Instituto Politécnico Nacional

Escuela Superior de Computo

Aplicaciones para comunicaciones en Red
Practica 2

Integrantes
Anaya Martínez Victor Alejandro (3CV8)
Pérez Garduño José Emiliano (3CV7)
Meléndez Padilla Mauricio (3CV7)

Los Mecanismos de sincronización son una herramienta que evita la inconsistencia en un programa, su función radica en la detención de la ejecución de un proceso hasta que ocurra otro evento en otro proceso. De estos mecanismos existe variedad, por ejemplo, están las señales, tuberías, el paso de mensajes, los mutex, y lo que usamos en esta práctica, los semáforos.

La sincronización de procesos, así como el uso de secciones criticas es muy útil, y lo vemos todos los días en la vida real, con el mismo nombre que posee este mecanismo: los semáforos, en el tráfico vehicular nos ayuda a evitar accidentes, dicta cuando un auto tiene el camino libre para cruzar y cuando debe esperar a que otros autos crucen, de manera análoga funciona los semáforos posix, determinan cuando un proceso puede acceder a cierto recurso compartido (también conocido como sección critica), y cuando este está bloqueado, impidiendo que otro proceso acceda a dicho recurso. Las secciones criticas técnicamente son una "idea", un "concepto", en realidad puede ser un espacio de memoria, una variable global, un recurso modificable y accesible para varios hilos/procesos.

¿Cómo es que funcionan los semáforos?

Un semáforo básico es una estructura formada por una posición de memoria y dos instrucciones, una para reservarlo y otra para liberarlo. A esto se le puede añadir una cola de hilos para recordar el orden en que se hicieron las peticiones.

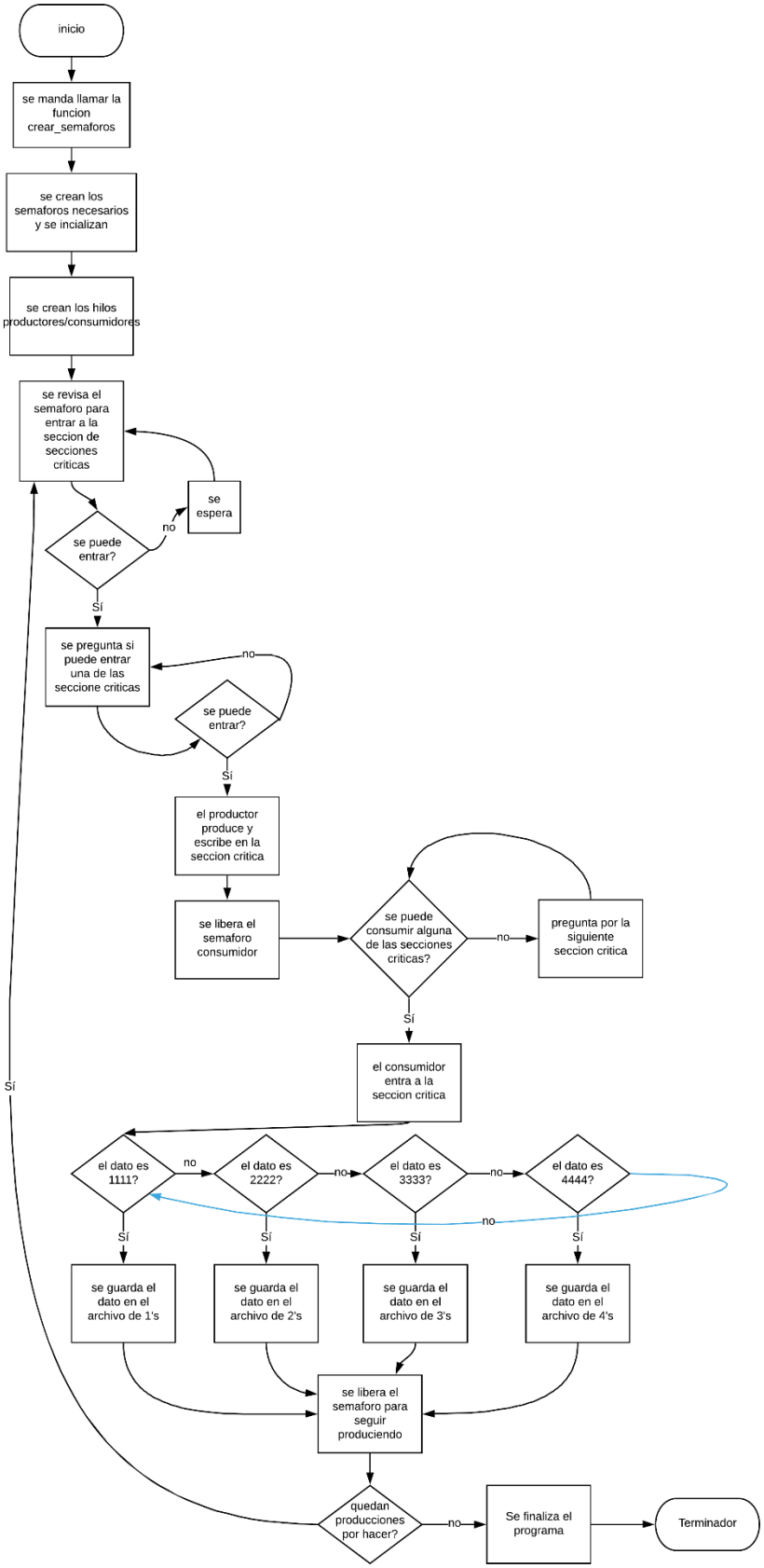
Se inicializa la posición de memoria a 1 (o al valor correspondiente si ese recurso concreto admite más de un acceso simultáneo). Esto se hace en el inicio del programa principal.

A continuación, cada vez que un hilo o un proceso quiera acceder a dicho recurso (por ejemplo, un fichero), hará primero una petición con la primera de las llamadas disponibles. Cuando el S.O. ejecuta esa llamada, comprueba el valor que hay en la posición de memoria del semáforo, y si es distinta de cero, se limita a restarle 1 y devolver el control al programa; sin embargo, si ya es cero, duerme al proceso que hizo la petición y lo mete en la cola de procesos, en espera de que el semáforo se ponga a un valor distinto de cero.

Por último, cuando el proceso ha terminado el acceso al recurso, usa la segunda llamada para liberar el semáforo. Cuando el S.O. la ejecuta, comprueba si la cola del semáforo está vacía, en cuyo caso se limita a incrementar el valor del semáforo, por el contrario, si tiene algún proceso, lo despierta, de modo que vuelve a recibir ciclos de CPU y sigue su ejecución. Si había varios procesos en espera, se irán poniendo en marcha uno tras otro a medida que el anterior va liberando el semáforo. Cuando termina el último, el semáforo se vuelve a poner a 1. Se trata, por tanto, del mismo proceso que seguiríamos con la variable, pero con la ventaja de que es un mecanismo estándar para todos los procesos, y como es una operación atómica (esto es, que durante su ejecución no se admiten cambios de tarea), no surge el problema de que una conmutación pueda producir errores aleatorios.

La función que cumplen es como se mencionó antes, evitar que dos procesos

entren en una misma variable, pero ¿esto para que nos sirve? Expliquémoslo de una manera sencilla y que todos puedan entender, imaginemos que estamos en una pastelería, en la cual hay dos chef, pero un solo horno donde solo cabe un pastel, al primer chef se le solicita hacer un pastel, este chef responde al llamado haciendo el pastel y metiéndolo al horno, pero al segundo chef también se le solicito hacer un pastel entonces el segundo chef sacara el pastel que puso el primer chef en el horno y meterá el suyo, y para cunado el chef 1 se de cuenta de lo que acaba de pasar ya será demasiado tarde, lo que acaba de pasar en este ejemplo es que dos procesos (los chef) necesitaron hacer uso del mismo recurso compartido (el horno) y como no hay nadie vigilando el horno situaciones como esta pueden pasar.


inicio
se manda llamar la funcion crear_semaforos
se crean los semaforos necesarios y se incializan
se crean los hilos productores/consumidores
se revisa el semaforo para entrar a la seccion de secciones criticas
se puede entrar?
no
se espera
Sí
se pregunta si puede entrar una de las seccione criticas
no
se puede entrar?
Sí
el productor produce y escribe en la seccion critica
se libera el semaforo consumidor
se puede consumir alguna de las secciones criticas?
no
pregunta por la siguiente seccion critica
Sí
el consumidor entra a la seccion critica
el dato es 1111?
no
el dato es 2222?
no
el dato es 3333?
no
el dato es 4444?
no
Sí
Sí
Sí
Sí
se guarda el dato en el archivo de 1's
se guarda el dato en el archivo de 2's
se guarda el dato en el archivo de 3's
se guarda el dato en el archivo de 4's
se libera el semaforo para seguir produciendo
quedan producciones por hacer?
Sí
no
Se finaliza el programa
Terminador

A continuación, se explicará la imagen anterior que es el funcionamiento del programa.

El programa inicia con la inicialización de los semáforos seguido por la creación de los hilos productores y los hilos consumidores. Independientemente de que hilo se cree primero, ya sea un consumidor o un productor, el primero en poder ejecutar su función será el hilo productor ya que el hilo consumidor tiene un semáforo que no le permite entrar a la sección donde están las secciones criticas puesto que como el programa se acaba de iniciar, no hay nada que consumir por lo que no es necesario que el consumir entre, una vez aclarado esto podemos proseguir. El hilo productor entra a la sección donde están las secciones críticas, llamémosla "Sección main", una vez dentro de la sección main va a preguntar si la primera sección critica está disponible, en caso de que lo este, va a entrar y va a guardar el dato en la sección critica, luego, en este punto puede suceder una de dos que entre otro productor a producir o que un consumidor entre a consumir la producción que ya fue realizada. Este ciclo se va a repetir constantemente. El funcionamiento del consumidor es un poco diferente al del productor, en cuanto a los semáforos funcionan de la misma forma, se bloquea el semáforo consumidor para que otro consumidor no consuma el dato, una vez dentro de la sección critica el consumidor leerá el dato, y dependiendo del valor de este dato, va a guardarlo en un archivo, una vez guardado el valor se liberara el semáforo productor para que este pueda seguir produciendo.

Conclusiones:

Anaya Martínez Victor Alejandro

Este tema ha sido el más divertido que he visto, al principio fue difícil entender cómo funcionaban los semáforos sin embargo solo con la práctica y un poco de documentación es fácil entenderlos. La verdad es que resultan muy útiles para evitar la inconsistencia de datos porque si bien al usar hilos, la velocidad de ejecución puede disminuir drásticamente también puede haber errores, y si, la velocidad es algo que nos importa, pero es más importante conservar la información sin errores, aunque se tome un poco más de tiempo y un poco más de trabajo hacerlo.

Meléndez Padilla Mauricio

Para esta práctica, el objetivo de emplear un método de sincronización de procesos, específicamente semáforos, nos demuestra la utilidad y necesidad de poder controlar el funcionamiento y tiempos de acceso a zonas de memoria por parte de diferentes procesos para evitar que intenten acceder al mismo tiempo y exista una colisión con ellos, el definir las reglas de su manejo y marcar los límites de uso y acceso de cada proceso es vital para el correcto funcionamiento de un programa con tareas corriendo de manera paralela, sin estas maneras de controlarlas, el programa puede volverse "impredecible" o simplemente no funcionar y realizar su objetivo.

José Emiliano Pérez Garduño:

El uso de semáforos nos ayuda a prevenir errores en tiempo de ejecución, ofrecen una gran ayuda al momento de trabajar con programas multi-hilos, pero de manera similar a la práctica anterior, no podemos escoger que hilos se ejecutaran primero y como consecuencia se debe planificar una correcta implementación de semáforos o de cualquier mecanismo de sincronización.

Al terminar esta práctica logramos reforzar los conocimientos que adquirimos a lo largo del parcial y además logramos arreglar problemas que no sabríamos ni como plantear previamente, ya que, a la hora de desarrollarla, nos dimos cuenta de los errores que surgían a partir de la utilización de muchos hilos, el tener que hacerlos funcionar de manera secuencial, evitar conflictos a la hora de trabajar con la zona crítica y los

memory leaks que puedan llegar a darse al no utilizar fclose.