UNIVERSITAT DE BARCELONA

Grado en Ingeniería Informática

Facultad de Matemáticas e Informática

PRÁCTICA 5.

PRODUCTOR-CONSUMDOR CON HILOS

Alumnos: Zixin Zhang - William Sotto

Asignatura: Sistemas Operativos II

Grupo B

Introducción

Durante esta práctica de laboratorio implementaremos un algoritmo productor-consumidor utilizando múltiples hilos, sincronizados mediante variables condicionales de los monitores, sobre el programa del análisis de vuelos y aeropuertos visto anteriormente. Con este paradigma conseguiremos separar, en hilos diferentes, la tarea de la lectura de datos de disco de la extracción de la información de los datos.

Preguntas

- P1. ¿Cuál es la motivación de esta recomendación? Realiza un experimento para diferentes valores de B < H y B ≥ H y comenta los resultados.

  Cuando B < H, tenemos una espera innecesaria, porque los consumidores están agotando rápidamente el búfer y no se esperan a que el productor lo llene nuevamente. Es decir, el productor no estaría produciendo al ritmo suficiente para satisfacer la demanda de los consumidores. Sin embargo con B ≥ H hay un balance eficiente. El productor puede llenar el búfer con bloques de datos antes de que los consumidores comiencen a procesarlos, evitando así situaciones en las que los consumidores esperan bloqueados por nuevos datos.

  5 = B > H = 4:

```
Origin: YAK -- Number of different destinations: 2
Origin: YKM -- Number of different destinations: 0
Origin: YUM -- Number of different destinations: 0
Tiempo para procesar el fichero: 4.968316 segundos
soliam@soliam-IdeaPad-3-15ITL6:~/Documentos/SOII/Practica5$ ./pra5 airports.csv 2000.csv 4
```

  5 = B < H = 10:

```
Origin: YAK -- Number of different destinations: 2
Origin: YKM -- Number of different destinations: 0
Origin: YUM -- Number of different destinations: 0
Tiempo para procesar el fichero: 5.145189 segundos
soliam@soliam-IdeaPad-3-15ITL6:~/Documentos/SOII/Practica5$ ./pra5 airports.csv 2000.csv 10
```

  (En el caso de H=4 tenemos los mismos resultados, pero con H=10 hay algunos resultados erróneos)

- P2. ¿Cuál es la/s sección/es crítica/s del código que deben protegerse y por qué? Justifica tu respuesta.

  Las secciones críticas de nuestro código se encuentran en la interacción del Buffer de comunicación, ya que tanto como el productor como los consumidores, pueden estar intentando acceder al buffer compartido a la vez. Por ejemplo, varias instancias del hilo consumidor podrían estar intentando acceder al búfer al mismo tiempo que el productor está colocando un nuevo bloque o también entre consumidores podrían intentar acceder al mismo

bloque al mismo tiempo. Para eso protegemos toda esta sección crítica con una variable llamada bufferMutex. Garantiza la exclusión mutua.

- P3. ¿Por qué el uso de *strcpy* es ineficiente en este caso? Justifica tu respuesta.
  En el programa la función *extract_fields_airport*, usa *strcpy* para copiar la cadena *word*. Esto puede ser inseguro si la cadena de entrada no tiene el tamaño esperado. En vez de eso se podría considerar el uso de *strncpy* para evitar desbordamientos.

```
for (int i = 0; i < N && line; i++) {
    // Colocar la línea en el buffer
    strncpy(buffer.lines[buffer.in][i], line, MAXCHAR - 1);
    buffer.lines[buffer.in][i][MAXCHAR - 1] = '\0';
```

  No lo consideramos como una sección crítica ya que en esta práctica sólo tenemos un productor, por lo tanto la lectura solo se hará por un hilo.

- P4. ¿Cómo sabrán los consumidores que el productor ya ha leído todos los bloques de datos?
  Para que los consumidores sepan que el productor ya ha leído todos los bloques de datos, hemos agregado una variable adicional en la estructura *CommunicationBuffer* que indique el estado del productor. Este, se encarga de anunciar el final del productor poniendo su variable produced_finished a 1.

```
// Estructura para el buffer de comunicación
struct CommunicationBuffer {
    char lines[B][N][MAXCHAR];
    int in;  // Índice de inserción
    int out;  // Índice de extracción
    int count;  // Contador de bloques en el buffer
    int producer_finished;
};
```

Conclusión

Gracias a la realización de esta práctica, hemos adquirido conocimientos en la implementación de modelos de productor-consumidor con hilos en C, utilizando estructuras de datos compartidas, mecanismos de sincronización y estrategias de lectura de archivos. La aplicación demuestra una gestión segura y eficaz de la concurrencia al procesar grandes conjuntos de datos como lo son los vuelos entre aeropuertos, destacando la importancia de proteger secciones críticas y optimizar el uso de memoria para lograr un rendimiento aceptable.

CONTRIBUCIÓN

Contribución al Proyecto: Hemos hecho una división y trabajado cada uno por su parte contestando las preguntas y luego de un acuerdo escoger la mejor de cada para la entrega.