**INSTITUTO POLITECNICO NACIONAL**

*ESCUELA SUPERIOR DE CÓMPUTO*



**Reporte**: Practica 2

**Asignatura**: Aplicaciones para comunicaciones de Red

**Profesor**: Rangel Gonzales Josué

**Alumnos**: Herrera Ramírez Absalom

García Yáñez Salatiel

**Introducción**

Un semáforo es una estructura diseñada para sincronizar dos o más threads o procesos, de modo que su ejecución se realice de forma ordenada y sin conflictos entre ellos.

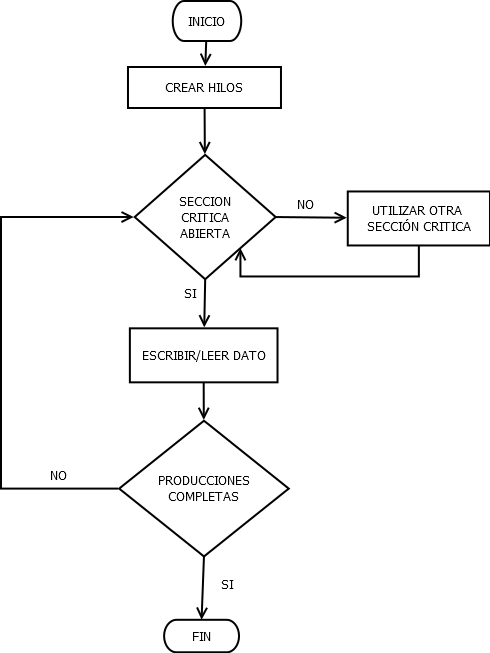
El por qué no se pueden usar directamente otras estructuras mas clásicas, como por ejemplo usar una variable común para decidir si se puede o no acceder a un recurso, se debe a que estamos en un sistema multitarea: hacer esto implicaría realizar una espera activa Por otro lado, puede ocurrir algo mucho peor: supongamos que un proceso comprueba la variable, y ve que el recurso está libre, por lo que procedería a cambiar dicha variable de valor y seguir. Pues bien, si justo después de la comprobación pero antes de que cambie el valor se conmuta de tarea (puede pasar, pues el sistema operativo puede hacerlo en cualquier momento), y el nuevo proceso comprueba la variable, como todavía no se ha actualizado, creerá que el recurso está libre, e intentará tomarlo, haciendo que ambos programas fallen. Lo peor del caso es que se tratará de un error aleatorio: unas veces fallará y otras no.

Para evitarlo, se idearon los semáforos. Un semáforo básico es una estructura formada por una posición de memoria y dos instrucciones, una para reservarlo y otra para liberarlo.

Se empieza por inicializar 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 thread 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.

En la siguiente practica utilizaremos los semáforos para producir un numero determinado de acciones, las cuales serán guardadas en unos archivos de texto,también utilizaremos hilos, para que sea mas rápido el proceso(por eso los semáforos).



**Propuesta de solución**

Se crean semáforos para las secciones críticas y los archivos de texto. Cada sección critica tendrá dos semáforos, un semáforo controlará los productores y otro los consumidores para evitar que ingrese mas de un consumidor o productor a cada sección. Es importante tener un semáforo en el archivo de texto, ya que si no se tiene no se podrá controlar que se cuenten producciones incorrectas o una sobreescritura.

Del lado de los consumidores; un consumidor puede tener distinto numero de consumos que los demás, pero los consumos totales que se tienen en los archivos de texto deben de ser iguales para todos.

Cuando un productor termine de producir, el consumidor será el encargado de entrar a la zona crítica, obtener la producción y contar cuántos produjo, para así tener un control de las producciones totales y evitar que se produzcan de mas o menos.

**Conclusiones:**

**Absalom Herrera Ramrírez**: El uso de los hilos es muy útil a la hora de agilizar programas, ya que se pueden realizar diferentes operaciones al mismo tiempo, permitiendo así un mejor tiempo de respuesta, ya que se aprovechan mejor los recursos de la máquina. Además con los semaforos tenemos más posibilidades para hacer operaciones más complejas que sin ellos producirían problemas de inconsistencia de datos.

**Salatiel García Yañez**: Los hilos nos ayudan a hacer varios procesos al mismo tiempo, pero sería imposible usarlos sin controlar una sobreescritura de nuestros datos. Con los semáforos podemos controlar que nuestros datos se escriban de manera ordenada y adecuada, permitiéndonos hacer procesos que demandan de muchas operaciones de manera rápida y ordenada.