Sincronización de procesos con mutexes y variables de condición - Opcional 3

Pablo Mouriño Lorenzo, Diego Pérez Álvarez

Sistemas Operativos II
Grupo 04

Resumen

En este informe se modificará la solución obtenida anteriormente para el problema del productor-consumidor usando mutexes y variables de condición, con el objetivo de implementar un tercer tipo de procesos que lea la información que le pasan los consumidores y sume los valores de todos los caracteres numéricos que lean en total los D hilos de este tipo.

Palabras clave: carreras críticas, productor-consumidor, hilos, sincronización, mutexes

I. Introducción

A partir de la solución obtenida en el ejercicio 1, añadiremos D hilos que asumirán la función de 'consumidores finales' de modo que reciban los caracteres numéricos que llegan al consumidor y sumen en una variable compartida los valores numéricos de los mismos, para devolver el resultado final al finalizar la lectura de todos los archivos.

Para llevar a cabo esto seguiremos el enfoque del programa original, utilizando un nuevo mutex y variables de condición para gestionar el acceso al buffer compartido entre los hilos 'consumidores' y los 'consumidores finales'.

En este informe se incluirá un apartado sobre el programa, cómo ejecutarlo y su funcionamiento. También se incluirá un apartado de conclusiones en el que desarrollaremos los resultados obtenidos.

II. Programa

Al igual que el programa principal, este se ha desarrollado en un único programa, escrito en C, de forma que se crean los hilos productores, consumidores y consumidores finales a partir del programa. A cada productor se le deberá designar un archivo de texto del cual extraerá únicamente los caracteres alfanuméricos, colocándolos posteriormente en el buffer compartido, el cual tendrá un funcionamiento de cola FIFO. A continuación, los consumidores tomarán los caracteres del array y los colocarán en archivos de texto de salida, creados anteriormente por cada consumidor.

Además de este comportamiento, idéntico al del problema del productor-consumidor habitual, los consumidores deberán también identificar los caracteres que son de tipo numérico y pasárselos a un buffer compartido con un nuevo tipo de hilos, los 'consumidores finales'. Estos se encargarán de recibir estos valores, calcular su valor numérico, y sumarlo a una variable compartida que irá almacenando la suma resultante para imprimirla por pantalla al finalizar el

proceso.

Será imprescindible controlar el acceso a las nuevas regiones críticas y las variables de condición necesarias, para así evitar la aparición de carreras críticas.

A. Ejecución

Para la ejecución de este programa es suficiente con compilarlo de la siguiente manera, enlazando la biblioteca `pthread.h` de ser necesario:

gcc -Wall -o opcional3 opcional3.c -lpthread

Es necesario pasarle 5 parámetros por línea de comandos para que se ejecute correctamente: `P` (número de hilos que actuarán como Productores), `C` (número de hilos que actuarán como Consumidores), `D` (número de hilos que actuarán como Consumidores Finales), `N` (número de posiciones que tendrá el buffer) y `M` (número de posiciones que tendrá el nuevo buffer entre los Consumidores y los Consumidores Finales). Si alguno de estos parámetros se omite o no se introduce correctamente, el programa no se ejecutará.

B. Funcionamiento

La base del programa es igual a la solución original, añadiendo la lógica necesaria para la gestión de un nuevo tipo de hilos y una nueva región crítica:

1. Se añade un mutex `mutexFinal` para el acceso a la región crítica entre los consumidores y los consumidores finales, en este caso un nuevo buffer compartido al que se le reserva memoria dinámicamente.

2. Una variable de condición `condpFinal` para bloquear los hilos Consumidores hasta que no se cumpla la condición de que hay una posición libre en el buffer Final para insertar un nuevo caracter numérico.

3. Una variable de condición `condcFinal` para bloquear los hilos Consumidores Finales hasta que no se cumpla la condición de que hay alguna posición ocupada en el buffer Final para que consuma el hilo.

4. Un nuevo bucle de creación de D hilos que realizarán la función de Consumidores Finales, y su correspondiente bucle de espera para que no termine el hilo principal antes de que todos hayan finalizado, para así poder imprimir por pantalla el resultado de la suma.

Como funciones auxiliares a la de consumidorFinal, se añade una de comprobación de si un carácter es o no numérico, y funciones que aplican la lógica correspondiente a una cola FIFO para el nuevo buffer compartido.

La función de Consumidor se ha modificado para, una vez fuera de la región crítica original en la que lee el caracter del principio de la cola, comprobar si el caracter que se acaba de leer es numérico, y de ser el caso, obtiene el mutex para acceder a la región crítica del Buffer Final. Una vez obtiene el bloqueo, se comporta como el productor: comprueba si el buffer está lleno para bloquearse usando variables de condición, y una vez puede continuar inserta el elemento en el Buffer Final y envía una señal correspondiente a la variable de condición de los consumidores finales antes de soltar el mutex.

Por parte de los consumidores finales, estos se comportan como un consumidor normal que sigue el mismo procedimiento para acceder a la región crítica mediante mutexes y variables de condición. Usando la misma lógica que el consumidor inicial, comprueba que no se han

procesado ya todos los elementos. A continuación, obtiene el primer elemento de la cola y, sabiendo que es un número, obtiene su valor mediante la función `atoi` y lo suma a la variable compartida, con la seguridad de que no se producirán carreras críticas debido a que solo se modifica desde dentro de la región crítica del Buffer Final.

Cuando finaliza el programa, imprimimos el resultado de la suma y liberamos todos los recursos reservados.

III. Conclusiones

En este informe se ha abordado la mejora de la solución al problema del productor-consumidor añadiendo un paso final en el que se computaba la suma del valor de todos los caracteres numéricos en los archivos proporcionados.

Se ha añadido la funcionalidad correspondiente gestionando un nuevo buffer compartido mediante mutexes y variables de condición de una forma similar a la original, pero encadenando el trabajo de tres tipos de hilos diferentes a través de dos buffers, evitando la ocurrencia de carreras críticas y bloqueos.

Este trabajo nos demuestra la versatilidad del uso de mutexes y variables de condición, ya que nos permiten reutilizar soluciones con facilidad y efectividad.