ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

"¿CÓMO TRABAJA TU SISTEMA OPERATIVO?"

AUTORÍA
MARÍA CATALÁ CARBONERO
TEMÁTICA
SISTEMAS OPERATIVOS
ETAPA
CICLO MEDIO Y SUPERIOR DE INFORMÁTICA

Resumen

Los sistemas operativos nacen de la informática. Su función es ser el intermediario entre el hardware y el usuario. En los tiempos actuales, en los que el desarrollo de las tecnologías es vertiginoso, es interesante pararse un poco a comprobar cómo trabaja el sistema operativo a nivel de procesos, ya sea Guadalinex, Windows, Applet, etc.

Palabras clave

Sistemas operativos, procesos, programa, hardware, software, dispositivos.

1. INTRODUCCIÓN

El manejo del hardware suele ser demasiado complejo, por lo que el sistema operativo es el componente software que realiza este tipo de funciones. Por lo tanto un sistema operativo se va a encargar de tareas como:

- Decidir que proceso se ejecuta y en qué momento.
- Como se reparte la memoria del sistema entre datos y programas.
- Presentar a los usuarios un sistema de ficheros.
- Decidir como utilizarán los programas los periféricos a su disposición.

Por lo tanto, se puede decir que un proceso es un programa en ejecución, y que en el sistema operativo, cada proceso se representa por medio de su propio "Bloque de control de proceso"(PCB), que es un registro de datos que contiene información relacionada con un proceso concreto.

ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

2. ESTADOS DE UN PROCESO

Al ejecutarse un proceso, éste cambia de estado. Existen 2 modelos básicos de estados de un proceso:

- Modelo en 5 estados
- Modelo en 7 estados.

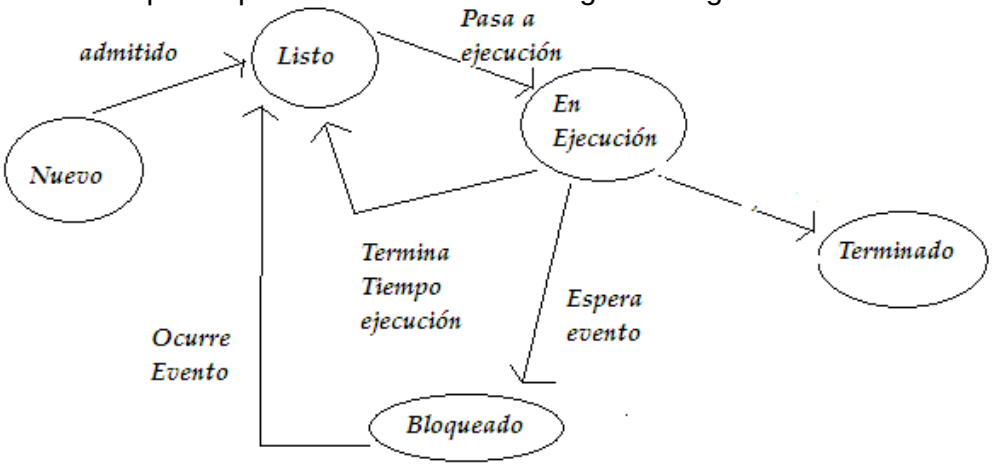
2.1. Modelo de 5 estados

Los estados en los que se puede encontrar un proceso son:

- **En Ejecución**: el proceso está actualmente ejecutándose en el procesador.
- **Listo**: el proceso está preparado para ejecutarse en cualquier momento.
- **Bloqueado**: el proceso actualmente no puede ejecutarse hasta que no se produzca un evento externo que le permita continuar.
- **Nuevo**: un proceso está en este estado cuando acaba de crearse, pero el SO aún no lo ha admitido entre los procesos que pueden ejecutarse.
- **Terminado**: el proceso acaba su ejecución (no se le asigna más tiempo de CPU).

Actividad 1: Realizar una simulación de las transiciones por las que pasa un proceso en las situaciones siguientes:
a) El proceso es creado en un instante el cuál la carga del sistema es muy alta. Posteriormente la carga del sistema vuelve a ser normal.
b) El proceso es planificado para ejecutar por primera vez.
c) El proceso ejecuta instrucciones que no necesitan llamadas al sistema y es planificado en varias ocasiones agotando siempre el quanto de tiempo que le asigna el planificador.
d) El proceso realiza una escritura en disco y tarda mucho tiempo en completarse. Antes de la finalización de esta lectura la carga del sistema es muy alta transitoriamente.

Sus transiciones son las que se pueden observar en la siguiente figura:

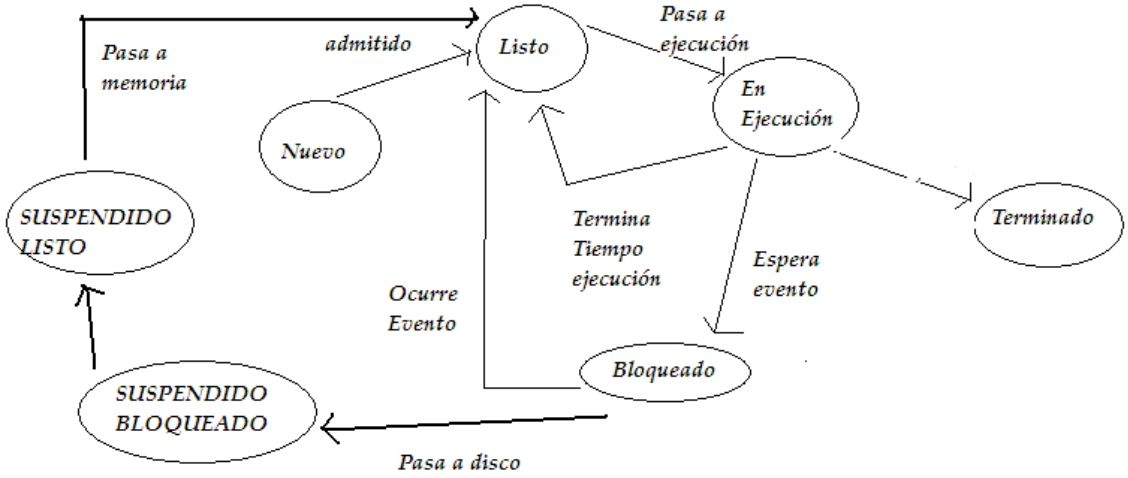


ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

2.2. Modelo de 7 estados

Se añade al modelo anterior un nuevo estado SUSPENDIDO, que indica que un proceso NO se encuentra en memoria principal sino que está almacenado en disco (en memoria secundaria). Dicho estado se subdivide en 2:

- **Suspendido-bloqueado**: es un proceso suspendido que está esperando a que se produzca 1 evento externo a él.
- **Suspendido-listo:** es un proceso suspendido que puede ser llevado a memoria principal en cualquier momento, es decir, al estado listo.



3. PLANIFICACIÓN DE PROCESOS

La planificación de procesos es el conjunto de mecanismos del SO que establece el orden en que los procesos se van a ejecutar.

3.1. Algoritmos de planificación

- ***FCFS(First Come, First Server)***
  - Es un algoritmo *no apropiativo* (no permite desalojar de la CPU un proceso que no haya acabado su ejecución)
  - Los procesos son despachados a la CPU en el mismo orden de llegada al sistema, siguiendo una *filosofia FIFO*(el primero en llegar es el primero en ser servido)
  - *VENTAJAS*: su simplicidad
  - *DESVENTAJA:* puede ocurrir un efecto convoy (todos los procesos esperan a que termine un proceso lento).
- ***Algoritmo Round-Robin(RR)***
  - Reparte el tiempo de asignación de la CPU de forma equitativa enre todos los procesos. Para ello se define un *quanto* de tiempo(tiempo de CPU que se le da a cada uno de los procesos).

ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

- La cola de procesos listos se trata como una cola circular que será recorrida por el planificador de la CPU asignando a cada proceso un intervalo de tiempo que será como mucho, el de un quanto.

- ***Algoritmo por prioridades (apropiativo y no apropiativo)***
  - Consiste en asignar un determinado valor de prioridad a cada proceso, y se escoge para ser despachado por la CPU el proceso con mayor prioridad.
  - DESVENTAJA: la *inanición*: si hay un flujo constante de procesos con alta prioridad, puede ocurrir que procesos con baja prioridad nunca obtengan la CPU.
  - SOLUCIÓN: *envejecimiento*: la prioridad de los procesos aumenta gradualmente de acuerdo con el tiempo que llevan en la cola de procesos listos.

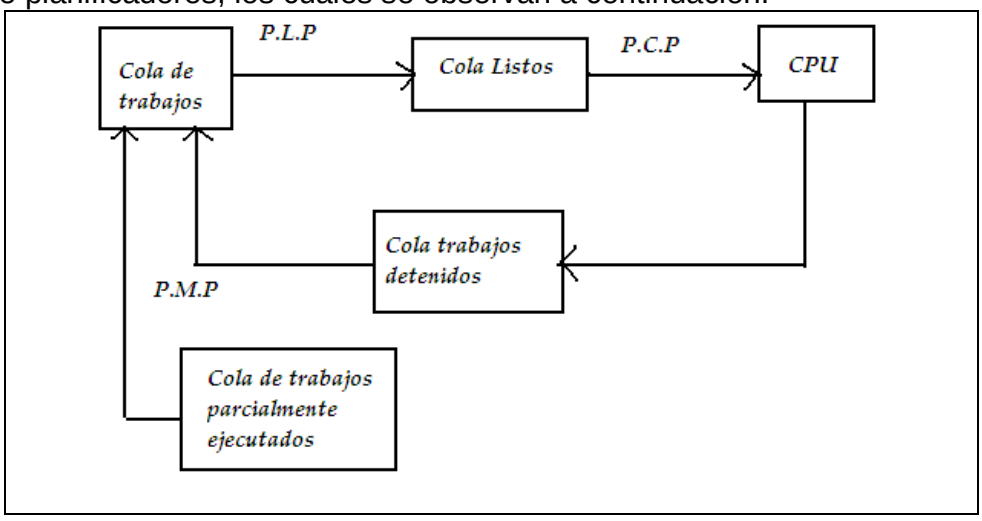
Actividad 2: Utilizar los siguientes algoritmos de planificación para poder planificar los siguientes procesos:

PROCESO	LLEGADA	CPU
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Algoritmo FCFS no apropiativo → P1, P2, P3, P4, P5 (van entrando en la CPU en orden de llegada).
Algoritmo RR (q=2) → P1, P2, P3, P4, P5, P1, P2, P3, P4, P5, P2, P4
Algoritmo por prioridades → estos procesos no tienen una asignación de prioridades.

3.2. Tipos de planificador

Existen 3 tipos de planificadores, los cuales se observan a continuación:



ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

- *Planificador a largo plazo (P.L.P):* Trabaja con la cola de trabajos, y selecciona el siguiente trabajo de lotes a ejecutar (es decir, el siguiente proceso que va a ir a la cola de listos).
- *Planificador a corto plazo (P.C.P):* Decide que proceso va a entrar en la CPU a continuación.
- *Planificador a medio plazo (P.M.P):* Su función escoger los procesos de memoria principal y pasarlos a disco (memoria secundaria) para reorganizarlos.

4. COORDINACIÓN DE PROCESOS

El hecho de que un SO multitarea ejecute a la vez varios procesos no solamente entraña la complejidad de la planificación sino que plantea un nuevo problema: LA COMPARTICIÓN de recursos.

4.1. Sección Crítica

La *sección crítica* de un programa es aquella que debe de acceder a recursos compartidos de manera exclusiva.
Hay que garantizar que cuando un proceso está en su sección crítica usando un recurso, ningún otro proceso puede estar ejecutando sus secciones críticas para ese recurso.
Las condiciones o requisitos que deben cumplir las soluciones propuestas para resolver el problema de la sección crítica son:

- **Exclusión mutua**: de todos los procesos con Sección crítica que quieren acceder a un mismo recurso, solo UNO gana el permiso de entrar en ella en un momento dado.
- **Progreso**: ningún proceso que no esté en su SC, ni tiene interés en entrar, puede evitar que otros procesos que sí quieren entrar en la SC lo hagan.
- **Espera limitada**: cuando un proceso quiere entrar en su SC, es necesario asegurar que lo hagan en un tiempo finito.
- **Velocidad relativa** de los procesos es NO NULA.

Existen unas soluciones propuestas para intentar conseguir estas cuatro condiciones, las podemos clasificar en:

- Soluciones Hardware
- Soluciones Software
- Soluciones con auxilio del SO.

4.1.1 Soluciones Hardware

El propio hardware implementa funciones que garantizan la exclusión mutua.
1. Inhabilitación de interrupciones
Los procesos continúan su ejecución hasta que terminen o se ejecute una interrupción. Mientras se está ejecutando su SC no pueden generase interrupciones. Esta solución asegura la exclusión mutua pero el rendimiento es bajo y es inadecuado para arquitecturas multiprocesador.
2. Instrucciones especiales de la máquina
Se trata de que el HW garantice accesos exclusivos a una posición de memoria determinada, mediante el uso de instrucciones máquina. Instrucción TEST AND SET: esta instrucción se usaría en la SC de

ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

entrada para asegurarse que el proceso entra en la SC. Hasta que el primer proceso no libere la instrucción "Test and Set" no podrá entrar otro proceso en la SC.
INCONVENIENTE: garantiza la exclusión mutua, pero NO la espera limitada, además no es solución adecuada para arquitecturas multiprocesadores.

4.1.2 Soluciones Software

Se basa en el uso de una variable compartida con valores 0 si no hay procesos en SC y 1 si sí los hay. No soluciona la *exclusión mutua*.

1. Alternancia estricta

Se trata de dar el *turno* a cada uno de los procesos de forma secuencial a través de una variable turno. El inconveniente de éste es que viola la propiedad de *progreso*.

2. Algoritmos

Son algoritmos que solucionan el problema de la sección crítica pero casi no se utilizan por la excesiva complejidad a la hora de implementarlos en un sistema real. Algunos de estos algoritmos son los de Peterson, Deker o Lamport.

4.1.3 Soluciones con auxilio del SO

Se basan en una *variable compartida* a la que solamente se puede acceder mediante dos acciones:

- WAIT: hace que un proceso espere hasta que la variable sea mayor que cero y la decrementa en uno.
- SIGNAL: hace que la variable se incremente en una unidad.

Su funcionamiento se basa en que un proceso ejecuta un *wait* al comienzo de su SC, y si más procesos hacen lo mismo, ninguno más accederá hasta que el proceso que se encuentre en la SC ejecute un *signal*.

Actividad 3: Hacer un esquema de las diferentes soluciones que adopta el sistema operativo frente a la sección crítica prestando gran atención a la diferencia entre utilizar el hardware o el software para solucionar este tipo de problemas.

5. ESQUEMAS DE GESTIÓN DE MEMORIA

Las políticas de gestión de memoria pueden agruparse en 3 grupos: políticas de sustitución, de carga, o de ubicación.

5.1. Política de sustitución

Son las que determinan QUÉ información hay que sacar de la memoria principal, es decir, son las responsables de la creación de zonas de memorias libres. Las políticas son:

LRU: La página que se sustituye es aquella que no ha sido utilizada durante el mayor tiempo, es decir, cuya última referencia es más antigua (usada menos recientemente). Esto supone que el pasado de una página es un buen indicador de su comportamiento futuro.

ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

FIFO: La página que se sustituye es la que lleva más tiempo en memoria principal, es decir, cuya primera referencia (la que le llevó a memoria principal) es más antigua. Para saber qué página se sustituye se lleva una cola de candidatos a sustitución y se va actualizando cada vez que hay un fallo de página (es decir, cada vez que se carga en memoria principal se actualiza).

RELOJ: mantiene las páginas en una lista circular con forma de reloj, con una manecilla que apunta a la tabla página más antigua. Cuando ocurre un error de página, se inspecciona la página a la que apunta la manecilla. La acción que se realiza depende del bit R:

Si R=0, retira la página de memoria, si R=1, limpia R y avanza la manecilla.

Actividad 4: Supongamos que tenemos una memoria virtual de 8 páginas y con 4 marcos de páginas y una cadena de referencia como la siguiente: 71321350346. Calcula el número de fallos de página que se produce con cada uno de los algoritmos LRU, FIFO Y RELOJ. ¿Cuál es el de mayor eficacia para la cadena de referencia especificada?

5.2. Política de carga

Determinan CUÁNDO hay que cargar información en la memoria principal. Existen dos políticas:

Carga bajo demanda: la falta de un bloque provoca una petición de carga, con lo que los algoritmos de ubicación y/o sustitución harán sitio en la memoria para el nuevo bloque.

Carga anticipatoria: cargan los bloques por adelantado, por lo que deben basarse en predicciones del comportamiento futuro del programa.

5.3. Política de ubicación

Determinan en QUÉ lugar de la memoria principal hay que colocar la información leída, es decir, deben elegir una parte de la zona de memoria libre. Existen varias técnicas:

Mejor ajuste (best fit): el proceso lo almacena en el menor hueco donde quepa ese proceso (así se desperdicia menos memoria). Sus desventajas son que proporciona en general los peores resultados porque crea un montón de huecos muy pequeños, y además es muy lento. En cambio, como este algoritmo busca el hueco más pequeño para el proceso, garantiza que el fragmento que se deja es lo más pequeño posible, y por lo tanto, se debe compactar más frecuentemente.

Primer ajuste (first fit): el proceso lo guarda en el primer hueco libre en el que quepa, de forma que se revisa la lista de huecos hasta encontrar un espacio lo suficientemente grande.

Peor ajuste (worst fit): el proceso lo almacena en el mayor hueco donde quepa ese proceso. El inconveniente es que se desperdicia mucha memoria, ya que en ese hueco también podría albergar otro proceso.

ISSN 1988-6047 DEP. LEGAL: GR 2922/2007 Nº 23 – OCTUBRE DE 2009

Actividad 5: Tenemos un ordenador con 1MB de memoria principal. El Sistema Operativo ocupa los primeros 124KB de memoria y el resto está disponible para los procesos de usuario. La siguiente tabla muestra información de una serie de procesos que han de ser ejecutados. Simula la evolución de la memoria usando los algoritmos de ubicación de mejor, peor y primer ajuste.

Proceso	*Llegada*	*Duración*	*Tamaño*
A	0	15	100kb
B	3	5	500kb
C	6	10	200kb
D	10	7	50kb
E	12	2	460kb
F	15	10	300kb
G	17	3	250kb

6. BIBLIOGRAFÍA

- Stalling, W. (1996). *Sistemas Operativos. Segunda Edición*: Prentice Hall.
- Tanembaum, A. (1996). *Sistemas Operativos Modernos*: Prentice Hall

Autoría

- María Catalá Carbonero
- IES Florencio Pintado, Peñarroya-Pueblonuevo, Córdoba
- mcata44@hotmail.com