INTERBLOQUEOS

Costas Rueda Juan Pablo
307081

13 de noviembre del 2023

Verónica López Martínez
Universidad Autónoma de Querétaro
Facultad de informática
Sistemas Operativos

Tabla de contenido

Resumen

Al momento en el cual un usuario o persona hace uso de una computadora, suelen ocurrir muchos procesos simultáneamente, estos procesos y tareas hacen uso de recursos compartidos que el propio sistema operativo y el hardware del sistema proporcionan. El problema es que ocurren interbloqueos cuando 2 o mas recursos quieren acceder a los mismos recursos al mismo tiempo. Estos quedan en un bloqueo permanente en caso de que los demás procesos tengan recursos que los demás necesiten. Para evitar este tipo de problemas, se han diseñado varios modelos y protocolos como los semáforos, sección critica, el modelo de operación normal, etc. que se encargan de gestionar los recursos compartidos y de aprobar o denegar acceso.

Palabras claves: Sistemas Operativos, Interbloqueos, Recursos, Computadoras

Introducción

Se podría argumentar que los recursos son los componentes lógicos y físicos más importantes de una computadora, sin ellos, la computadora simplemente no servirá, y desde un punto de vista más realista, la computadora no existiría. Y en este contexto, los interbloqueos son el enemigo mas grande de los recursos que hacen funcionar al sistema, así que es muy importante que los sistemas operativos se puedan entender lo que son los interbloques, como ocurren y como se pueden evitar.

La información mostrada en este escrito ah sido recopilada de múltiples sitios de internet confiables, además de mucha información de conocimiento de propio, así que algunos fragmentos de este documento no van a incluir referencia directa. Esta información se encuentra en la clase proporcionada por la Mtra. Verónica López Martínez de la Universidad Autónoma de Querétaro.

Cabe recalcar que este escrito incluye muy pocos ejemplos prácticos, este escrito de basa principalmente en descripciones y ejemplos teóricos, esto de debe a limitaciones de conocimientos y experiencia practica.

Desarrollo

Fundamentos y multiprogramación

Dentro de un sistema, existen muchos tipos de recursos y varían en muchas áreas, unos ejemplos de dichos recursos serian; un archivo en el disco duro, un espacio de memoria en la RAM, una variable de un proceso, un campo de información en una base de datos, un dispositivo conectado externamente, el procesamiento del CPU y GPU, la información almacenada en la cache, los propios procesos, etc. Básicamente la gran mayoría de una **computadora** se compone de múltiples tipos de **recursos**.

Debido a esto, muchos de los procesos que esta ejecutando la computadora hacen uso de los famosos **recursos compartidos**, lo especial de estos recursos es que estos pueden ser accedidos por múltiples procesos, pero esto puede ocasionar un problema.

Como usuario de computadora, siempre tenemos varios programas diferentes corriendo al mismo tiempo, por ejemplo, justo en este momento mi computadora esta corriendo el sistema operativo que tiene instalado, y dentro de ese sistema operativo estoy utilizando la aplicación Word para escribir este ensayo, pero en el fondo estoy reproduciendo música y tengo minimizada un editor de código. Así que en todo momento tenemos varios procesos corriendo al **mismo tiempo**, en donde cada uno puede que este haciendo uso de varios recursos compartidos.

Pero todo esto puede ocasionar un problema, ¿qué pasaría si dos o más **procesos** quieren acceder a los mismos recursos? Para responder lo anterior usare el siguiente ejemplo: Digamos que 2 niños, **Pedrito** y **Juan**, están dibujando algunas cosas para su clase, y están compartiendo los mismos lápices de colores. La maestra le asigna a cada uno un dibujo y Pedrito va a necesitar el color **rojo** y el **azul**, así que primero procede a tomar el color rojo y cuando quiere tomar el azul, Juan ya tiene ese lápiz en su mano. Así que Pedrito decide esperar a que Juan termine su dibujo para que pueda tomar el lápiz azul, pero Juan también necesita el rojo y el azul, así que Juan decide esperar a Pedrito a que el termine su dibujo para que pueda tomar el color rojo. En este caso **cada uno de los niños tiene un lápiz que el otro necesita para poder empezar** y están esperando a que acaben para que puedan empezar, así que básicamente **están atorados permanentemente y ninguno de los 2 puede avanzar**.

Esto mismo pasa con los procesos de una computadora, se quedan atorados permanentemente sin que ninguno de los 2 puedan avanzar. Así que los procesos se quedan sin terminar y los recursos de cada uno no se liberan. A esto se le conoce como **interbloqueo**.

En base a lo anterior, se han diseñado varias herramientas y metodologías para evitar que ocurran estos interbloqueos. Antes de explicar cuales son estas herramientas y metodologías, primero debo de explicar que es la **sección critica**.

La sección critica

La sección crítica es una parte del código que permite al proceso acceder a un recurso compartido que no debería de ser accedido por otro **hilo**, así que si un proceso está en la sección critica y esta usando un recurso compartido, cualquier otro proceso que quiera acceder a ese recurso tendrá que **esperar** a que se libere dicho recurso (Seccion Critica. Wikipedia, 2023).

Con todo esto, uno se podría preguntar, ¿no es más fácil hacer que varios procesos puedan acceder al mismo recurso al mismo tiempo? y la respuesta es si y no. Ya depende mucho del caso y para que se ocupe ese recurso. Para responder la pregunta, tenemos que hablar de los **problemas de sincronización**.

Problemas de sincronización

Para demostrar los problemas de sincronización, les muestro el siguiente caso:

Digamos que tenemos un archivo que tiene el valor 5 adentro de:

```
# numero.txt
5
```

Y tenemos 2 procesos que hacen uso de ese recurso:

```
# Proceso a
numero += 9
print(numero)
```

```
# Proceso b
numero -= 2
print(numero)
```

En este caso, primero queremos que se le sume 9 al valor original, y luego queremos que se le reste 2. Para que salgamos con un resultado de 12, (5 + 9 = 14 || 14 – 2 = 12, numero final termina siendo **12**). Pero si los 2 procesos se ejecutan al mismo **los 2 absorben el valor de 5 al mismo tiempo** y no se enteran del cambio que hizo el otro, **así que el valor final del numero termina siendo del proceso que termino al final** (5 + 9 = 14 || 5 – 2 = 3, numero final termina siendo **3**). A este tipo de problemas se le conoce como un **problema de sincronización**, en donde debido a que un recurso de modifico sin ser previsto o esperado.

Así que ya depende para que se ocupe el recurso, como se ocupe y si se va a modificar el recurso, pero en la mayoría de los casos es necesario que solo un proceso tenga acceso a un recurso simultáneamente (Seccion Critica. Wikipedia, 2023).

Ahora sí, ya podemos ir a los métodos para evitar los interbloqueos y los problemas de sincronización.

La exclusión mutua

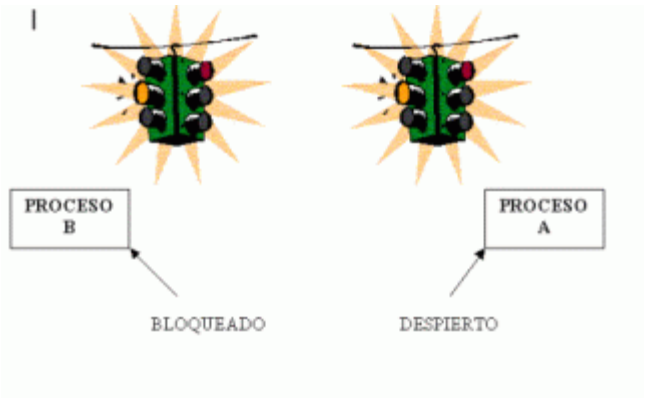
La exclusión mutua es la actividad que realiza el sistema operativo para **evitar** que dos o más procesos ingresen al mismo tiempo a un área de datos compartidos o accedan a un mismo recurso, en otras palabras, es la condición por la cual, de un conjunto de procesos, sólo uno

puede acceder a un recurso dado o realizar una función dada en un instante de tiempo (Web Programacion, 2008).

Las actividades o algoritmos que se utilizan en la exclusión mutua son varias, estas son:

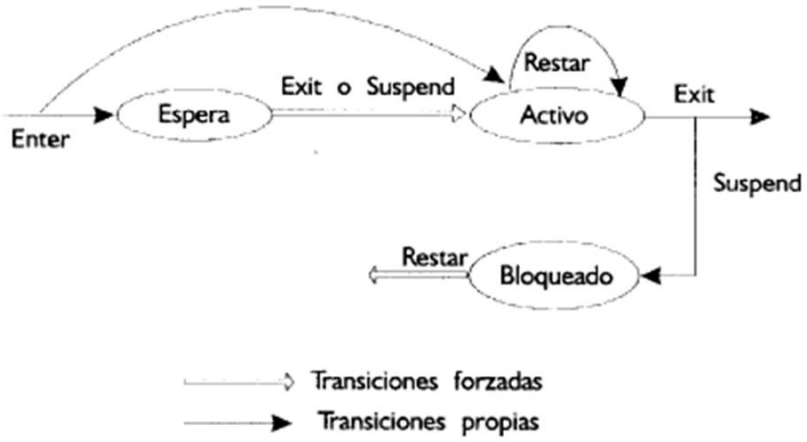
- **Semáforos**. Esta técnica permite resolver la mayoría de los problemas de sincronización entre procesos y forma parte del diseño de muchos sistemas operativos y de lenguajes de programación concurrentes. Un semáforo binario es un indicador de condición que registra si un recurso está disponible o no (Web Programacion, 2008).

Ilustración 1 – Galuf Space



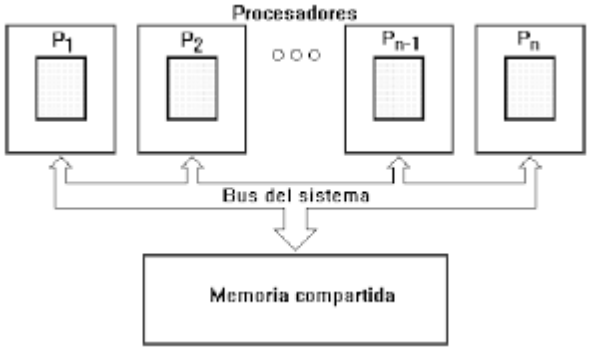
- **Monitores**. Un monitor es una estructura formada por una cabecera que los identifica, un conjunto de variables globales a todos los procedimientos del monitor, un conjunto de procedimientos y un bloque de inicialización, el cual se ejecuta una única vez, cuando se crea el monitor (Web Programacion, 2008).

Ilustración 2 - eq2-sistemasoperativos.blogspot.com



- **Solución de Peterson**. Básicamente se utilizan banderas, parecidos a los semáforos, que tienen un valor booleano para permitir o denegar acceso, en el cual primero se verifica si el recurso al que se quiere acceder esta disponible, y si es así. Este le permite al proceso acceder a la sección critica, pero si no está disponible, no deja que pueda acceder a dicho recurso (Infor, s/n).

Ilustración 3 – www.aiu.edu



Transacciones atómicas

Quiero aprovechar este momento para hablar de las transaccione atómicas, en una **base de datos**, puede que tengas muchas tablas, con muchas columnas, muchos registros y muchos usuarios que están interactuando con la información de la base de datos constantemente. Así que aplicar muchos de los métodos mencionadas anteriormente no pueden ser ideales por cuestiones de disponibilidad y rendimiento, así que para evitar problemas de sincronización en las bases de datos, se aplican las transacciones atómicas. Estos en pocas palabras, las **transacciones atómicas** son tareas en donde un proceso manipula un recurso y **hace cambios que se vean reflejados instantáneamente** cuando este termine con éxito, en caso de que este no termine con éxito, todos los cambios son revertidos y no ven reflejados para los otros procesos. Para explicar lo anterior, pondré de ejemplo un **INSERT** en una tabla de una base de datos. Digamos que acabo de crear una tabla para tener un registro de los empleados de una empresa, y voy a ejecutar un **QUERY** para insertar 1000 empleados. Esto claramente va a tomar un poco de tiempo, digamos que unos 30 segundos, pero si dentro de esos 30 segundos hago un **FETCH** para sacar los datos de la tabla, no voy a recibir nada de información, debido a que se están insertando mediante una transacción atómica, **otros procesos no pueden ver esa información y los cambios no se han visto reflejados**. Ya que los 1000 empleados se hayan insertado y la tarea

haya concluido con éxito, si hago un **FETCH**, ya podre ver a los 1000 empleados, los cambios fueron instantáneos para los demás procesos, aunque este haya tardado 30 segundos. En cambio, digamos que en el empleado 800 se escribieron mal los datos y no se pudo insertar al empleado, debido a esto, la tarea se va a cancelar y los cambios se van a revertir a cuando estaban antes, así que la tabla de empleados se mantendrá vacía. Estos casos de las transacciones atómicas se utilizan en casos donde muchos procesos quieren acceder a los recursos compartidos, como, las bases de datos, sistemas bancarios y servidores como tal, pero también tienen su aparición e importancia a en los sistemas de multioperacion. (Sistemas en tiempo real, s/n)

Métodos para tratar con los interbloqueos

Ya vimos varias maneras en cómo **prevenir interbloqueos**, como la exclusión mutua, y como evadir los interbloqueos, como las transacciones atómicas, pero aun así es probable que lleguen a ocurrir interbloqueos en casos especificos, así que también es importante desarrollar tareas para la **detección y recuperación de interbloqueos**.

Para la detección, se pueden utilizar algoritmos como el algoritmo de la pila, este algoritmo utiliza una pila para representar el estado de los procesos. La pila se actualiza constantemente a medida que los procesos solicitan y liberan recursos. La detección de interbloqueos se realiza mediante la comprobación de la pila para identificar cualquier proceso que haya solicitado un recurso que ya está en uso por otro proceso. (Learn Microsoft, 2023)

La detección de interbloqueos es una técnica importante para evitar la aparición de interbloqueos en un sistema operativo. Sin embargo, no es una solución perfecta. La detección de interbloqueos puede ser costosa en términos de tiempo y recursos, y puede no ser capaz de detectar todos los interbloqueos. En algunos casos, puede ser más eficiente y efectivo **evitar la aparición de interbloqueos** en primer lugar. Esto se puede hacer utilizando estrategias de prevención de interbloqueos (Learn Microsoft, 2023).

Como ya vimos varias veces, los interbloqueos se ocasionan cuando varios procesos quieren acceder a los mismos recursos, pero terminan en un **bloqueo permanente**. Así que para lidiar con esto hay 2 opciones:

- **El aborto**, básicamente termina la ejecución del proceso para que el recurso sea liberado y utilizado por el otro proceso
- **El privar**, en el cual se le quita los recursos a un proceso para que puedan ser ejecutados por el otro, esto implica que el proceso podrá continuar, pero es muy posible que no pueda cumplir con todas sus tareas.

Para decidir que recurso será abortado o privado, se pueden utilizar múltiples algoritmos, algunos de esto serian; un algoritmo de prioridad, en donde el proceso con mayor nivel de prioridad sale beneficiado, un algoritmo cuantitativo, en donde se proceso con mayor cantidad de recursos compartidos sale beneficiado, o un algoritmo aleatorio, en donde el proceso beneficiado es completamente aleatorio.

Podemos aplicar este caso en el ejemplo de los niños con los dibujos, primero se decide a que niño se le dará prioridad o mas importancia, elijamos a Pedrito porque es el favorito de la maestra.

En el caso de **aborto**, le indicas a Juan que se salga del salón por el resto de la clase, así que Juan ya no puede hacer su tarea y suelta el lápiz azul, permitiendo a Pedrito tomar el lápiz y **terminar** de hacer su dibujo.

Y en el caso de **privar**, le indicas a Juan que suelte el lápiz azul, permitiendo que Pedrito pueda terminar su dibujo. Pero sin los lápices de color, Juan no puede terminar su dibujo y tiene 0 en su tarea.

Ya con todo esto, se vio en amplitud la gestión y control de recursos compartidos, pero solo falta ver como se gestionan los recursos en general, el **modo de operación normal** es el encargado de todo esto.

Modo de operación normal

El modo de operación normal se refiere a la forma estándar en que el sistema utiliza y gestiona los recursos. Esto implica la ejecución de procesos, la asignación de memoria, el acceso a dispositivos y otras operaciones cotidianas. En resumen, el modo de operación normal abarca todas las operaciones del sistema.

En el caso donde un proceso quiere acceder a un archivo, ocurre lo siguiente:

1. **Solicitud del Proceso**: Un programa desea abrir un archivo para leer información almacenada en él.
2. **Generación de una Solicitud**: El programa envía una solicitud al sistema operativo indicando que necesita acceso de lectura a un archivo específico.
3. **Validación de la Solicitud**: El sistema operativo verifica si el programa tiene los permisos necesarios para acceder al archivo y si el archivo existe.
4. **Asignación del Recurso**: Si la solicitud es válida, el sistema operativo permite al programa abrir el archivo en modo de lectura. Asigna los recursos necesarios, como buffers de memoria, para facilitar la lectura de datos.
5. **Uso del Recurso**: El programa lee la información del archivo utilizando los recursos asignados por el sistema operativo.
6. **Liberación del Recurso**: Después de leer la información, el programa cierra el archivo, liberando así los recursos asignados. Otros programas pueden ahora acceder al mismo archivo si es necesario.

Esto muestra la secuencia en la que se emplea un recurso, primero se realiza una solicitud de recurso, luego se hace uso de ese recurso, y por último, el recuso es liberado. Un ejemplo de esta interacción se puede mostrar al trabajar con un programa como Microsoft Word. Cuando el usuario intenta realizar acciones como cambiar el nombre o mover un archivo que está siendo utilizado por Word, el sistema operativo utiliza registros para evitar conflictos. El registro de control del sistema operativo mantiene un seguimiento de los procesos en ejecución, mientras que Word utiliza sus propios registros para gestionar la apertura y modificación de archivos.

Los registros mencionados en el ejemplo son generados por el modo de operación normal, y existen varios tipos de registros, están los registros de modo que indican si el modo de operación es kernel o usuario, el registro de segmento, el registro de estado, etc.

Estos registros son utilizados para conocer información importante sobre los procesos y los recursos.

Conclusión

Entonces, como ya vimos anteriormente, existen muchos retos a la hora de lidiar con interbloqueos. A lo largo de los años se han creado metodologías para prevenir y lidiar con ellos, además de crear métodos para lidiar en casos donde si haya ocurrido un interbloqueo. Uno podría pensar que todo esto ya se solucionó y que no hay que preocuparse por los interbloqueos, pero eso esta muy fuera de los correcto, los interbloqueos como tal siguen ocurriendo en casos muy específicos y los propios métodos de prevención y corrección causan problemas de compatibilidad y rendimiento. Aunque la industria ya tiene muchos años trabajando con la misma arquitectura en la multiprogramación, macOS recientemente se cambió a ARM de RISC la cual trabaja con diferentes métodos y las computadoras quánticas están por llegar, las cuales pueden causar aun mas problemas con los recursos.

Dado a que los interbloqueos son un problema que va a estar presente por muchos años a seguir, así que es muy importante tener bien documentado cuales son los métodos que se han desarrollado para lidiar con los interbloqueos.

Uno de los retos mas grandes a la hora de realizar este escrito es redactarlo y describirlo de manera que sea entendible para la gran mayoría de gente posible, los sistemas operativos son un tema muy complejo y difícil de entender, pero lo que es aún más complicado es enseñarlo o mostrarlo a un publico que no conoce muy bien el tema, así que me puse la tarea de dar ejemplos y muestras que se relacionen con la vida diaria de los demás, como fue el ejemplo de los niños con los lápices. Además de que decidí redactar el escrito de manera amigable para el público, a diferencia de otros escritos que son muy especializados y difíciles de entender si no estas familiarizado con el tema.

Referencias

Infor. (s/n de s/n de s/n). *EL ALGORITMO DE PETERSON PARA N PROCESOS*. Obtenido de Infor: https://www.infor.uva.es/~cllamas/concurr/pract98/sisos4/index.html

Learn Microsoft. (14 de June de 2023). *Detección de interbloqueos*. Obtenido de Learn Microsoft: https://learn.microsoft.com/es-es/windows-hardware/drivers/devtest/deadlock-detection

Seccion Critica. Wikipedia. (19 de November de 2023). *Sección crítica*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Secci%C3%B3n_cr%C3%ADtica

Sistemas en tiempo real. (s/n de s/n de s/n). *Acciones atómicas, Tareas Concurrentes y Fiabilidad*. Obtenido de Sistemas en tiempo real: https://uned-sistemas-tiempo-real.readthedocs.io/es/latest/tema07.html

Web Programacion. (1 de January de 2008). *Exclusión mutua*. Obtenido de Web Programacion: https://webprogramacion.com/exclusion-mutua/