REPÚBLICA BOLIVARIANA DE VENEZUELA

MINISTERIO DEL PODER POPULAR PARA LA EDUCACIÓN UNIVERSITARIA

UNIVERSIDAD NACIONAL EXPERIMENTAL MARÍTIMA DEL CARIBE

SISTEMAS OPERATIVOS I

SOP-I 513

SECCIÓN: A

Unidad II

ADMINISTRACION DE PROCESOS

Integrantes Equipo 5:

Bolivar, José

Mendoza, Guillermo

Partidas, Nayroska

Silva, Anyel

Profesor:

Lic. Padilla P, Juan Vicente

Catia la Mar, noviembre 2024

INDICE.

p.p.

Concurrencia

Se refiere a la existencia de múltiples eventos, circunstancias o situaciones que ocurren al mismo tiempo. En el contexto de programación y computación, “concurrencia” se refiere a la capacidad de un sistema para ejecutar múltiples tareas de manera simultánea o intercalada, mejorando así la eficiencia y la velocidad del procesamiento.

Es como organizar varias cosas a la vez para que todo funcione sin problemas, como cuando una persona puede realizar varias tareas al mismo tiempo sin perder el hilo de ninguna. La concurrencia es un concepto fascinante y fundamental en muchos campos, especialmente en la informática y la programación.

Concurrencia en Programación

La concurrencia en programación se refiere a la capacidad de un sistema para manejar múltiples tareas al mismo tiempo. Esto se puede lograr de varias maneras, como con hilos (threads), procesos y corrutinas. Estas técnicas permiten que un programa realice múltiples operaciones simultáneamente, mejorando la eficiencia y el rendimiento.

Conceptos Clave

- Hilos (Threads): Son las unidades más pequeñas de procesamiento dentro de un proceso. Un programa puede tener múltiples hilos que se ejecutan simultáneamente.
- Procesos: Son instancias de programas en ejecución. Un sistema operativo puede manejar varios procesos a la vez.
- Corrutinas: Son una estructura más ligera que los hilos, usadas para ejecutar funciones de manera concurrente sin la sobrecarga de la gestión de hilos.

Ejemplo

Imagina un programa que descarga archivos de internet mientras procesa datos. Sin concurrencia, el programa tendría que esperar a que se complete la descarga para comenzar a procesar los datos. Con concurrencia, la descarga y el procesamiento pueden ocurrir simultáneamente, mejorando la eficiencia.

Ventajas

- Eficiencia: Permite un uso más eficiente de los recursos del sistema.

- Velocidad: Reduce el tiempo total de ejecución de tareas.

- Reactividad: Mejora la capacidad de respuesta de los programas.

Desafíos

- Sincronización: Asegurar que los hilos y procesos no interfieran entre sí puede ser complicado.

- Deadlocks: Ocurren cuando dos o más hilos quedan atrapados esperando recursos que están bloqueados por otros hilos.

- Condiciones de Carrera: Situaciones donde el resultado del programa depende del orden de ejecución de los hilos.

Aplicaciones

- Sistemas Operativos: Manejan múltiples aplicaciones simultáneamente.

- Servidores Web: Atender múltiples solicitudes de usuarios a la vez.

- Juegos: Ejecutar lógica de juego y renderizado gráfico simultáneamente.

Interbloqueo (Deadlock):

El interbloqueo (deadlock) es una situación en la programación concurrente donde dos o más procesos o hilos se bloquean mutuamente esperando que el otro libere un recurso, lo que lleva a una condición en la que ninguno de los procesos puede continuar. Este es un problema crítico en sistemas que requieren sincronización y acceso compartido a recursos.

Condiciones Necesarias para el Interbloqueo

Para que ocurra un interbloqueo, deben cumplirse las siguientes cuatro condiciones simultáneamente:

1. Exclusión Mutua: Al menos un recurso debe estar en modo no compartido, es decir, solo un proceso puede usar el recurso a la vez.

2. Retención y Espera: Un proceso que ya tiene asignado un recurso puede solicitar recursos adicionales que están siendo mantenidos por otros procesos.

3. No-preempción: Los recursos no pueden ser forzados a ser liberados por los procesos que los poseen, sino que deben ser liberados voluntariamente por el proceso que los tiene.

4. Espera Circular: Debe existir un conjunto de dos o más procesos esperando recursos de forma circular, donde cada proceso está esperando un recurso que otro proceso en el conjunto posee.

Ejemplo Clásico

Un ejemplo clásico de interbloqueo es el problema de los filósofos comensales, donde cada filósofo necesita dos tenedores para comer, pero solo pueden tomar un tenedor a la vez. Si cada filósofo toma un tenedor y espera por el segundo, todos se quedan bloqueados, incapaces de continuar.

Estrategias para Evitar Interbloqueos

1. Prevención de Interbloqueos: Modificar el sistema para garantizar que una de las condiciones necesarias para el interbloqueo no pueda ocurrir.

- Eliminación de la Exclusión Mutua: Hacer que los recursos sean compartibles.

- Evitar la Retención la y Espera: Requerir que los procesos soliciten todos los recursos necesarios al inicio.

- Preempción: Permitir que un proceso quite recursos de otros procesos.

- Evitar la Espera Circular: Ordenar los recursos y requerir que los procesos soliciten recursos en un orden predeterminado.

2. Detección y Recuperación: Permitir que el sistema caiga en interbloqueo, detectar cuándo ocurre y tomar acciones para recuperarse.

- Detección: Usar algoritmos para detectar ciclos en el grafo de asignación de recursos.

- Recuperación: Forzar a uno o más procesos a liberar sus recursos y reiniciarlos.

3. Evitación de Interbloqueos: Usar información adicional sobre qué recursos se necesitarán en el futuro para decidir si conceder o no recursos a un proceso.

Condiciones necesarias para que se produzca el deadlock.

El deadlock (o interbloqueo) es una situación en sistemas operativos donde dos o más procesos no pueden continuar su ejecución porque cada uno está esperando que el otro libere un recurso. Para que ocurra un deadlock, deben cumplirse las siguientes cuatro condiciones simultáneamente:

- Mutual Exclusión (Exclusión Mutua): Al menos un recurso debe estar en un estado que no pueda ser compartido por más de un proceso a la vez. Esto significa que un recurso debe estar asignado exclusivamente a un solo proceso.
- Hold and Wait (Sostener y Esperar): Un proceso que ya tiene al menos un recurso está esperando para obtener recursos adicionales que están siendo retenidos por otros procesos.
- No Preemption (No Expulsión): Los recursos no pueden ser forzadamente retirados de los procesos que los están manteniendo. Un recurso sólo puede ser liberado voluntariamente por el proceso que lo tiene.
- Circular Wait (Espera Circular): Existe un conjunto de procesos {P1, P2, ..., Pn} tal que P1 está esperando un recurso que está siendo sostenido por P2, P2 está esperando un recurso que está siendo sostenido por P3, y así sucesivamente, hasta que el último proceso Pn está esperando un recurso que está siendo sostenido por P1.

La combinación de estas condiciones crea una situación en la que ninguno de los procesos involucrados puede proceder, resultando en un deadlock.

Condiciones necesarias para prevenir el deadlock.

Prevenir el deadlock es crucial para garantizar que los sistemas operativos y los programas funcionen de manera eficiente y sin bloqueos.

- Protocolo de Prevención de Exclusión Mutua: Dado que la exclusión mutua es una condición necesaria para el deadlock, permitir que algunos recursos sean compartidos por múltiples procesos puede ayudar a prevenirlo. Sin embargo, esto no siempre es posible, especialmente para recursos que no pueden ser compartidos (e. g., impresoras).

- Protocolo de Prevención de Retención y Espera: Asegurarse de que un proceso solicite todos los recursos que necesita al inicio, y no durante su ejecución, previene la situación en la que un proceso que ya posee recursos solicita más recursos y queda bloqueado.

- Protocolo de No Expulsión: Permitir la expulsión de recursos en ciertos casos puede prevenir el deadlock. Por ejemplo, si un proceso que tiene un recurso es preemptado (interrumpido) y el recurso es liberado, otros procesos pueden utilizarlo.

- Protocolo de Prevención de Espera Circular: Imponer una ordenación lineal de todos los tipos de recursos y exigir que todos los procesos soliciten los recursos en ese orden. Esto evita la formación de ciclos que conducen al deadlock.

Además de estas estrategias, los sistemas operativos suelen implementar técnicas adicionales para manejar y prevenir deadlocks:

- Detección y Recuperación: Los sistemas pueden detectar deadlocks y tomar acciones correctivas, como abortar uno de los procesos involucrados o forzar la liberación de algunos recursos.

- Evitación de Deadlock: Utilizando algoritmos como el Algoritmo del Banquero de Dijkstra, el sistema puede decidir si conceder o no una solicitud de recursos basándose en si la concesión podría llevar al deadlock.

Implementar estas técnicas y condiciones ayuda a prevenir la aparición de deadlocks y asegura un funcionamiento más robusto y eficiente del sistema operativo y las aplicaciones que dependen de él.

Algoritmo del Banquero.

El Algoritmo del Banquero, propuesto por Edsger Dijkstra, es un algoritmo de evitación de deadlock utilizado en sistemas operativos para gestionar la asignación de recursos y evitar situaciones de interbloqueo. El nombre se deriva de una analogía con un banquero que presta recursos (dinero) a sus clientes (procesos) de tal manera que no se arruine (evite el deadlock).

Principios Básicos

- Solicitudes Seguras: Antes de asignar recursos solicitados por un proceso, el sistema verifica si conceder la solicitud mantendría el sistema en un estado seguro. Un estado es seguro si es posible que todos los procesos completen su ejecución sin caer en deadlock.
- Estados Seguros e Inseguros: Si una solicitud de recursos deja el sistema en un estado seguro, se concede. Si deja el sistema en un estado inseguro, la solicitud se pospone hasta que pueda ser atendida sin riesgo de deadlock.

Funcionamiento del Algoritmo

- Vectores de Máximos y Asignación: Cada proceso declara el número máximo de recursos que puede necesitar. El sistema mantiene un registro de los recursos actualmente asignados a cada proceso y los recursos disponibles.
- Chequeo de Seguridad: Antes de asignar recursos solicitados, el sistema simula la asignación y verifica si existe una secuencia de procesos en la que todos los procesos puedan completar su ejecución sin quedar bloqueados. Si tal secuencia existe, el estado es seguro.

Secuencialidad

Los archivos secuenciales son un tipo de archivo en los que la información puede leerse y escribirse empezando desde el principio del archivo. Debemos tomar en consideración algunas características que deben tener los archivos secuenciales:

1. La escritura de nuevos datos siempre se hace al final del archivo.

2. Para leer una zona concreta del archivo hay que avanzar siempre, si la zona está antes de la zona actual de lectura, será necesario "rebobinar" el archivo.

3. Los ficheros sólo se pueden abrir para lectura o para escritura, nunca de los dos modos a la vez.

Archivos Secuenciales

Se refiere al procesamiento de los registros, no importa el orden en que se haga, para eso los registros están organizados en forma de una lista y recuperarlos y procesarlos uno por uno de principio a fin.

Rendimientos de los archivos Secuenciales; dependiendo del dispositivo de almacenamiento utilizado el archivo se puede mostrar el usuario como si fuera un sistema secuencial. Al finalizar un archivo secuencial se denota con una marca de fin de archivo. (End end-of-file) Seriabilidad: Cuando hablamos de seriabilidad nos referimos a la capacidad de reproducir un producto x en número limitado de veces.

Seriabilidad

La serialización se considera el criterio fundamental para asegurar la corrección en el control de la concurrencia. Un grupo de transacciones entrelazadas es considerado correcto si es serializable, lo que significa que produce el mismo resultado que se obtendría al ejecutar esas transacciones en serie. Cuando se tiene un conjunto de transacciones entrelazadas, cualquier ejecución de estas se denomina calendarización.

Teoría de la seriabilidad.

La teoría de Seriabilidad es una herramienta matemática que permite probar si un sincronizador trabaja o no correctamente. Desde el punto de vista de la teoría de Seriabilidad, una transacción es una representación de una ejecución de operaciones de lectura y escritura y que indica el orden en el que se deben ejecutar estas operaciones. Además, la transacción contiene un Commit o un Abort como la última operación para indicar si la ejecución que representa terminó con éxito o no

Diciplina de planificación.

La planificación de procesos es un aspecto fundamental de un sistema operativo que juega un papel crucial en la determinación del orden y el tiempo de ejecución de múltiples procesos. Al asignar recursos de manera efectiva y gestionar la ejecución de procesos, la planificación de procesos garantiza que el sistema informático opere de manera eficiente y óptima.

Planificación no aprobativa.

Una disciplina de planificación se considera no apropiativa cuando, una vez que la CPU se ha asignado a un proceso, no puede ser retirada. En contraste, una disciplina es apropiativa si permite reasignar la CPU. En los sistemas no apropiativos, los trabajos de mayor duración retrasan a los de menor duración, pero el tratamiento de todos los procesos es más equitativo. Los tiempos de respuesta son más constantes, ya que los trabajos nuevos con alta prioridad no pueden interrumpir a los que ya están en espera

Referencias bibliográficas.

2 y 3 Obj., Crit. y Tipos de Planificación. (n.d.). Disponibles en: https://lsi.vc.ehu.eus/pablogn/docencia/manuales/SO/TemasSOuJaen/PLANIFICACIONDEPROCESOS/2y3Obj.,Crit.yTiposdePlanificacion.htm#apropiativa_y_noapropiativa

¿Qué es la Planificación de Procesos? - Términos y Definiciones de Ciberseguridad. (s.f.). Disponibles en: https://www.vpnunlimited.com/es/help/cybersecurity/process-scheduling

Alarcón, J. M. (2018, September 7). ¿Qué es un deadlock o interbloqueo? JASoft.org. Disponibles en: https://www.jasoft.org/Blog/post/191;Que-es-un-deadlock-o-interbloqueo

Andreegoso. (2016, May 3). U2 || 2.4 Concurrencia y secuencialidad. Sistemas Operativos. Disponibles en: https://andreegoso.wordpress.com/2016/03/15/u2-concurrencia-y-secuencialidad/

Concurrencia (informática) _ AcademiaLab. (s.f.). Disponibles en: https://academia-lab.com/enciclopedia/concurrencia-informatica/

Colaboradores de Wikipedia. (2024, April 8). Concurrencia (informática). Wikipedia, La Enciclopedia Libre. Disponibles en https://es.wikipedia.org/wiki/Concurrencia_%28inform%C3%A1tica%29

Colaboradores de Wikipedia. (2024, June 24). Algoritmo del banquero. Wikipedia, La Enciclopedia Libre. Disponibles en: https://es.wikipedia.org/wiki/Algoritmo_del_banquero

Colaboradores de Wikipedia. (2024, October 4). Bloqueo mutuo. Wikipedia, La Enciclopedia Libre. Disponibles en: https://es.m.wikipedia.org/wiki/Bloqueo_mutuo

Lili. (2011). “SERIABILIDAD.”. Disponibles en: https://sistemasoperativos-lili.blogspot.com/2011/08/seriabilidad.html

Martin. (2011). SERIABILIDAD. Disponible en: https://sistemasoperativosmartin.blogspot.com/2011/08/seriabilidad_25.html