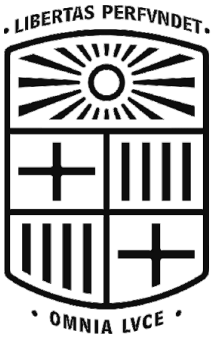


UNIVERSITAT DE BARCELONA

PRÁCTICA 6

SISTEMES OPERATIUS II

GRUP MB

AUTORES Y NIUBs:

Oriol Saguillo González NIUB 20150502

María Isabel González Sánchez NIUB 20221913

Í N D I C E

1. Preguntas de la sección 2

Para esta práctica tenemos una serie de preguntas concretas a responder:

1. **El buffer tendrá un tamaño para poder guardar B bloques, y se recomienda que B sea igual o superior al número F de hilos secundarios. ¿Por qué ha de ser así? Podéis hacer algún experimento para ver qué pasa si cogeis, por ejemplo, B = 1 amb F = 2 fils?**
   El número de bloques a leer, B, debería ser mayor o igual al número de hilos ya que al solo tener un único bloque a procesar, uno de los hilos no hará nada. Es decir, uno de los dos hilos se dormirá y el otro procesará los datos de dicho bloque. Al final solo estará trabajando uno de ellos y, por lo tanto, sería lo mismo que si tuviésemos un único hilo.

2. **Los hilos secundarios acceden a recursos (variables) compartidos entre ellos. ¿Cuáles serán las secciones críticas? ¿Qué partes del código son las que se han de proteger? ¿Hace falta proteger la lectura del fichero? ¿Hace falta proteger la extracción de datos del bloque? ¿Hace falta proteger la actualización de la variable num_flights?**
   Las secciones críticas de este código son cuando hacemos el cambio de celdas del productos al buffer y del buffer al consumidor. Está parte, además de asegurar que no haya conflicto entre los bloques del buffer, también debe proteger la escritura del contador Así, podremos saber el número de bloques que hay pendientes de lectura para el consumidor, todo ello con el mismo mutex. En la otra sección, que sería la escritura, completamos la matriz y la protegemos con otro mutex diferente, ya que esta sección es independiente de la anterior. Con respecto a la lectura del fichero, en este caso no será una sección crítica, ya que solo habrá un productor y será este el que se encarga de hacer dichas lecturas.

3. **Una forma de proceder para hacer la transferencia de la información es "copiar" la información entre las celdas. Para hacer esto hace falta usar funciones como strcpy, funciones que permiten copiar cadenas de caràcteres. El hecho de hacerlo conllevará una solución que será bastante ineficiente. ¿Podéis imaginar por qué será así?**
   Usar "strcpy" y funciones similares de las librerías de C lo que hacen es copiar carácter a carácter el contenido de las posiciones de memoria que queremos intercambiar. El problema reside ahí, que siempre estaremos ejecutando una operación innecesaria de coste O(n). Por otro lado, usando punteros, solo intercambiamos las referencias a las posiciones de memoria sin tocar su contenido en memoria. Por lo tanto, será más eficiente que "strcpy".

2. Experimentos sobre la práctica

En esta sección, estudiaremos cómo afecta el aumento de líneas o de bloques de líneas al tiempo de lectura y procesamiento de los vuelos de los archivos 2007.csv y 2008.csv, sabiendo que en nuestro código se ha implementado el paradigma del Productor-Consumidor.

Para ello, presentaremos 2 tablas comparando dichos tiempos de ejecución con diversos intentos para cada prueba, ya que la primera vez que ejecutemos el código, el fichero se quedará parcialmente en la memoria caché asociada a la máquina virtual. Como aclaración, en esta última práctica, el número de hilos secundarios será 2:

TABLA 1 (B=10)	**INTENTOS**				
	N	PRIMERO	SEGUNDO	TERCERO	CUARTO
2007.csv	1	55.307104	51.826610	55.824720	54.564736
	10	27.491601	27.257051	28.553047	28.803113
	100	22.645353	22.807203	24.583840	20.844351
	1000	22.620641	21.271809	24.057680	23.399174
	10000	22.904461	23.437946	23.404028	22.703474
2008.csv	1	53.659442	53.166406	49.149979	51.790952
	10	30.719493	30.573484	30.129055	32.480853
	100	22.921555	23.357142	23.701675	22.839102
	1000	22.071154	22.259995	22.928250	22.928250
	10000	23.238173	20.673025	21.107999	19.779355

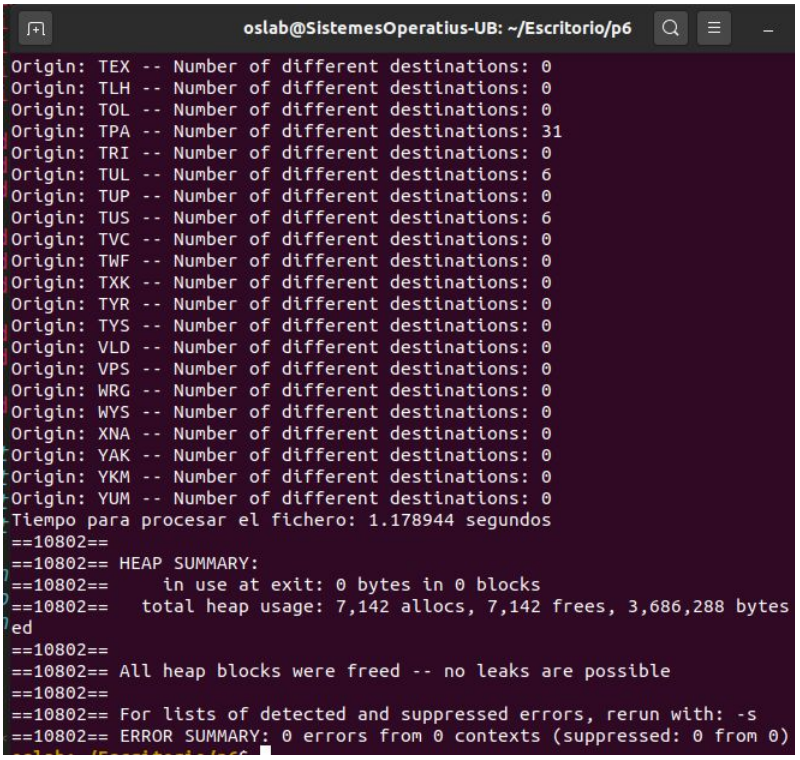
En cuanto a la 1ª tabla, compararemos el tiempo de ejecución de los archivos csv respecto al número de líneas del fichero (N) que leeremos seguidas: 1, 10, 100 o 10.000 líneas. En este caso hemos fijado el número de bloques a 10, aunque más tarde estudiaremos en profundidad este aspecto del código. Mirando detenidamente la tabla, al trabajar con 2 hilos y entre 100 y 1000 líneas por lectura del fichero, el tiempo también se reduce considerablemente al de trabajar las anteriores prácticas.

TABLA 2 (N=500)	INTENTOS				
	B	PRIMERO	SEGUNDO	TERCERO	CUARTO
2007.csv	1	19.160013	19.077600	18.949000	19.014451
	2	19.137027	19.500514	20.123754	19.222962
	5	19.099195	19.507528	19.568115	19.429998
	10	19.274170	19.029196	19.269808	19.262438
2008.csv	1	19.579881	18.180008	18.024978	19.070090
	2	18.374145	18.546097	18.464170	18.403827
	5	19.241876	18.192332	17.903310	17.995979
	10	18.584822	18.035968	18.039017	17.845583

En cuanto a la 2ª tabla, compararemos el tiempo de ejecución de los archivos csv respecto al número de bloques de líneas (B) que leeremos seguidos: 1, 2, 5 o 10 bloques. Cabe aclarar que, como anteriormente hemos visto que lo más óptimo ha sido entre 100 y 1000 líneas, hemos escogido unas 500 líneas como punto medio. Si observamos esta tabla, al trabajar con 2 hilos y a partir de 5 bloques de líneas para la lectura del fichero, el tiempo también se reduce considerablemente pero ya se estanca.

3. Problemas del código

En este código, nos encontramos con el extraño problema de que a veces el consumidor se queda dormido sin despertar durante algunas ejecuciones. Lo que podemos extraer es que, dependiendo de cuál de los hilos coja primero la llave, el otro se perderá o no. Haciendo un análisis exhaustivo, no hemos podido sacar una solución más óptima.Además, hemos comprobado con el Valgrind y el fichero pequeño (por rapidez más que nada) si pudiese ser una fallo de memoria y, en efecto, no lo es.



En sí, creemos que los conceptos de la pràctica se ven bien reflejados, a pesar de este extraño error.

Como último detalle, en la pràctica se nos pide implementar un paradigma de Productor-Consumidor para múltiples productores y consumidores. En nuestro caso, lo tenemos para múltiples consumidores, pero con los productores la única diferencia que veríamos es que, en el bloque, se deberían leer las líneas del fichero justo después de que el hilo productor se despertarse. Al tener que implementarse para un productor, lo dejamos como está pero dejamos escrita como sería su implementación.
En los casos que funciona, este sería el resultado de nuestra ejecución. Para este ejemplo hemos usado el archivo 2008.csv con 500 líneas, 2 bloques y 2 hilos: