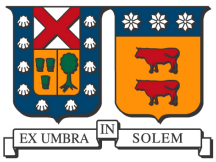

Universidad Técnica Federico Santa María
Departamento de Informática

Tarea 3 - Sistemas Operativos - INF246

Benjamin Jorquera *201473521-9*

Introducción

En informática, un semáforo es un tipo de datos variable o abstracto que se utiliza para controlar el acceso a un recurso común mediante múltiples procesos en un sistema concurrente, como un sistema operativo multiprogramación. Para poder entender mejor su comportamiento, se procederá a explicar en el siguiente informe la implementación y el análisis de un algoritmo que resuelva el problema de escritura/lectura en un mismo archivo a través de 2 procesos de forma concurrente.

1 Resultados y Desarrollo

1.1 Resolución

El algoritmo cuenta con una funcion main principal y dos funciones extras, que haran el trabajo de escritura y lectura descrito anteriormente, ahora bien, ¿Como podriamos asignar recursos a dichas funciones, para que trabajen concurrentemente sobre un mismo archivo?. Para poder resolver dicha pregunta, se utilizaron 2 semaforos de tipo "contadores" y Multi-threading, la razon se de dicho tipo de semaforos centra en el hecho de poder entender y ocupar sus funciones principales: wait() y signal(), funciones que no implementa un semaforo binario de tipo mutex; ademas, el algoritmo cuenta con dos threads (hebra o hilo en español), estos son considerados como procesos ligeros, ya que pueden compartir informacion (en esta caso el archivo que queremos modificar) y llegan a mejorar el rendimiento a traves del paralelismo, llamaremos subprocesos a estos threads.

En palabras mas simples, el algoritmo declara los semaforos y los subprocesos, los inicializa de tal manera que cada subproceso llama a la función de escritura y de lectura respectivamente, y así, mientras el usuario lo desee, podrá escribir en el archivo mediante un subproceso y luego leer lo que contiene mediante el otro. Esto es lograble gracias a la acción de los semaforos, que mientras un subproceso esta realizando su tarea, estos llaman a la función wait(), que como su nombre lo dice, esperan hasta recibir una señal, y cuando terminan, envían esa señal al semaforo correspondiente, para que así trabajen concurrentemente y no ocurran de manera simultanea, priorizando además el proceso de escritura.

1.2 Paso a paso

Primero se declaran las variables globales: los semaforos, s1 y s2, el archivo compartido y las dos funciones que se ocuparan, llamadas funcion1 y funcion2, tanto el tipo como los parametros de ambas son puntero a void.

Luego en el main, declaramos los threads, aqui es donde se inicializan los semaforos (sem_init) y los subprocesos (pthread_create y pthread_join) que harán lo siguiente:

Funcion1: tras declarar un arreglo de string, el usuario escribe por pantalla, lo que luego es reescrito en el archivo, para luego llamar a sem_post, la cual manda una señal al semaforo s1, y finalmente sem_wait para esperar la lectura del archivo. Notar que este subproceso cuenta con una condición de que si el usuario desea salir del programa, debe pulsar el entero 0, al hacerlo llama a sem_post para que el subproceso 2 de lectura no quede esperando infinitamente y hace un break.

Funcion2: Este subproceso llama inmediatamente a sem_wait, ya que no tiene aun nada que leer, cuando recibe la señal del subproceso 1 (si es que el usuario no ha pulsado 0) declara un arreglo de string y lee el contenido del archivo, escrito anteriormente por el subproceso 1, luego llama a sem_post.

Finalmente, cuando el usuario desea salir del programa, los semaforos se destruyen. Así termina este algoritmo resolutivo.

Nota: Se usa la función sleep() para que se vea como "animación".

LaTeX