CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS (CUCEI)

Departamento de ciencias computacionales

Seminario de solución de problemas de uso, adaptación, explotación de sistemas operativos

Violeta del Rocío Becerra Velázquez

Jose Pedro Reyes Alvarez

222790897

Ingeniería Informática (INNI)

D02

2.3 Exclusión mutua

31 de marzo del 2025

Índice

Soluciones para garantizar la exclusión mutua

La exclusión mutua es un requisito fundamental en sistemas concurrentes, ya que evita que múltiples procesos accedan simultáneamente a recursos compartidos, lo que podría generar inconsistencias en los datos o fallos en la ejecución. Para lograrla, existen diversas soluciones, que pueden clasificarse en métodos basados en hardware, software y mecanismos proporcionados por el sistema operativo. A continuación, se describen algunas de las soluciones más utilizadas.

1. Monitores

Los monitores son una abstracción de alto nivel utilizada para la sincronización en la programación concurrente. Un monitor encapsula variables compartidas, procedimientos y mecanismos de sincronización en una única estructura, garantizando que solo un proceso pueda ejecutar los procedimientos internos a la vez.

Características principales:

- Ofrecen una forma estructurada de controlar el acceso a los recursos compartidos.
- Utilizan variables de condición para suspender procesos cuando un recurso no está disponible.
- Son implementados por algunos lenguajes de programación como Java (con synchronized) y C# (con lock).

2. Semáforos

Los semáforos son estructuras de control utilizadas para restringir el acceso a un recurso compartido mediante contadores y operaciones atómicas como wait() y signal(). Fueron introducidos por Edsger Dijkstra en 1965 y pueden ser de dos tipos:

- Semáforos binarios (equivalentes a un mutex), que solo permiten dos estados: 0 (bloqueado) y 1 (disponible).
- Semáforos contadores, que permiten gestionar múltiples instancias de un recurso.

Ejemplo de operaciones básicas:

- wait(S): Si el valor del semáforo S es mayor que 0, se decrementa y el proceso continúa. Si es 0, el proceso se bloquea.
- signal(S): Incrementa el valor del semáforo y permite que otros procesos continúen.

3. Mutex (Mutual Exclusion Locks)

Un mutex es un mecanismo de sincronización que garantiza que solo un hilo o proceso pueda acceder a una sección crítica a la vez. A diferencia de los semáforos, un mutex solo puede ser desbloqueado por el mismo proceso que lo bloqueó.

Características:

- Solo permite dos estados: bloqueado o desbloqueado.
- Evita problemas como la inversión de prioridad si se usa junto con políticas adecuadas.
- Su uso es común en sistemas operativos y bibliotecas como POSIX Threads.

4. Algoritmo de Dekker

El algoritmo de Dekker fue uno de los primeros en resolver el problema de la exclusión mutua mediante programación sin hardware especial. Permite que dos procesos se alternen el acceso a una sección crítica sin interferencias.

Funcionamiento:

- Cada proceso tiene una variable booleana (quiereEntrar) que indica si desea acceder a la sección crítica.
- Una variable turno define qué proceso tiene prioridad en caso de que ambos quieran acceder al recurso.

5. Algoritmo de Peterson

Este algoritmo es una mejora del de Dekker y ofrece una solución eficiente para la exclusión mutua entre dos procesos sin necesidad de soporte especial del hardware.

Funcionamiento:

- Cada proceso indica que quiere entrar en la sección crítica.
- Se usa una variable turno para decidir quién tiene prioridad.
- Se evita el problema del "interbloqueo" al permitir que los procesos cedan el turno cuando ambos quieren entrar al mismo tiempo.

6. Paso de Mensajes

El paso de mensajes es una técnica en la que los procesos se comunican enviándose datos en lugar de compartir memoria. Se usa principalmente en sistemas distribuidos y en la programación con concurrencia distribuida.

Características:

- Puede ser síncrono (el emisor espera la respuesta antes de continuar) o asíncrono (el emisor sigue ejecutando sin esperar).
- Se implementa mediante mecanismos como colas de mensajes, sockets o memoria compartida.
- Evita la necesidad de mecanismos tradicionales de exclusión mutua, ya que no hay memoria compartida.

Ejemplos de cada solución para la exclusión mutua

1. Monitores

Ejemplo:

En un sistema bancario, cuando un usuario intenta retirar dinero de su cuenta a través de un cajero automático, el sistema debe asegurarse de que nadie más pueda modificar el saldo de esa cuenta hasta que la transacción se complete. El sistema bloquea temporalmente el acceso a la cuenta mientras se procesa la operación.

2. Semáforos

Ejemplo:

En una empresa con varias impresoras compartidas, se usa un semáforo para gestionar el acceso de los empleados. Cuando alguien envía un documento a imprimir, el sistema verifica si hay una impresora disponible. Si no la hay, el usuario debe esperar hasta que una impresora esté libre.

3. Mutex (Mutual Exclusion Locks)

Ejemplo:

Un sistema de edición de documentos colaborativo como Google Docs usa mecanismos de exclusión mutua para evitar que dos personas editen exactamente la misma parte del documento al mismo tiempo, evitando sobrescribir cambios.

4. Algoritmo de Dekker

Ejemplo:

Imagina dos cajeros de supermercado que necesitan acceso a la misma caja registradora. Siguen un protocolo en el que verifican si el otro cajero está usando la caja antes de proceder. Si ambos quieren acceder al mismo tiempo, uno de ellos cede el turno y espera su oportunidad.

5. Algoritmo de Peterson

Ejemplo:

En un sistema de turnos para el servicio de atención al cliente, si dos empleados desean atender al siguiente cliente al mismo tiempo, se establece un mecanismo para decidir cuál de ellos tiene prioridad. Si ambos están disponibles al mismo tiempo, el sistema define un orden basado en la última persona que atendió un cliente.

6. Paso de Mensajes

Ejemplo:

Un chat en línea donde los mensajes enviados por un usuario deben ser entregados a otro usuario sin que se pierdan ni se dupliquen. El servidor de mensajería gestiona la entrega y recepción de mensajes asegurando que solo una copia llegue al destinatario.

Síntesis sobre la exclusión mutua

¿Qué es la exclusión mutua?

La exclusión mutua es un concepto fundamental en sistemas operativos y programación concurrente que impide que varios procesos accedan simultáneamente a un mismo recurso compartido. Esto es crucial para evitar errores y garantizar la integridad de los datos.

Diferentes soluciones para la exclusión mutua

Para evitar problemas de concurrencia, se han desarrollado diferentes soluciones, entre ellas:

Monitores: Permiten gestionar el acceso a recursos compartidos mediante estructuras que encapsulan variables y métodos sincronizados.

Semáforos: Utilizan contadores para controlar el acceso de múltiples procesos a un recurso, permitiendo o bloqueando su uso.

Mutex: Un mecanismo de bloqueo que garantiza que solo un proceso pueda acceder a un recurso a la vez.

Algoritmo de Dekker: Solución de software para la exclusión mutua entre dos procesos sin necesidad de soporte de hardware.

Algoritmo de Peterson: Método eficiente para coordinar el acceso a recursos entre dos procesos mediante banderas y turnos.

Paso de Mensajes: Técnica utilizada en sistemas distribuidos donde los procesos se comunican enviando mensajes en lugar de compartir memoria.

Ejemplos prácticos

Monitores

Bloqueo de acceso a una cuenta bancaria mientras se realiza una transacción.

Semáforos

Control de acceso a impresoras compartidas en una oficina.

Mutex

Edición colaborativa de documentos donde solo una persona puede modificar una sección a la vez.

Algoritmo de Dekker

Cajeros de supermercado turnándose para usar una misma caja registradora.

Algoritmo de Peterson

Sistema de turnos en atención al cliente.

Paso de Mensajes

Comunicación entre usuarios en un chat en línea.

Conclusión

En esta actividad, aprendí la importancia de la exclusión mutua para evitar problemas como condiciones de carrera y corrupción de datos en sistemas concurrentes. Existen diversas soluciones, como **monitores, mutex y semáforos**, que permiten controlar el acceso a recursos compartidos, mientras que **Dekker, Peterson y el paso de mensajes** ofrecen alternativas sin memoria compartida.

Comprendí cómo estos mecanismos se aplican en situaciones reales, como sistemas bancarios, impresoras compartidas y chats en línea. En conclusión, la exclusión mutua es clave para el desarrollo de sistemas seguros y eficientes.

Fuentes

Capítulo 5: Stallings, W. (2010). Capítulo 5: Concurrencia. Exclusión mutua y sincronización. En *Sistemas operativos*. Pearson Educación.

Capítulo 6: Stallings, W. (2010). Capítulo 6: Concurrencia. Interbloqueo e inanición. En *Sistemas operativos*. Pearson Educación.

Universidad Autónoma de Puebla. Facultad de Ciencias de la Computación. (s.f.). *Unidad 4: Exclusión mutua, región crítica y sincronización con hilos*. Recuperado de https://www.cs.buap.mx/~hilario_sm/slide/pcp2016/unidad%204%20pcp-2017.pdf

González, F. J. (2003). Concurrencia: Exclusión mutua y sincronización. *Escuela Universitaria de Informática (Segovia), Universidad de Valladolid*. Recuperado de https://www.infor.uva.es/~fjgonzalez/apuntes/Tema6.pdf

Castellanos, L. (03/02/2015). Solución de Peterson. *Sistemas Operativos*. Recuperado de https://lcsistemasoperativos.wordpress.com/tag/solucion-peterson/