**1.- Introducción.**

En la actualidad, cualquier ordenador realiza varias cosas al mismo tiempo, ya que la mayoría de los sistemas operativos son *multitarea*. Los únicos sistemas multitarea **real** son aquellos que cuentan con más de un procesador. Un proceso es el programa que se está ejecutando en ese momento en el ordenador. **Hemos de considerar que varios de estos programas podrán estar ejecutándose a la vez**, y para ello, el sistema operativo tendrá que llevar a cabo una adecuada gestión de los recursos físicos del propio sistema informático.

Un programa es un archivo ejecutable que típicamente se encuentra almacenado en memoria secundaria. Considerado como archivo, **un programa es una entidad pasiva o estática**. Cuando un programa es ejecutado se convierte en una **entidad activa o dinámica denominada proceso** que va pasando por diversos estados y utilizando distintos recursos del computador. En conclusión, **un proceso es un programa en ejecución**.

Un proceso es la instancia de un programa en ejecución que necesita para realizar su tarea recursos como:

* Tiempo de procesador
* Memoria
* Archivos
* Dispositivos de entrada / salida

Estos recursos se pueden asignar en el momento de la creación del proceso o bien durante su ejecución

Un proceso es la unidad básica de trabajo de un sistema informático.

La administración de estos procesos y de sus recursos asociados constituye una de las tareas básicas y primordiales del núcleo del sistema operativo que, entre otros, comprenderá funciones tales como:

1. *La creación y destrucción de procesos.*
2. *El despacho (****dispatcher****), es decir, la asignación de procesos a ejecución en el procesador.*
3. *Los cambios de estado que llamamos transiciones.*
4. *La suspensión y reanudación de procesos.*
5. *La sincronización de procesos.*
6. *La comunicación entre procesos.*
7. *La manipulación de bloques de control de proceso (PCB).*

Se dice que un proceso se ejecuta en primer plano, **foreground**, si los usuarios pueden interaccionar con el proceso durante su ejecución. Son ejemplos típicos de aplicaciones los programas de ofimática, los navegadores, los reproductores multimedia, etc.

Por el contrario, se dice que un proceso se ejecuta en segundo plano, **background**, si un usuario no puede interaccionar con el proceso durante su ejecución. Los procesos background proporcionan prestaciones como servicios de páginas web, registro de eventos, servicios de impresión, criptografía ...

De forma general se pueden distinguir **tres tipos de procesos**:

**Procesos de usuario**. - Procesos asociados a la ejecución de programas invocados por los usuarios. Se ejecutan en modo usuario excepto cuando realizan llamadas al sistema que pasan a ser ejecutados en modo supervisor. **Se suelen ejecutar en primer plano o en segundo plano**.

**Servicios o demonios**. - Procesos no asociados a ningún usuario que realizan tareas periódicas relacionadas con la administración del sistema (por ejemplo, administración y control de redes, administración de trabajos de impresión, etc.). Se ejecutan en modo usuario excepto cuando realizan llamadas al sistema que se ejecutan en modo supervisor. **Se suelen ejecutar en segundo plano**.

Aclarar que el nombre demonio se utiliza en sistemas operativos basados en UNIX y servicio es el nombre que reciben estos procesos en entornos Windows.

**Procesos del sistema operativo o simplemente procesos del sistema**. - Son procesos que realizan tareas de administración del sistema operativo, como, por ejemplo, el intercambio de procesos desde la memoria principal a memoria secundaria. **Se ejecutan normalmente en modo supervisor y en segundo plano.**

**Otras formas de clasificar a los procesos.**

Según el **propietario** (clasificación típica de Linux) del proceso tenemos:

* **Procesos de sistema**: Asociados al funcionamiento del núcleo del sistema o de los diferentes servicios en funcionamiento (demonios). Muchos de estos procesos están asociados a usuarios virtuales del sistema (lp, www, mail, etc.).
* **Procesos de superusuario**: Asociados a la cuenta del administrador raíz (root).
* **Procesos de usuario**: Asociados a la ejecución de aplicaciones de un usuario determinado.

También podemos clasificar **los procesos que se ejecutan concurrentemente** como:

* **Independientes**: no afectan ni son afectados por otros procesos.
* **Cooperantes**: pueden compartir datos y por lo tanto afectar y ser afectados por otros procesos.

Según la **forma de ejecución y ubicación en la memoria** pueden ser:

* **Residentes**: los procesos residentes están siempre en la memoria mientras dura la ejecución.
* **Intercambiables**: pueden ser llevados de la memoria principal al disco duro mientras están bloqueados. Así, la memoria liberada puede ser utilizada por otros procesos que la necesiten.

Finalmente, **según los recursos que comparten** encontramos procesos:

* **Pesados**: los procesos pesados son independientes y tienen todos sus propios recursos.
* **Ligeros**: También llamados hilos de ejecución o *threads*, comparten entre sí espacio de memoria y recursos de entrada y salida.

**2.- Estados de los procesos**

El proceso es un elemento *dinámico* que puede pasar por diferentes estados a lo largo de su “existencia”. Podemos distinguir los siguientes estados:

**Ejecutándose o Activo**. Es el proceso que está ejecutándose en un instante dado.

***Preparado****.* - En él se encuentran todas las tareas que están listas para ejecutarse pero que esperan a que el procesador quede libre.

**Bloqueado o suspendido**. - En este estado se encuentran las tareas que no están preparadas para ejecutarse por algún motivo, por ejemplo, porque están esperando a que termine una operación de e/s.

***Nuevo***. - Programa que existe pero que aún no se le han podido asignar los recursos necesarios y por tanto aún no está preparado. Los sistemas operativos de actuales **no contemplan este estado** ya que todos los procesos que se crean se pasan al estado "preparado”.

**Terminado o finalizado**. - En este estado, un proceso ha terminado su ejecución en el sistema. Puede ser que haya terminado normalmente o por causa de algún fallo.

**Estado de un proceso “zombi”**

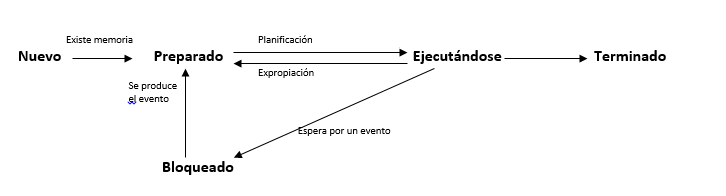
Podemos considerarlo como una variante del estado finalizado. Es un proceso que ha finalizado (ya no es considerado por el procesador), **pero que debe mantener en memoria cierta información de utilidad para su proceso padre**. En Linux, los procesos finalizados pasan directamente a estado zombi hasta que el proceso padre se entera y los borra definitivamente. Normalmente, la presencia de procesos zombi durante mucho tiempo puede indicar algún problema en el sistema.

Sólo un proceso puede estar en estado de ejecución en un procesador y en un momento determinado, pero muchos procesos pueden estar preparados o en espera bloqueados.

Se denomina **estado global del sistema**, en un instante dado, al conjunto de recursos y procesos existentes con sus estados correspondientes.

**2.1.- Transición entre los estados.**

La figura siguiente muestra, de forma general, las transiciones entre los distintos estados (5):



El sistema operativo (s.o.) posee un módulo, el **planificador (scheduler)** que se encarga de activar los procesos que están en el estado de preparado, de acuerdo con unos criterios determinados (p.e. en función de la prioridad). Una determinada interrupción hace que la tarea que esté activa en ese momento deje de ejecutarse a favor del sistema operativo que decidirá de entre los procesos que ya estén preparados, cuál de ellos tiene que ponerse en estado activo. **Aquí entra en juego el planificador**.

El estado activo también puede volver al estado preparado por la acción del planificador del s.o. Este hecho recibe el nombre de *expropiación*.

Un proceso pasa de nuevo a preparado cuando el sistema, a través del planificador, lo decide. Un proceso en estado bloqueado puede pasar a preparado si “sucede algo” (evento) que esperaba dicho proceso para poder seguir la ejecución. Un proceso en ejecución puede pasar a bloqueado si solicita un recurso que no está disponible. Un proceso pasa al estado terminado cuando finaliza.

.

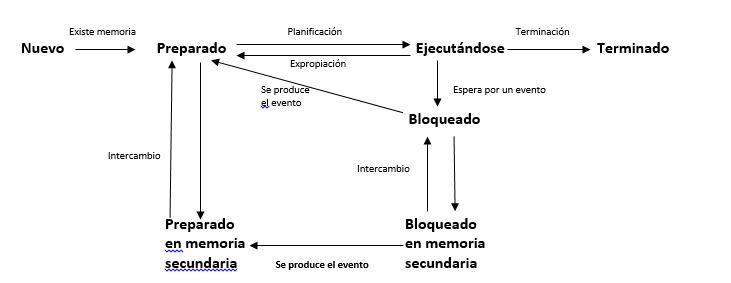
Aparte de los cinco estados descritos, otros dos estados importantes definidos en algunos sistemas operativos son **Preparado en memoria secundaria** y **Bloqueado en memoria secundaria**. Se trata de procesos en el estado Preparado o en el estado Bloqueado **que han sido intercambiados total o parcialmente desde la memoria principal a la memoria secundaria**, para reducir el grado de multiprogramación o hacer sitio para nuevos procesos. Posteriormente un proceso en memoria secundaria puede ser intercambiado de vuelta a la memoria principal. El área de memoria secundaria donde se intercambian los procesos se le denomina área o espacio de intercambio (fichero o ficheros de paginación en sistemas Windows y ficheros o particiones swap en sistemas Linux).

Este intercambio de procesos entre memoria principal y secundaria y viceversa se llama **swapping.**

**Observación:** es importante saber dónde se encuentra el proceso en cada estado: **memoria RAM, procesador o memoria secundaria**.

El esquema con los nuevos estados sería el siguiente**:**

**Esquema ampliado de la transición entre los distintos estados (7) de los procesos.**



**3.- Algoritmos de planificación de procesos.**

Como hemos comentado, una de las obligaciones de un sistema operativo como gestor de procesos es la planificación de procesos: **la ejecución de múltiples procesos optimizando el uso del procesador.**

Así pues, las políticas de planificación de procesos surgen como necesidad de cumplir dos de los principales objetivos funcionales de los sistemas operativos: la multiprogramación y el tiempo compartido:

* **Multiprogramación**: tiene como objetivo *maximizar el aprovechamiento de la CPU teniendo siempre en ejecución algún proceso en todo momento*.
* **Tiempo compartido**: dado que una CPU sólo puede ejecutar un proceso al mismo tiempo se trata de *establecer un sistema de conmutación de la CPU entre los procesos con tal frecuencia que el usuario pueda interactuar con ella y tenga la impresión de que se están ejecutando en paralelo.*

**Observación**: los algoritmos pueden ser expropiativos o apropiativos (una vez que se le ha otorgado la “cpu” a un proceso, le puede ser retirada) o no expropiativos o no apropiativos (una vez que se le ha otorgado la “cpu” a un proceso, no le puede ser retirada).

Los algoritmos más utilizados son:

**1) Planificación por prioridades**. - Cada proceso tiene asignada una y el de mayor prioridad en el estado de preparado es el que toma el procesador. La asignación de ésta puede ser de dos tipos:

a) estática, es decir, que no cambia durante el tiempo en el que el proceso existe y b) dinámica, la cual puede ser modificada.

Se puede plantear **el problema de que los procesos con menor prioridad queden relegados sin posibilidades de utilizar el procesador**. Para evitar esto, la solución que se suele adoptar es la de ir aumentando la prioridad de aquellos que lleven un tiempo de espera muy elevado. Esta estrategia se conoce como prioridad por envejecimiento o **aging**. Pueden ser *expropiativos* y *no expropiativos*.

**2) Planificación FCFS**. - Los procesos acceden al procesador en el orden de llegada al estado de preparado. Es un método sencillo que se utiliza dentro de otros esquemas, por ejemplo, en los algoritmos por prioridades, dentro de los que tienen la misma prioridad, se puede aplicar esta planificación. Es *no expropiativo*.

1. **Planificación *SJF****.* - A cada proceso se le asocia una estima del tiempo que le resta para finalizar su ejecución y la selección se realiza en base a dicho tiempo: **se selecciona el trabajo con menor valor de tiempo restante de ejecución**. Es no expropiativo.

X=15, Y=10, Z=20

Z,Y,X

TE= 20+20+10+20+10+15=95

Y,X,Z

TE= 10+10+15+10+15+20=80

**4) Planificación SRT**. - Es la versión *expropiativa* del método SJF.

**5) Planificación circular o Round Robin**.- A todos los procesos en el estado de preparado se les asigna un tiempo de ejecución denominado *cuanto (q)*. El planificador va asignando el procesador a cada tarea de forma secuencial por el cuanto definido. Si un proceso necesita un tiempo de ejecución mayor que su cuanto definido, vuelve al estado preparado al final de la lista. Pueden ser *expropiativos* y *no expropiativos*.

**6) Colas multinivel**

Consisten en diferentes colas de procesos en estado de preparado. Cada cola tiene su prioridad. Para acceder a la ejecución en el procesador se escoge el proceso de la cola más prioritaria que no esté vacía. Dentro de cada cola se puede aplicar una política y un algoritmo de planificación diferente.

**7) Colas multinivel y realimentación**

Es el mismo caso de las colas multinivel, pero los procesos pueden avanzar y retroceder por las diferentes colas de prioridades. Por ejemplo, un proceso de prioridad alta que agote su quantum podría pasar a una cola de preparados de menos prioridad.

El sistema operativo Unix/Linux utiliza colas multinivel con realimentación utilizando el algoritmo **Round Robin** en cada cola. Cada segundo se recalculan las prioridades de los procesos en función de unas prioridades definidas por un lado por el sistema, según el tipo de proceso, y por otro por el mismo usuario mediante la configuración de la prioridad **nice/renice**.

**Observación**: la conmutación de tareas se produce en todos los algoritmos. Para realizar este cambio, se debe ejecutar el sistema operativo. Se denomina **sobrecarga** del sistema al tiempo que el procesador es utilizado por el sistema operativo para realizar tareas de gestión y que no son atribuibles a llamadas del sistema realizadas por otros procesos bien de usuario y/o servicios o demonios.

**3.1.- Criterios para la planificación**

Vemos los parámetros más importantes a tener en cuenta:

**Tiempo de retorno o de regreso (TR)**. - Es el intervalo de tiempo que transcurre desde que un proceso se crea hasta que se completa por el sistema.

**Tiempo de espera (TE)**. - Es el tiempo que el proceso espera hasta que se le concede el procesador.

**Nota**: El tiempo de retorno será la suma del tiempo de espera más el *tiempo de ejecución* ***(TEj).***

**TR = TE + TEj**

**TE = TR - TEj**

**Ejemplo**:

Hallar el **tiempo de retorno** y el **tiempo de espera** para cada uno de los procesos representados en la tabla de la **figura 1** si actúa:

a) PCP **FCFS**.

b) PCP **SJF.**

**Figura 1**

|  |  |  |
| --- | --- | --- |
| Proceso | Tiempo de llegada (ms) | Tiempo de ejecución (ms) |
| **A** | **0** | **3** |
| **B** | **1** | **4** |
| **C** | **2** | **3** |
| **D** | **3** | **5** |
| **E** | **4** | **2** |

**a) FCFS.**

**Diagrama de Gantt.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |

0 3 7 10 15 17

**Tiempos de retorno y de espera**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **Tiempos medios** |
| TR | **3** | **(7-1)6** | **(10-2)8** | **(15-3)12** | **(17-4)13** | 8,4 ms |
| TE | (3-3)0 | 2 | 5 | 7 | 11 | 5 ms |

**TE = TR - TEj**

**b) SJF.**

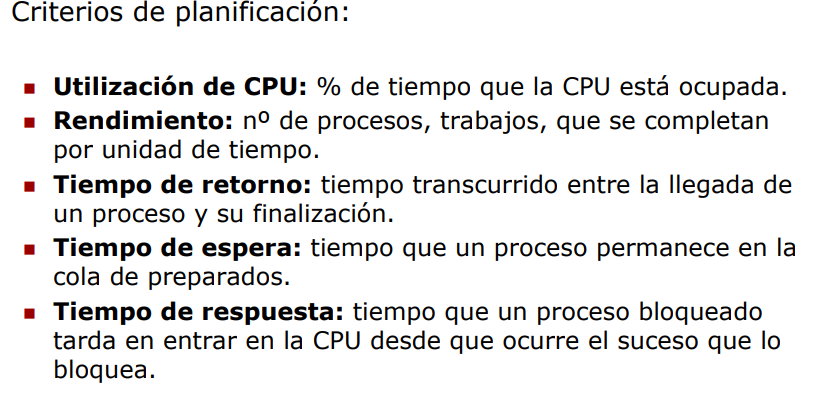
**Diagrama de Gantt**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | C | E | B | D |

0 3 6 8 12 17

**Tiempos de retorno y de espera**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **Tiempos medios** |
| TR |  |  |  |  |  |  |
| TE |  |  |  |  |  |  |



**Ejemplos en *ASO-UD07\_ACTIVIDADES\_PLANIFICADORES\_PROCESOS\_ALUM.***