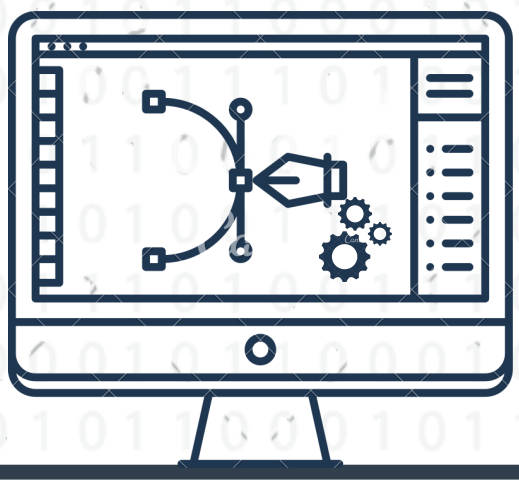
ESCUELA SUPERIOR DE COMPUTO



APLICACIONES PARA COMUNICACIONES EN RED

Tarea #2
Sincronización

Alumnos:
Caxantheje Ortiz Jazmin Lizeth
Lorenzo Pioquinto Alejandro
Rubio Haro Rodrigo R.

Profesor: Rangel Gonzalez Josue

CDMX. OCTUBRE, 2022.

INSTITUTO POLITÉCNICO NACIONAL

1. Sincronización

En muchos casos, los procesos se reúnen para realizar tareas en conjunto, a este tipo de relación se le llaman procesos cooperativos. Para lograr la comunicación los procesos deben sincronizarse, de no ser así puede haber errores no deseados. La sincronización es la transmisión y recepción de señales que tienen por objeto llevar a cabo el trabajo de un grupo de procesos cooperativos.

¿Para qué sirve?

Sirve para la coordinación y cooperación de procesos para asegurar la comparación de recursos de cómputo. La sincronización entre procesos es necesaria para prevenir y/o corregir errores de comunicación debido al acceso concurrente a recursos compartidos, tales como estructuras de datos o dispositivos de E/S. También permite intercambiar señales de tiempo entre procesos cooperantes para garantizar las relaciones específicas de procedencia impuestas por el problema a resolver.

¿Qué es sección crítica?

Se dice sección crítica cuando en un programa concurrente, hay una parte del código de un programa en la que se accede a un recurso compartido al cual no deben acceder más de un proceso o un hilo hijo en ejecución. La sección crítica por lo general termina en un tiempo determinado y el hilo, proceso o tarea solo tendrá que esperar un periodo determinado de tiempo para poder entrar. Se necesita un mecanismo de sincronización en la entrada y salida de la sección crítica para asegurar la utilización en exclusiva del recurso, por ejemplo un semáforo.
Solo un proceso puede estar en una sección crítica a la vez.

¿Qué es exclusión mútua?

Los algoritmos de exclusión mutua (comúnmente abreviada como mutex por mutual exclusión) se usan en programación concurrente para evitar el ingreso a sus secciones críticas por más de un proceso a la vez. La sección crítica es el fragmento de código donde puede modificarse un recurso compartido.
La mayor parte de estos recursos son las señales, contadores, colas y otros datos que se emplean en la comunicación entre el código que se ejecuta cuando se da servicio a una interrupción y el código que se ejecuta el resto del tiempo. Se trata de un problema de vital importancia porque, si no se toman las precauciones debidas, una interrupción puede ocurrir entre dos instrucciones cualesquiera del código normal y esto puede provocar graves fallos.

La técnica que se emplea por lo común para conseguir la exclusión mutua es inhabilitar las interrupciones durante el conjunto de instrucciones más pequeño que impedirá la corrupción de la estructura compartida (la sección crítica). Esto impide que el código de la interrupción se ejecute en mitad de la sección crítica.

¿Qué es un deadlock?

Los deadlocks (o interbloqueos por su nombre en español) son problemas especiales de concurrencia que impiden que una transacción pueda concretarse. Ocurren cuando dos sesiones están esperando recursos que están siendo bloqueados por el otro. Esto terminaría en una espera infinita, por lo que el sistema operativo escoge terminar una de esas sesiones, a la que se le denomina 'víctima de deadlock'.

Tipos de esperas:

- Activa: La espera activa es una técnica en la cual un proceso comprueba continuamente si una condición se cumple o si se produce un evento. Puede ser una estrategia válida en algunas circunstancias especiales, sobre todo en la sincronización de procesos en los sistemas con múltiples procesadores (SMP). Debe ser evitada, ya que consume tiempo de CPU sin realizar ninguna operación.
- Pasiva: En programas concurrentes es posible que un proceso nunca llegue a hacer nada si el planificador o el control de los recursos compartidos respectivamente no permite que el proceso pueda cumplir con sus objetivos. Es decir, el proceso está sometido a una espera infinita, o en otras palabras, sufre una inanición.

¿Cuáles mecanismos de sincronización existen?

Hay una gran variedad de mecanismos de sincronización, algunos son:

- Semáforos.
- Mutex.
- Candados.
- De barrera.
- De cola.
- Variables de condición.
- Señales.
- Paso de mensajes.
- Tuberías (pipes).

Semáforo Posix.

Los semáforos POSIX permiten que los procesos y los subprocesos hagan una sincronización de sus acciones. Un semáforo es un número entero cuyo valor nunca puede caer por debajo de cero. Se pueden realizar dos operaciones con semáforos: incrementar el valor del semáforo en uno (sem_post (3)); y disminuir el valor del semáforo en uno (sem_wait (3)).

- Para usar los semáforos Posix, debemos incluir: semaforo.h
- Para bloquear un semáforo o esperarlos, usamos la función: sem_wait: int sem_wait (sem_t * sem);
- Para liberar o señalar un semáforo, usamos la función: sem_post: int sem_post (sem_t * sem);