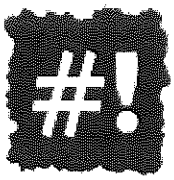


Preparador Informática

www.preparadorinformatica.com

TEMA 16. INFORMÁTICA / S.A.I.

SISTEMAS OPERATIVOS: GESTIÓN DE PROCESOS.

TEMA 16 INF / SAI: SISTEMAS OPERATIVOS: GESTIÓN DE PROCESOS.

1. INTRODUCCIÓN

Hoy en día, los ordenadores constan de uno o más procesadores, memoria principal, discos, interfaces de red, periféricos y otros dispositivos de entrada/salida. En general se tratan de sistemas complejos, por lo que si los programadores tuvieran que comprender el funcionamiento de todos estos componentes les resultaría muy difícil desarrollar programas.

Por esta razón, los ordenadores están equipados con una capa de software llamada sistema operativo, que interacciona directamente con el hardware del equipo encargándose de controlar todos los recursos del mismo y presentar al usuario una interfaz más fácil de entender y programar.

El mercado de los sistemas operativos está ampliando sus horizontes para dirigirse también a otros equipos. Prueba de ello, es que Microsoft apuesta por estar presente en todos los dispositivos con el lanzamiento de **Windows 10 IoT Core**. Se trata de una versión reducida de Windows 10 optimizada para pequeños Mini-PCs (Raspberry Pi 2 y 3, Dragonboard 410c, etc.) que son productos relacionados con IoT (*Internet of Things*) y, en general, del sector de embebidos.

De igual forma, actualmente existen distribuciones Linux para una amplia variedad de dispositivos y fines, como por ejemplo versiones de propósito general (Red Hat, Fedora, Debian, etc.), de auditoría informática (BlackArch), para pequeños Mini-PCs (Raspbian), para entornos educativos (Guadalinex, Lliurex, MAX...), etc.

El presente tema está dedicado a estudiar la importancia del sistema operativo como responsable de las actividades relacionadas con la gestión de procesos, como son la creación de procesos, planificación de procesos y la comunicación y sincronización de los mismos.

2. PROCESOS

Un proceso es un programa o un fragmento de programa en ejecución. Cada proceso necesita una serie de recursos, siendo el sistema operativo el encargado de proporcionárselos. En función del tipo de proceso, los recursos necesarios pueden ser diferentes, pero, por regla general, todos necesitan un espacio en memoria y un tiempo de uso del microprocesador (CPU).

En el sistema operativo, cada proceso se representa por medio de su propio bloque de control de proceso (PCB, Process Control Block). Un PCB es un registro de datos que contiene información básica del proceso como:

- Su estado actual.
- Su PID (número identificador de proceso).
- Valores de registros asociados a él, como el contador de programa, los punteros, acumuladores, etc.
- Valores de los recursos asignados: espacio en memoria, archivos, E/S, etc.

Los procesos, desde su creación, tienen la capacidad de comunicarse y sincronizarse con otros procesos y con recursos del sistema. Este hecho da lugar a diferentes tipos de procesos:

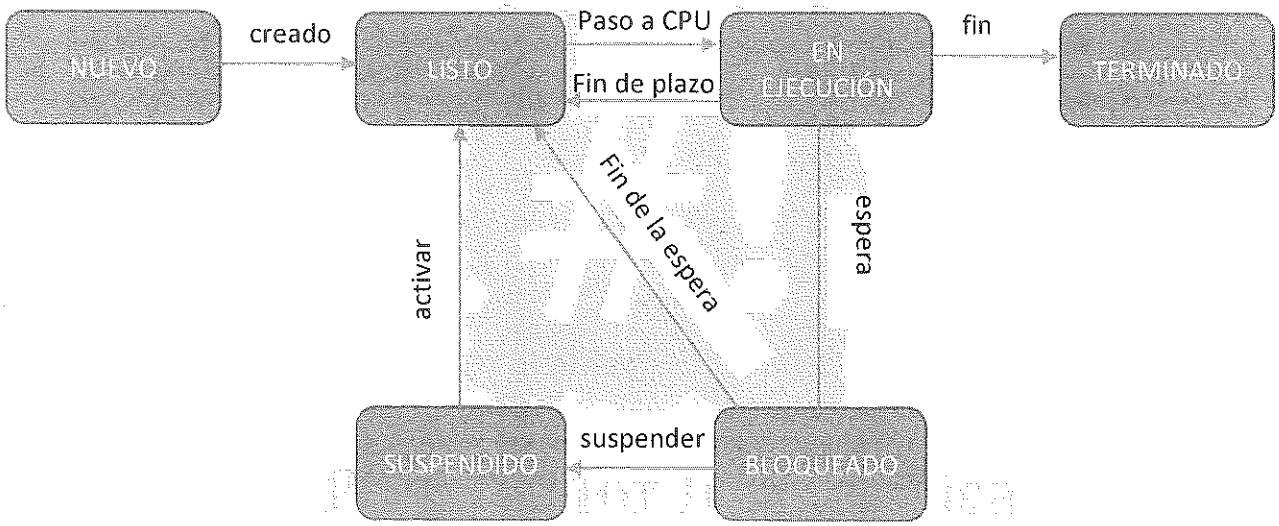
- **Independientes**: no se comunican con otros procesos. Estos tipos de procesos apenas existen.
- **Cooperativos**: se comunican y sincronizan para realizar una actividad común.
- **Competitivos**: necesitan hacer uso del mismo recurso y, por consiguiente, competirán por él.

En cualquier caso, la ejecución de los procesos exigirá concurrencia, cualidad que deberá gestionar el sistema operativo gracias a la técnica denominada **multiprogramación**, que permite que dos o más procesos puedan ejecutarse "a la vez".

2.1. ESTADOS DE UN PROCESO

Los diferentes estados en el ciclo de vida de un proceso son:

- **Nuevo**. Proceso nuevo creado.
- **Listo**. Proceso que está esperando la CPU para ejecutarse.
- **En ejecución**. Proceso que actualmente está ejecutándose en la CPU.
- **Bloqueado**. Proceso que está a la espera de que finalice una E/S.
- **Suspendido**. Proceso que se ha llevado a la memoria virtual para liberar memoria principal.
- **Terminado**. Proceso que ha finalizado.



2.2. HILOS

Los procesos se pueden dividir en hilos (threads) que pueden ser planificados para su ejecución. Realizar cambios de contexto entre hilos de un mismo proceso, es más rápido y menos costoso que el cambio de contexto entre procesos, ya que comparten la información de las variables de ese proceso.

La capacidad del sistema operativo de mantener varios hilos de ejecución dentro del mismo proceso se conoce con el nombre de multihilo. Si no existe, entonces se habla de monohilo. En general, todos los sistemas operativos modernos son multihilo.

3. GESTIÓN DE PROCESOS

El sistema operativo lleva a cabo la gestión de los procesos por medio de un componente llamado gestor de procesos, cuyo objetivo es controlar la ejecución de los programas.

Las principales funciones del gestor de procesos son:

- La creación, suspensión, reanudación, eliminación y planificación de procesos.
- La comunicación y sincronización entre procesos.

3.1. COMUNICACIÓN ENTRE PROCESOS

Se denomina comunicación entre procesos a los mecanismos que proporciona un sistema operativo para que estos puedan intercambiar información.

Estos mecanismos se pueden clasificar en dos grupos:

A) **Memoria compartida**: los procesos comparten algunas variables en memoria para el intercambio de información.

B) **Intercambio de mensajes**: el sistema operativo proporciona mecanismos para el envío y recepción de mensajes. Se distinguen los siguientes mecanismos:

- **Sockets**: utiliza un flujo de datos a través de una interfaz de red.
- **Puertos**: en los sistemas cliente/servidor, cada servidor tiene un puerto asignado por el cual recibe los mensajes de los clientes.
- **Tuberías o pipes**: fueron creadas en UNIX, y consisten en una cadena de procesos de forma tal que la salida de cada proceso es la entrada del próximo.
- Llamada a procedimiento remoto o Remote Procedure Call (**RPC**): mecanismo de alto nivel para realizar la comunicación entre procesos en sistemas distribuidos.

3.2. SINCRONIZACIÓN DE PROCESOS

3.2.1. CONCURRENCIA DE PROCESOS

Se dice que dos o más procesos son concurrentes si se están ejecutando a la vez. Si el sistema es **multiprocesador**, la simultaneidad es real, y entonces se habla de **paralelismo**. En cambio, en un sistema **monoprocesador** la simultaneidad es simulada por los cambios de contexto entre procesos y se habla de **concurrencia aparente** o **pseudoparalelismo**.

La multiprogramación es el conjunto de técnicas que permiten la ejecución concurrente de procesos

En un sistema operativo multiprogramado los procesos compiten por el acceso a unos recursos compartidos o cooperan para comunicar información. Ambas situaciones son tratadas por el sistema operativo mediante **mecanismos de sincronización**.

Cuando los procesos acceden a recursos compartidos pueden producirse resultados inesperados y errores. La solución a estos problemas consiste en impedir que más de un proceso lea o escriba simultáneamente datos compartidos (**exclusión mutua**).

La **sección crítica** es la parte del programa en la cual se accede al recurso compartido. De este modo, si se evita que dos procesos entren simultáneamente en una sección critica, se evitarán las **condiciones de competencia**.

3.2.2. MECANISMOS DE SINCRONIZACIÓN DE PROCESOS

Los principales mecanismos de sincronización de procesos son:

A) **Inhabilitación de interrupciones:** El proceso que entra en una sección crítica desactiva todas las interrupciones y las activa cuando sale. Este método no es recomendable ya que si el proceso tiene algún problema el sistema operativo no recuperaría el control.

B) **Algoritmo de Peterson:** Se trata de una solución software al problema de la exclusión mutua que no requiere que un proceso sea bloqueado por

otro proceso que no está dentro de su sección critica. Su principal inconveniente es que requiere de espera ocupada, lo cual implica un gasto inútil de tiempo de CPU.

C) **Instrucción TSL (Test and Set Lock):** La instrucción TSL lee en un registro el contenido de la palabra de una dirección de memoria y carga en ésta un valor distinto de cero. Ambas operaciones son indivisibles. Ningún otro procesador puede acceder a la palabra de memoria hasta que termine la instrucción, ya que la CPU que ejecuta la instrucción TSL bloquea el bus de memoria.

D) **Semáforos:** Este mecanismo fue inventado por Dijkstra y consiste en una variable de tipo entero no negativo sobre la que se definen dos funciones que se ejecutan de forma atómica: P (reservar) y V (liberar). Por ejemplo, si se desea controlar el acceso a un recurso, se inicializaría la variable semáforo a 1 indicando que el recurso está libre y cuando un proceso necesite utilizarlo ejecutaría la función P sobre dicho semáforo. El proceso que ejecute P sobre un semáforo a 0 quedará bloqueado hasta que otro proceso ejecute V sobre el mismo semáforo.

E) **Monitores:** Un monitor es un mecanismo que contiene los datos y los procedimientos necesarios para asignar un recurso compartido. Los procesos que deseen usar el monitor cuando ya hay un proceso activo deben esperar. Como la exclusión mutua está garantizada, se evitan los problemas de concurrencia.

3.2.3. INANICIÓN E INTERBLOQUEOS

Los mecanismos de sincronización de procesos pueden presentar dos problemas:

- **Inanición**: se produce cuando un proceso o hilo no consigue nunca el acceso a un recurso compartido.
- **Interbloqueo**: se produce cuando dos o más procesos esperan por un evento que no va a ocurrir.

3.3. PLANIFICACIÓN DE PROCESOS

Los procesos se ejecutan según un orden establecido por el sistema operativo. La forma en la que el sistema operativo gestiona los procesos es lo que se conoce como **planificación** y la herramienta que lo hace recibe el nombre de **planificador** (**scheduler**). Para realizar esta tarea se utilizan los algoritmos de planificación, que se pueden clasificar en dos tipos:

A) **Apropiativos:** los procesos pueden ser expulsados de la CPU.
   - Mayor coste de implementación
   - Mejor optimización del uso de la CPU
   - Utilizados en sistemas multiusuario y multitarea

B) **No apropiativos:** los procesos se ejecutan hasta que terminan o se bloquean.
   - Sencillos
   - Rendimiento menor
   - Utilizados en sistemas batch o monousuario

Los principales algoritmos de planificación son:

3.3.1. ALGORITMO FCFS

El algoritmo FCFS (First-Come, First-Served) es un algoritmo no apropiativo, que consiste en atender a los procesos por estricto orden de llegada a la cola de procesos listos. Cada proceso se ejecuta hasta que termina, o hasta que hace una llamada de E/S.

Procesos	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

	0	1	2	3	4	5	6	7	8
A	X	X	X	X	X				
B						X	X	X	
C									X

3.3.2. ALGORITMO ROUND ROBIN

El algoritmo Round Robin es un algoritmo apropiativo que consiste en darle a cada proceso un intervalo de tiempo de ejecución (quantum). Cada vez que se agota el quantum el proceso que está en ejecución deja de ejecutarse y se sitúa al final de la cola de procesos listos. La cola de procesos listo se gestiona con una política FIFO.

Procesos	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

Ejemplo: quantum= 2

	0	1	2	3	4	5	6	7	8
A	X	X				X	X		X
B			X	X				X	
C					X				

3.3.3. ALGORITMO SJF

El algoritmo SJF (Shortest Job First) o también conocido como SPN (Shortest Process Next) es un algoritmo no apropiativo que consiste en ejecutar el proceso más corto de los que hay en la cola de procesos listos.

Procesos	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

	0	1	2	3	4	5	6	7	8
A	X	X	X	X	X				
B							X	X	X
C						X			

3.3.4. ALGORITMO SRT

El algoritmo SRT (Shortest Remaining Time) es un algoritmo apropiativo que consiste en ejecutar el proceso que menos tiempo de CPU le quedé para acabar. De este modo, si se está ejecutando un proceso y llega otro más corto, el sistema operativo le quitará la ejecución de la CPU para asignársela al proceso más corto.

Procesos	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

	0	1	2	3	4	5	6	7	8
A	X					X	X	X	X
B		X		X	X				
C			X						

3.3.5. ALGORITMO POR PRIORIDADES

El algoritmo por prioridades consiste en ejecutar primero el proceso que tiene más prioridad. Cada proceso tiene asignada una prioridad. El planificador selecciona el proceso con prioridad más alta (a igual prioridad se selecciona con FCFS). Las prioridades pueden ser dinámicas (cambian con el tiempo) o estáticas (se mantienen).

4. GESTIÓN DE PROCESOS EN WINDOWS Y EN LINUX

A) WINDOWS

A nivel de usuario gran parte de las operaciones de gestión de procesos en Windows se hacen desde el Administrador de tareas. Esta herramienta es accesible desde la combinación de teclas [Ctrl]+[Alt]+[Supr] o pulsando con el botón derecho en la barra de tareas y elegir "*Iniciar administrador de tareas*"

B) LINUX

Linux dispone de un gran abanico de herramientas, sobre todo a nivel de consola, para gestionar los procesos. Algunos de los principales comandos son:

- **ps:** muestra un listado con el estado de los procesos.
- **kill:** permite enviar señales a los procesos para cambiar su estado.
- **nice:** permite cambiar la prioridad a un proceso

5. CONCLUSIÓN

En el presente tema se ha presentado una visión global de la gestión de procesos en los sistemas operativos. En la parte final del tema se han estudiado desde un punto de vista teórico los diferentes algoritmos de planificación de procesos, describiendo sus características. En la práctica, los sistemas operativos actuales utilizan un sistema híbrido de varias colas en las que se aplican diferentes algoritmos con la finalidad de optimizar los recursos y los tiempos de respuesta.

6. BIBLIOGRAFÍA

- Tanenbaum A. **Sistemas operativos modernos**. Editorial Prentice Hall
- Stallings W. **Sistemas operativos**. Editorial Prentice Hall
- Deitel, H. **Introducción a los Sistemas Operativos**. Addison-Wesley.
- Prieto, A. y otros. **Introducción a la informática**. Editorial McGraw-Hill.
- http://atc.ugr.es/APrieto_videoclases Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada.
- www.microsoft.com/windows
- www.linux.com (Web oficial de Linux Foundation)
- www.xataka.com (Web de actualidad sobre tecnología e informática).