Tarea 5. Filósofos, productor, lectores

Actividad 9

Nombre: Flores Estrada Abraham Miguel Angel

Codigo: 217443356

Carrera: INCO

Profesor: Becerra Velazquez, Violeta Del Rocio

Sección: D01

Materia: Seminario De Solución De Problemas De Sistemas Operativos

Departamento de ciencias computacionales

Fecha de entrega: 26/03/23

Centro Universitario de Ciencias Exactas e Ingeniería

Índice

Contenido

La cena de los filósofos

El problema de la cena de los filósofos es un problema clásico en la informática y la filosofía, que plantea cómo varios filósofos pueden compartir unos palillos para comer sin que se produzca un bloqueo o interbloqueo del sistema.El problema se presenta de la siguiente manera: se tienen cinco filósofos sentados alrededor de una mesa, y entre cada par de ellos hay un palillo. Los filósofos pasan su tiempo alternando entre dos estados: pensando y comiendo. Para comer, un filósofo necesita dos palillos, uno a su izquierda y otro a su derecha. Cuando un filósofo no tiene los dos palillos necesarios para comer, se queda esperando.El problema radica en que, si todos los filósofos intentan tomar los palillos al mismo tiempo, es posible que se produzca un bloqueo, es decir, que los filósofos no puedan avanzar y se queden esperando indefinidamente.

Existen varias soluciones al problema de la cena de los filósofos, siendo la más común la solución de Dijkstra, que consiste en asignar un orden de adquisición de los palillos a cada filósofo. Por ejemplo, el filósofo 1 siempre tomará primero el palillo de su izquierda y luego el de su derecha, mientras que el filósofo 2 hará lo contrario, tomando primero el palillo de su derecha y luego el de su izquierda. De esta manera, se garantiza que nunca todos los filósofos intenten tomar los palillos al mismo tiempo, evitando el bloqueo. Sin embargo, existen otras soluciones más complejas que también abordan este problema.

Productor-Consumidor

El problema del productor-consumidor es un problema clásico de concurrencia que se encuentra comúnmente en sistemas operativos y programación de sistemas. El problema se presenta cuando hay un conjunto de procesos productores y consumidores que comparten un recurso común, como una cola o un buffer. El objetivo es que los productores produzcan datos y los coloquen en el recurso compartido, mientras que los consumidores tomen los datos del recurso compartido y los procesen o utilicen. El problema radica en que, si se permite que los productores y los consumidores accedan al recurso compartido al mismo tiempo, pueden ocurrir problemas de sincronización, como el acceso simultáneo al mismo dato o la extracción de datos no válidos. Para resolver este problema, se pueden implementar diversas soluciones. Una solución común es

utilizar semáforos o monitores para sincronizar el acceso al recurso compartido. En este enfoque, se pueden utilizar contadores de semáforos para controlar el acceso de los productores y consumidores al recurso compartido. Por ejemplo, cuando un productor agrega un dato al recurso compartido, debe incrementar el semáforo para indicar que un nuevo dato está disponible. Cuando un consumidor toma un dato del recurso compartido, debe decrementar el semáforo para indicar que hay un espacio disponible para un nuevo dato.

Lectores-Escritores

El problema de los lectores-escritores es un problema clásico de concurrencia en el que varios procesos desean acceder a una base de datos o recurso compartido, pero algunos de ellos sólo desean leer, mientras que otros desean escribir y modificar el recurso. El problema radica en que, si varios procesos lectores y escritores acceden al recurso compartido al mismo tiempo, pueden ocurrir problemas de sincronización, como la lectura y escritura simultánea de datos o la escritura de datos inválidos.

La solución a este problema consiste en proporcionar una sincronización adecuada que permita que los procesos lectores y escritores accedan al recurso compartido de manera segura y eficiente. Hay varias soluciones comunes para el problema de los lectores-escritores, incluyendo:

- Solución de prioridad de escritura: se da prioridad a los procesos escritores. Es decir, cuando un proceso escritor solicita el acceso al recurso, se le concede el acceso de inmediato, y mientras hay un proceso escritor en la cola, no se permiten nuevos procesos lectores. De esta manera, se evita que los procesos lectores lean datos incorrectos.
- Solución de no prioridad: se permite que tanto los procesos escritores como los lectores accedan al recurso compartido. Se utiliza un semáforo o monitor para sincronizar el acceso, y se utiliza un contador para contar el número de procesos lectores que acceden al recurso. Si hay un proceso escritor en la cola, se espera a que todos los procesos lectores terminen antes de permitir que el escritor acceda al recurso.
- Solución de lectura preferente: se da prioridad a los procesos lectores. En este enfoque, se permite que los procesos lectores accedan al recurso compartido de manera concurrente, pero se utiliza un semáforo o monitor para asegurarse de que no haya procesos escritores presentes. Cuando un proceso escritor solicita acceso, se espera a que todos los procesos lectores terminen antes de permitir que el escritor acceda al recurso.

En general, la solución del problema de los lectores-escritores depende del contexto específico en el que se está utilizando y de los requisitos de sincronización y rendimiento. La elección de la solución correcta dependerá de las características específicas del sistema y de los requisitos de sincronización y rendimiento.

¿En qué consiste el problema de la concurrencia?

El problema de la concurrencia se refiere a las situaciones en las que varios procesos o hilos de ejecución intentan acceder y modificar los mismos recursos

compartidos al mismo tiempo. En sistemas donde se utilizan múltiples procesos o hilos de ejecución, la concurrencia puede causar problemas de sincronización y errores impredecibles que pueden ser difíciles de identificar y resolver.

La concurrencia puede dar lugar a situaciones en las que un proceso intenta acceder a un recurso que está siendo utilizado o modificado por otro proceso, lo que puede provocar inconsistencias en los datos y resultados impredecibles. Además, la concurrencia puede aumentar la complejidad del código y dificultar la implementación, el mantenimiento y la depuración del sistema.

¿Cuáles son los procesos concurrentes cooperantes?

Los procesos concurrentes cooperantes son procesos o hilos de ejecución que trabajan juntos para lograr un objetivo común, coordinando sus acciones y compartiendo recursos. Estos procesos interactúan entre sí y con el sistema operativo para realizar una tarea compleja de manera eficiente y efectiva. En general, los procesos concurrentes cooperantes se pueden dividir en dos categorías principales: los productores-consumidores y los lectores-escritores. Los productores-consumidores son procesos que producen y consumen datos de un recurso compartido. Los procesos productores generan datos y los almacenan en un buffer compartido, mientras que los procesos consumidores recuperan los datos del buffer y los utilizan para realizar alguna tarea.

¿En qué consiste la Exclusión mutua?

La exclusión mutua se refiere a la propiedad que garantiza que un recurso compartido no pueda ser utilizado por más de un proceso o hilo de ejecución al mismo tiempo. En otras palabras, la exclusión mutua asegura que un proceso o hilo de ejecución tenga acceso exclusivo a un recurso compartido mientras lo esté utilizando. La exclusión mutua es importante porque, si varios procesos o hilos de ejecución intentan acceder al mismo recurso compartido al mismo tiempo, pueden producirse errores de sincronización, resultados inconsistentes y conflictos en los datos. Para lograr la exclusión mutua, los sistemas operativos y los lenguajes de programación ofrecen mecanismos de sincronización y coordinación, como los semáforos, los mutexes y los monitores.

Interbloqueo

El interbloqueo, también conocido como deadlock, es una situación en la que dos o más procesos o hilos de ejecución se bloquean mutuamente, impidiendo que cualquier proceso o hilo de ejecución pueda continuar. El interbloqueo se produce cuando cada proceso o hilo de ejecución está esperando a que otro proceso libere un recurso que necesita para continuar, y al mismo tiempo, está reteniendo un recurso que otro proceso necesita para continuar.

El interbloqueo puede ser causado por problemas de sincronización y coordinación en la programación concurrente, donde varios procesos o hilos de ejecución están intentando acceder a los mismos recursos compartidos al mismo tiempo.

Inanición

La inanición, también conocida como starvation, es una situación en la que

un proceso o hilo de ejecución se ve impedido de acceder a los recursos que necesita debido a la competencia con otros procesos o hilos de ejecución. En otras palabras, un proceso o hilo de ejecución se queda "hambriento" de recursos y no puede realizar su trabajo debido a la falta de acceso a los recursos necesarios.

La inanición puede ocurrir en sistemas concurrentes cuando los recursos compartidos son utilizados de forma intensiva y hay una alta competencia por su acceso. Si un proceso o hilo de ejecución se ve impedido de acceder a los recursos que necesita durante un tiempo prolongado, puede entrar en un estado de inanición, lo que puede afectar negativamente al rendimiento y la eficiencia del sistema.

Excesiva Cortesía

Se produce cuando varios procesos o hilos de ejecución intentan acceder simultáneamente a un recurso compartido y es necesario garantizar que sólo uno de ellos pueda acceder a él en un momento determinado. El objetivo es evitar que dos o más procesos accedan al mismo recurso al mismo tiempo, lo que podría llevar a situaciones como lecturas inconsistentes, corrupción de datos o bloqueos del sistema.

Hilos

Un hilo o thread es una secuencia de instrucciones que puede ser ejecutada por un sistema operativo como si fuera un proceso independiente, pero comparte recursos con otros hilos dentro del mismo proceso. En otras palabras, un hilo es una unidad básica de ejecución dentro de un proceso.

Cada hilo tiene su propio contador de programa, pila de ejecución y conjunto de registros, pero comparte el espacio de memoria, los archivos abiertos y otros recursos del proceso principal. Los hilos permiten a los desarrolladores diseñar programas concurrentes y paralelos en los que varias tareas se ejecutan simultáneamente dentro de un mismo proceso.

Semaforos

Los semáforos son un mecanismo de sincronización de procesos ampliamente utilizado en los sistemas operativos para evitar problemas de exclusión mutua entre procesos o hilos de ejecución. Los semáforos son variables especiales que se utilizan para controlar el acceso a los recursos compartidos, como archivos, memoria o dispositivos.

Los semáforos funcionan mediante la gestión de un valor entero que se utiliza como contador. Cuando un proceso quiere acceder a un recurso compartido, debe solicitar permiso al semáforo. Si el valor del semáforo es mayor que cero, el proceso puede acceder al recurso compartido y el valor del semáforo se decrementa. Si el valor del semáforo es cero, el proceso se bloquea hasta que el semáforo tenga un valor mayor que cero.

¿Qué es lo que mejora el tener más de un núcleo?

Tener más de un núcleo en un procesador mejora el rendimiento y la capacidad de procesamiento de un sistema. Los núcleos adicionales permiten que el procesador pueda manejar múltiples tareas simultáneamente y de manera más eficiente, ya que cada núcleo puede ejecutar una tarea independiente de las otras.

Esto se conoce como paralelismo, que es la capacidad de realizar múltiples tareas simultáneamente.

En un sistema con un solo núcleo, cuando una tarea está en ejecución, todas las demás tareas tienen que esperar en la cola hasta que la tarea en ejecución se complete. Esto puede llevar a una ralentización del sistema y a una disminución del rendimiento

Conclusión

Con esta actividad se conoció más a fondo varios temas que son importantes para los programadores, ya que nos permiten crear sistemas operativos y aplicaciones que sean estables, eficientes y capaces de manejar múltiples procesos y tareas simultáneamente. Por lo tanto, es esencial tener un conocimiento profundo de estos temas para poder crear sistemas operativos y aplicaciones de alta calidad. Asi mismo, problema del Productor-Consumidor es importante en el campo de la programación paralela y la concurrencia, y su aplicación se encuentra en muchos sistemas informáticos que necesitan coordinar la producción y el consumo de recursos compartidos.

Bibliografía

Mchoes, F. (s.f.). *Sistemas operativos.* Obtenido de https://dokumen.tips/documents/sistemas-operativos-flynn-mchoes.html?page=1

Operating Systems. (s.f.). Obtenido de http://160592857366.free.fr/joe/ebooks/ShareData/Understanding%20Operating%20Systems%206e%20By%20Ann%20McIver%20McHoes%20and%20Ida%20M.%20Flynn.pdf

S, A. (s.f.). *Sistemas operativos modernos.* Obtenido de https://drive.google.com/file/d/1AjlX7yK7AiTJvBAzQBQ_7lUElbpzbV8Y/view

Sistemas operativos y su gestion. (s.f.). Obtenido de https://www.yumpu.com/es/document/read/14441086/gestion-de-los-recursos-de-un-sistema-operativo-mcgraw-hill

Universidad tecnologica de panama . (s.f.). *Sistemas operativos.* Obtenido de https://ridda2.utp.ac.pa/bitstream/handle/123456789/5074/folleto_sistemas_operativos.pdf?sequence=3&isAllowed=y

WikiHow. (s.f.). *Cómo crear un archivo por lotes.* Obtenido de https://es.wikihow.com/crear-un-archivo-por-lotes