Procesos e hilos

Pedro O. Pérez M., MTI

Multiprocesadores
Tecnológico de Monterrey

pperezm@tec.mx

08-2019

Contenido

Introducción

Multitarea, multiprogramación y multiprocesamiento
- Multitarea
- Multiprogramación
- Multiprocesamiento

Procesos e hilos
- Procesos
- Hilos

Sincronización de procesos (o hilos)
- Comunicación entre procesos (o hilos)
- Problema de la sección crítica
- Interbloqueos

Introducción

- Las primeras computadoras de escritorio solo contaban con un CPU, y solo eran capaces de ejecutar un programa la vez. Posteriormente vino la multitarea y fue posible que las computadoras pudieran ejecutar múltiples programas (tareas o procesos) al mismo tiempo.
- Aunque, para ser sinceros, no se ejecutan al mismo tiempo. El CPU es compartido entre los programas y el sistema operativo el encargado de determinar cuál programa y por cuánto tiempo.
- Un proceso es la unidad de trabajo de la mayoría de los sistemas operativos. Un proceso puede ser visto como un programa en ejecución.

Multitarea

La multitarea, o tiempo compartido (time sharing, multitasking), es una extensión lógica en la que el CPU cambia de trabajo con tanta frecuencia que los usuarios pueden interactuar con cada trabajo mientras se está ejecutando.

- El tiempo de espera debe ser menor a un segundo.
- Cada usuario tiene, al menos, un programa ejecutándose en la memoria (proceso).
- Existe muchos trabajos listos para ejecutarse (calendarización del CPU).
- Si los procesos no ¸caben.^en memoria, existe un proceso que mueve a otro procesos desde y hacia memoria para que haya espacio en la memoria (swapping). Este es lo que se conoce como memoria virtual (memoria secundaria).

Multiprogramación

La multiprogramación (multiprogramming) se desarrolla como una solución para el uso eficiente de un servidor:

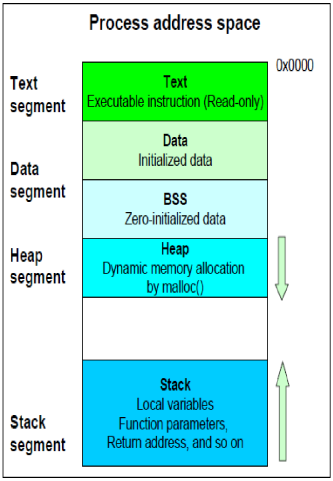
- Un solo usuario no puede mantener al CPU y a los dispositivos de E/S ocupados toso el tiempo.
- La multiprogramación organiza los trabajos para que el CPU siempre tenga algo que hacer.
- Existe una fila de trabajos (o tareas) en memoria. Desde esta fila, un trabajo es seleccionado a través de un algoritmo de calendarización.
- Cuando un trabajo está esperando a que termine una operación de E/S, el sistema operativo lo desaloja para poder ejecutar otro trabajo.

Multiprocesamiento

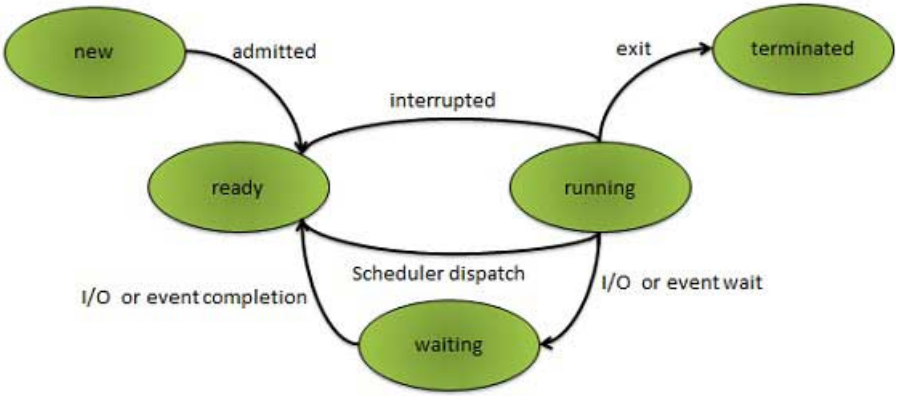
- Sistemas monoprocesador: pueden contar con procesadores de propósito especial (manejo de dispositivos, memoria de vídeo, etc.), sin embargo no se consideran parte del procesador principal.
- Sistemas multiprocesador:
  - Han crecido en uso e importancia.
  - También conocidos como sistemas paralelos, sistemas estrechamente acoplados.
  - Ventajas: mayor rendimiento, economía a escala, confiabilidad.
  - Dos tipos: asimétricos (un procesador controla los demás), simétricos (cada procesador realiza la tarea que haya disponible).

Procesos

- Un proceso es más que el código del programa, que a veces es conocido como el segmento del código.
- También incluye la actividad actual, representada por el valor del contador del programa y los valores de los registros del procesador.
- Un proceso, generalmente, también incluye una pila que contiene datos temporales (como los parámetros de las funciones, la dirección de retorno y las variables locales) y un segmento de datos que contiene variables globales.
- Un proceso también puede incluir un "heap", que es la memoria que se asigna dinámicamente durante el tiempo de ejecución del proceso.

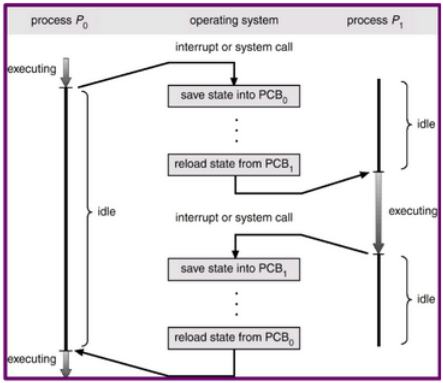


Estados de un proceso



Cambio de contexto

- Ciertas operaciones hacen que el sistema operativo cambie la tarea actual que se ejecuta en el CPU y ejecute una rutina del sistema u otro proceso. Cuando se produce esta ïnterrupción", el sistema necesita guardar el contexto actual del proceso que se está ejecutando en el CPU para que se pueda restaurar posteriormente.
- El contexto está representado en el PCB del proceso; incluye el valor de los registros del CPU, el estado del proceso y la información de la administración de la memoria.
- En general, realizamos un "guardado"(state save) del estado actual del CPU y luego una restauración del estado para reanudar las operaciones.



Operaciones sobre procesos

- Creación de procesos:
  - Un proceso (padre) puede crear más procesos (hijos).
  - Un proceso creado puede requerir sus propios recursos o estar restringido a los recursos del proceso padre.
  - Cuando un proceso crea a otro:
    - El padre continúa ejecutándose concurrentemente con su hijo.
    - El padre espera hasta que alguno o todos los hijos han terminado de ejecutarse.
  - Con respecto al espacio de direcciones:
    - El proceso hijo es un duplicado del padre.
    - El proceso hijo carga un nuevo programa.

- Terminación de procesos:
  - Cuando un proceso termina, éste debe indicar a su proceso padre un valor de terminación.
  - Un proceso padre puede termina la ejecución de cualquiera de sus hijos.
    - El proceso hijo ha excedido alguno de los recursos asignados.
    - La tarea asignada del proceso hijo ya no es necesaria.
    - El padre abandona el sistema. El sistema operativo no permite que un proceso continúe si su padre ha terminado.

Definición

- Los hilos a veces se llaman procesos livianos. Ambos, proceso e hilo, proporcionan un entorno de ejecución, pero la creación de un nuevo hilo requiere menos recursos que la creación de un nuevo proceso.
- Los hilos existen dentro de un proceso; cada proceso tiene al menos un hilo. Los hilos comparten los recursos del proceso, incluida la memoria y los archivos abiertos. Este lo convierte en una comunicación eficiente, pero potencialmente problemática.

¿Por qué usarlos?

¿Por qué deberíamos tener múltiples hilos de ejecución dentro de un único contexto de proceso?

- Considera, por ejemplo, una aplicación GUI donde el usuario puede emitir un comando que requiere mucho tiempo para terminar (por ejemplo, bajar un archivo muy grande). A menos que se diseñe este comando para que se ejecute en un hilo separado, no se podrá interactuar con la GUI principal de la aplicación (por ejemplo, para actualizar una barra de progreso) porque no responderá mientras se lleva a cabo el cálculo.

Por supuesto, el diseño de aplicaciones multihilos requiere que el desarrollador maneje situaciones que simplemente no concurrente cuando se desarrollan aplicaciones secuenciales de proceso único. Por ejemplo, cuando dos o más hilos intentan acceder y modificar un recurso compartido (condición del carrera), el desarrollador debe asegurarse de que esto no dejará al sistema en un estado incoherente o de interbloqueo.

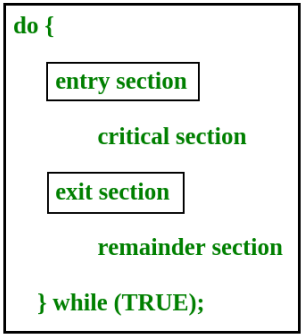
Comunicación entre procesos

- Los procesos que se ejecutan concurrentemente puede ser independientes o cooperativos.
- Un proceso cooperativo es aquel que puede afectarse o verse afectado por otro procesos que se estén ejecutando en el sistema. Los procesos cooperativos pueden compartir directamente un espacio de direcciones lógico o compartir los datos sólo a través de archivos o mensajes.

- Existen varias razones para proporcionar un entorno que permita la cooperación entre procesos:
  - Compartir información.
  - Acelerar los cálculos.
  - Modularidad.
  - Conveniencia.
- Cuando varios procesos manipulan o acceden a los mismos datos concurrentemente y el resultado de la ejecución depende del orden concreto en que se produzcan los accesos se conoce como condición de carrera.

¿Qué es una sección crítica?

- Considera un sistema que consta de n procesos $(P_0, P_1, ..., P_{n-1}, P_n)$. Cada proceso tiene un segmento de código llamado sección crítica.
- Cuando un proceso está ejecutando su sección crítica, ningún otro proceso puede ejecutar su correspondiente sección crítica.
- Cualquier solución al problema de la sección crítica deberá satisfacer tres requisitos:
  - Exclusión mutua.
  - Progreso.
  - Espera limitada.



- El problema de la sección crítica puede resolverse de forma fácil en un entorno de un solo procesador si pudiéramos impedir que se produjeran interrupciones mientras se está modificando una variable compartida.
- Lamentablemente, esta solución no resultada adecuada en un entorno multiprocesador. Desactivar las interrupciones en un sistema multiprocesador puede consumir mucho tiempo, ya que hay que pasa el mensaje a todos los procesadores.

Semáforos

- Un semáforo S es una variable entera a la que, sin contar a la inicialización, sólo se accede mediante dos operaciones atómicas estándar: wait (P - proberen)y signal (V - verhogen).
- Los sistemas operativos diferencian entre semáforos contadores (más de una instancia a controlar) y semáforos binarios (una sola instancia a controlar).

Semaphore Structure:

```
typedef struct {
    int value;
    struct process *list;
} semaphore;
```

Wait Operation:

```
wait(semaphore *S) {
        S->value--;
        if (S->value < 0) {
            add this process to S->list;
            block();
        }
}
```

Signal Operation:

```
signal(semaphore *S) {
        S->value++;
        if (S->value <= 0) {
            remove a process P from S->list;
            wakeup(P);
        }
}
```

Interbloqueos

Un proceso solicita recursos y, si los recursos no están en ese momento, el proceso pasa a al estado de espera. Es posible que, algunas veces, un proceso en espera no pueda nunca cambiar de estado, porque los recursos que ha solicitado están ocupados por otros procesos que a su vez estén esperando otros recursos. Cuando se produce una situación como esta, se dice que ha ocurrido un interbloqueo.

Una situación de interbloqueo puede surgir si se dan simultáneamente las sigueitne cuatro condiciones:

- **Exclusión mutua**. Al menos un recurso debe estar en modo compartido.
- **Retención y espera**. Un proceso debe estar reteniendo al menos un recurso y esperando adquirir otro recursos adicionales que estén detenidos por otro procesos.
- **Sin desalojo**. Los recursos no pueden ser desaolojados.
- **Espera circular**. Debe existir un conjunto $(P_0, P_1, ..., P_n)$ de procesos en espera, tal que P_0 está esperando un recurso detenido por P_1, P_1 esté esperando a un recurso retenido por P_2, ..., P_{n-1} esté esperando a un recurso retenido por P_n y P_n esté esperando a un recurso retenido por P_0.