0 + 3 + 3 = 6

P2 = 1 + (5 – 2) + (9 – 6) + (11 – 10) + (12 – 11) + (13 – 12) = 1 + 3 + 3 + 1 + 1 + 1 = 10

P3 = 2 + (6 – 3) = 2 + 3 = 5

P4 = 3 + (7 – 4) + (10 – 8) + (12 – 11) = 3 + 3 + 2 + 1 = 9

El **Tiempo medio de espera** = (6 + 10 + 5 + 9)/4 = **7,5**