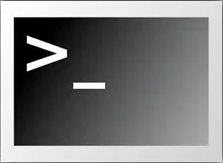
**UC3M Campus de Colmenarejo**



Sistemas Operativos



Práctica 3

 Programación del funcionamiento de un mercado de valores

(Multithread)

Lucas González de Alba | 100383228 | 100383228@alumnos.uc3m.es

Víctor Alonso López | 100383276 | 100383276@lumnos.uc3m.es

Gonzalo Fernández | 100383212 | 100383212@alumnos.uc3m.es

Ingeniería Informática

UC3M, Colmenarejo

**Índice:**

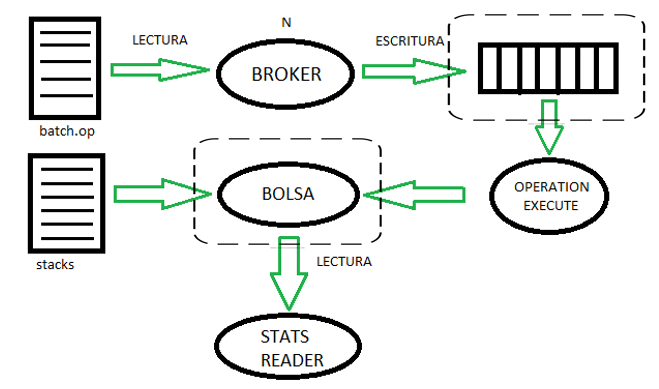
Funcionamiento del código: ............................................................pág. 1

Batería de pruebas: ........................................................................pág 4

Conclusiones: ………………………………………………………………………………pág. 7

**Descripción código:**

En esta práctica se debe simular el funcionamiento del modelo de un mercado de valores, el cual incluye Brokers, el ejecutor de las instrucciones y el encargado de consultar el estado actual de la bolsa. El objetivo es familiarizarse tanto con el uso de los hilos que proporciona el Sistema operativo como con las soluciones que se pueden implementar para gestionar la concurrencia.



Tras un primer estudio preliminar hemos llegado a las siguientes conclusiones. Es necesario asegurar la integridad de los datos compartidos y que no se deben permitir condiciones de Carrera. En baso a esto se han desarrollado dos secciones críticas. La primera se trata de la cola de operaciones donde puede escribir n Brokers cómo leer el Executer.  Cabe mencionar que los brókers en su totalidad pueden escribir de forma simultánea, aunque solo cada uno hace escrituras completes (requisito del enunciado). La segunda sección crítica corresponde a la propia bolsa, donde puede ocurrir que escriba tanto el Executer cómo leer el Stats Readers simultáneamente. La solución que vamos a llevar a cabo para la resolución de estas secciones críticas es el uso de Mutex. Adicionalmente, también se deberán gestionar las limitaciones de la cola a la hora de insertar las mencionadas operaciones.

En primer lugar, comenzaremos por explicar la implementación del bróker. Dentro de ésta función generamos un iterador para recorrer el batch operator y realizamos un lock del mutex de la cola, para poder añadir una nueva operación (Preservar la integridad de los datos, es decir, tamaño de la cola, operaciones a la espera de desencolado etc…). Antes de insertar, comprobamos que la cola no se encuentre llena puesto que en caso de que lo esté realizaremos el unlock y comenzaremos una espera hasta que la cola deje de estar vacía (Esta espera se implementa mediante la llamada al Sistema *pthread\_cond\_wait*). Una vez se haya podido insertar, realizamos el unlock del mutex de la cola y tras esto, mandamos la señal not\_empty\_queue, ya que al menos se ha encolado un elemento. El proceso expuesto se repetirá tantas veces como operación pendiente se tengan.

En segundo lugar, procede el comentario del operation\_executer donde lo primero que decidimos realizar es un bucle hasta que se pueda cerrar el programa. Tras ello nos apropiamos del mutex de la cola con un lock y comprobamos que la cola no esté vacía. Si esto se cumple significa que se tiene al menos un elemento que desencolar, aunque si la condición no se cumple es necesario realizar una espera hasta que esta no se encuentra vacía.  Lo siguiente que hacemos es solicitar el mutex de stocks para inscribir la operación en la bolsa (Segunda sección critica) De aquí en adelante el objetivo es poder empezar a desencolar mediante un bucle hasta que la cola se vacíe. Por último, generamos las señales que indican que la cola no está vacía y liberamos los mutex de stocks y la cola mediante unlock.

En cuanto al stats readers, su función principal es imprimir el estado de la bolsa. Para ello es necesario acceder al Mercado de valores, el cual va modificándose en función del operation executer.

Puesto que la ejecución del programa permite N stats reader y solo un ejecutor es necesario implementar un mutex que controle el acceso concurrente en el mercado.

No podemos permitir una lectura mientras se esté ejecutando una escritura y viceversa, así que las operaciones lock y unlock protegerán el estado de las acciones. Operation executer se apropia del mutex y procede a introducir la operación desencolada (*dequeue\_operation(newExecutor.market->stock\_operations,&op*)>=0) procesándola (*process\_operation(newExecutor.market, &op) < 0*). Si el mutex ha sido tomado por el ejecutor, ningún lector podrá imprimir estadísticas (la instrucción mutex\_lock en stats\_reader previene los accesos no permitidos pausando el hilo temporalmente).

Y si un lector está imprimiendo ocurre lo mismo, es decir, el hilo del ejecutor se detiene hasta que este se libere el mutex (unlock).

Por ultimo quedan las dos funciones que inicializan y destruyen los mecanismos de concurrencia (condiciones de mutex y mutex). Ambos métodos *init\_concurrency\_mechanisms()* y *destroy\_concurrency\_mechanisms()* se complementan creando y destruyendo los recursos en memoria. Cabe mencionar que se han incluido comprobaciones para detectar posibles fallos en la ejecución de las llamadas al Sistema.

En conjunto estas cinco funciones representan el Mercado bursátil, pero para lograr una ejecución completa es necesario crear desde la función main diversos hilos que compitan de manera ordenada y disciplinada por operar leer o escribir en la bolsa.

**Batería de pruebas:**

A continuación, se muestra la batería de pruebas a la que ha sido sometida el código. El requisito primordial de compilación ha sido superado

|  |  |  |  |
| --- | --- | --- | --- |
| Entrada | Descripción | Resultado esperado | Resultado obtenido |
| Batch con seis operaciones  Batch\_operations.txt | Compilación y ejecución correcta con un solo bróker. | Correcta compilación y carga de datos. Esperamos una carga secuencial de las operaciones del bróker en la cola y su posterior desencolado. |  |
| Batch con cero operaciones  Batch\_operations0.txt | Compilación y ejecución correcta con un broker vacio. | No se debe producir ningún encolado ni desencolado | No se produce ningún encolado |
| Dos batch con varias operaciones  Batch\_operations.txt  Batch\_operations.txt | Compilación y ejecución correcta con más de un broker | 12 encolados    12 desencolados    Concurrencia real del programa  (Interpolación de los brokers) |  |
| Un batch cuyo formato no es txt | Compilación y ejecución correcta | Funcionamiento correcto | Funciona correctamente |
| Un batch cuyo formato no es esperado por el iterador (parser) | Error del iterador | Error del iterador | Entrada del fichero:    Resultado: |
| Program principal sin operations\_executer | Verificar el funcionamiento para cuando no se introduce el operation executer | Correcta compilación y carga de datos. Esperamos una carga secuencial de las operaciones del bróker en la cola sin producirse desencolado.    Al no exitir ejecutor los mecanismos de concurrencia de stats\_reader en el acceso al mercado falla. |  |
|  | Verificar el funcionamiento para cuando no se introducen brokers | El programa comienza pero no finaliza puesto que en ningún caso se inician las transacciones bursátiles |  |
| Todos los brokers lanzan la misma acción |  | Inserción adecuado de las operaciones y procesado de las mismas. | Inserción adecuado de las operaciones y procesado de las mismas. |
| Operación invalida | Comprobar numero invalido | Entrada    Error: Numero invalido |  |
| Operación invalida | Comprobar numero desbordado | Desbordamiento | No se produce desbordamiento sino que el valor se trunca a 99999 |
| Ejecución completa con múltiples brokers y stats\_readers | Comprobar el correcto funcionamiento al introducir una alta cantidad de operaciones | Correcto funcionamiento | No se producen fallos de concurrencia puesto que los lectores no se superponen entre si y no permitimos lectura mientras se produzca escritura en el mercado de valores. Además podemos observar como existe concurrencia real entre los múltiples bróker, los cuales **compiten** entre sí por los recursos solapándose de manera **controlada.** |

**Conclusión:**

En este apartado trataremos los errores que han surgido en el desarrollo de la práctica y nuestra opinión al respecto. En cuanto a los problemas encontrados, destacan la complejidad de cerrar el programa dado que entra en juego apropiación indebida de mutex y variables compartidas, sincronización de hilos (*pthread\_join ()*), y que dado que uno de los brokers resultó tedioso su posterior eliminación. Finalmente hemos logrado modificar el programa integrando una concurrencia real entre brokers y su acceso a la cola, así como el executer y stats\_reader. En cuanto a la realización de esta práctica podemos concluir que nos ha ayudado bastante, entre otras cosas, a poder comprender todo lo referente a la utilización del mutex, hilos, condiciones de Carrera y congruencia. También mencionar todo lo referente al uso de los lock y unlock cuya presencia en la práctica es bastante relevante, así como las señales de tipo pthread\_mutex\_cond.

No obstante, en general, nuestra sensación ha sido que hemos dispuesto de poco tiempo para la realización de la práctica, debido a la ingesta cantidad de trabajos y exámenes en las fechas en las que nos encontramos. Cada uno de los miembros del equipo coincidimos en que con más tiempo y un ambiente un tanto más sosegado hubiéramos afrontado la práctica de una manera más reconfortante y académica.

Con todo y con ello estamos orgullosos de nuestro trabajo y animamos al equipo de dirección a proseguir una enseñanza práctica basada en el aprendizaje escalonado y la guía experta.