



Instituto Politécnico
Nacional

Escuela Superior de
Cómputo



Aplicaciones para Comunicaciones en Red

Docente: **Dr. Josué Rangel González.**

Práctica 2. ***Problema del productor y consumidor***

Presentan:

César Uriel Hernández Castellanos
Mauricio Joel Martínez Islas
Carlos Pimentel González

Fecha de entrega: **2 de marzo de 2020**

Grupo: **3CV5**

Índice

1. Introducción.	3
1.1. El problema del productor-consumidor	3
1.2. Definición de semáforo	3
1.3. Sincronización	4
1.4. Inconsistencia de datos	4
1.5. Sección crítica	4
1.6. Definición de hilo	4
1.7. Estados de un hilo	5
2. Descripción del problema.	5
3. Descripción de la solución.	6
4. Salida del programa.	11
5. Conclusiones.	12

Índice de figuras

1.	Funcionamiento de un semáforo.	3
2.	Funcionamiento de un semáforo.	3
3.	Sección critica.	4
4.	Ejecución de un hilo en un procesador	4
5.	Ciclo de vida de un hilo [1]	5
6.	Función myWait(id).	6
7.	Función mySignal(id).	6
8.	Función eliminarSemaforo(id).	7
9.	Función que realiza la creación de semáforos e hilos productores y consu- midores.	7
10.	Función del productor.	8
11.	Función del consumidor	9
12.	Salida del programa	11

1. Introducción.

1.1. El problema del productor-consumidor

En computación, el problema del productor-consumidor es un ejemplo clásico de problema de sincronización de multiprocesos. El programa describe dos procesos, productor y consumidor, ambos comparten un buffer de tamaño finito. La tarea del productor es generar un producto, almacenarlo y comenzar nuevamente; mientras que el consumidor toma (simultáneamente) productos uno a uno. El problema consiste en que el productor no añada más productos que la capacidad del buffer y que el consumidor no intente tomar un producto si el buffer está vacío.

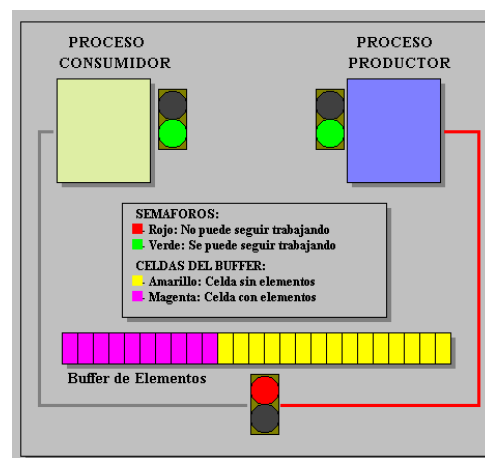


Figura 1: Funcionamiento de un semáforo.

1.2. Definición de semáforo

Un semáforo es una variable especial que constituye una serie de opciones elementales para poder restringir o garantizar el acceso a los recursos en un sistema operativo con un entorno de multiprocesamiento

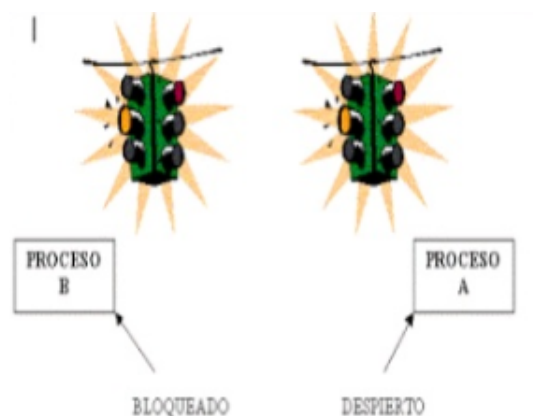


Figura 2: Funcionamiento de un semáforo.

1.3. Sincronización

Hace referencia a la coordinación de procesos que se ejecutan simultáneamente para completar una tarea, con el fin de obtener un orden de ejecución y evitar estados inesperados

1.4. Inconsistencia de datos

Esta se presenta cuando se repiten innecesariamente datos en los archivos, decimos que hay redundancia de datos cuando la misma información es almacenada en el mismo sistema.

1.5. Sección crítica

Se denomina sección crítica o región crítica, en programación concurrente, a la porción de código de un programa de ordenador en la que se accede a un recurso compartido que no debe ser accedido por más de un proceso o hilo en ejecución. La sección crítica por lo general termina en un tiempo determinado el hilo, proceso o tarea solamente tendrá que esperar un periodo determinado de tiempo para entrar

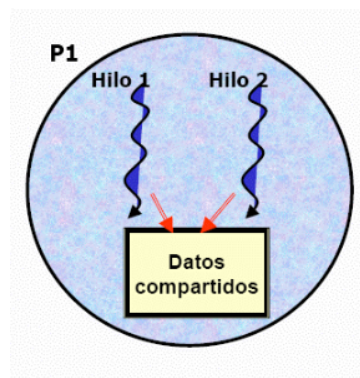


Figura 3: Sección crítica.

1.6. Definición de hilo

En sistemas operativos, un hilo, proceso ligero o subproceso es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo

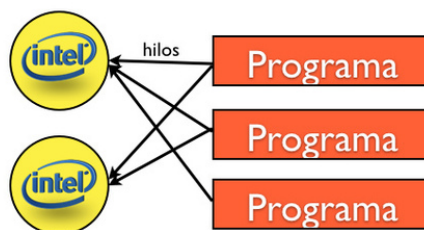


Figura 4: Ejecución de un hilo en un procesador

1.7. Estados de un hilo

Existen tres estados para un hilo (son los mismos que existen para el modelo tradicional de un proceso):

1. En ejecución (utilizando el procesador)
2. Listo (ejecutable, detenido momentáneamente para dejar que se ejecute otro hilo)
3. Bloqueado (no se puede ejecutar hasta que ocurra algún evento)

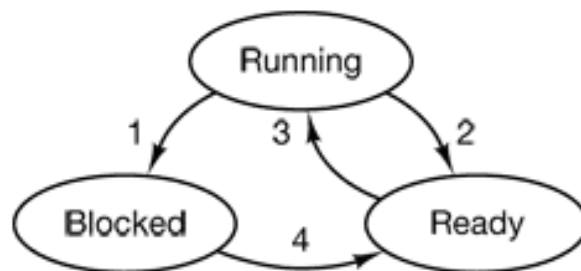


Figura 5: Ciclo de vida de un hilo [1]

2. Descripción del problema.

Genera un sistema de producción-consumo, utilizando la técnica de sincronización basada en semáforos System V. Las características que debe de contener la práctica son las siguientes:

- Generar un proceso, encargado de crear 4 hilos productores y 3 hilos consumidores.
- Generar un sección critica que este dividida en 5 filas.
- Genera un arreglo de semáforos (nosotros decidimos el numero de semáforos a crear), que permita la correcta sincronización.
- Cada productor realizara al menos 1000 producciones
- Los valores a producir son
 - Hilo 1: 1,2,3,4,...,1000
 - Hilo 2: a,b,c,...,z
 - Hilo 3: caracteres Unicode
 - Hilo 4: números primos
- Una vez que un productor realizo una producción tendrá que salir de la sección critica y tendrá que intentar volver a entrar a ella, si es que existen lugares disponibles.
- Los consumidores asignan los valores leídos a archivos tipo log

3. Descripción de la solución.

Un semáforo únicamente puede acceder usando las siguientes operaciones (myWait mySignal) que se encuentran explicadas a continuación:

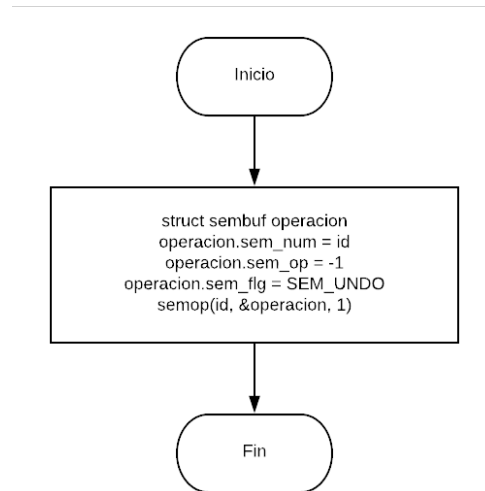


Figura 6: Función `myWait(id)`.

En la *figura 6* es posible observar la implementación de la función `myWait`, la cual se manda a llamar cuando un hilo desea acceder a un recurso, además de decrementar el semáforo

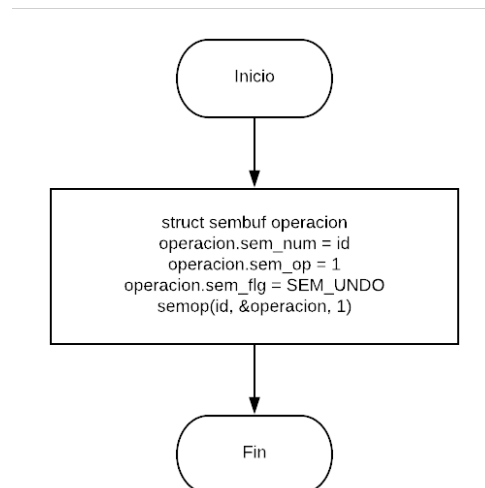


Figura 7: Función `mySignal(id)`.

En la *figura 7* se muestra la función `signal` que se llama a `signal` cuando un hilo libera un recurso

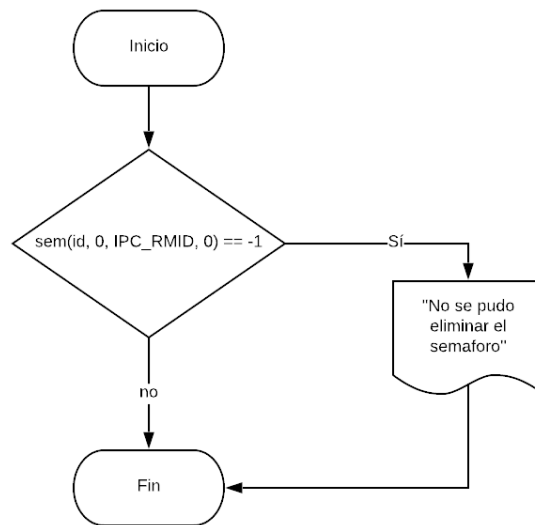


Figura 8: Función eliminarSemaforo(id).

En la *figura 8* se muestra el funcionamiento de la función que elimina un semáforo, para esto se cambia los permisos y características de un conjunto de semáforos. Se debe llamar con un ID de semáforo válido. El valor de semnum selecciona un semáforo de una matriz por su índice. El argumento cmd es el indicador de control IPCRMID el cual elimina un conjunto de semáforos especificado, en caso de que la función nos retorne menos uno, nos indicará que la eliminación no se hizo de manera exitosa.

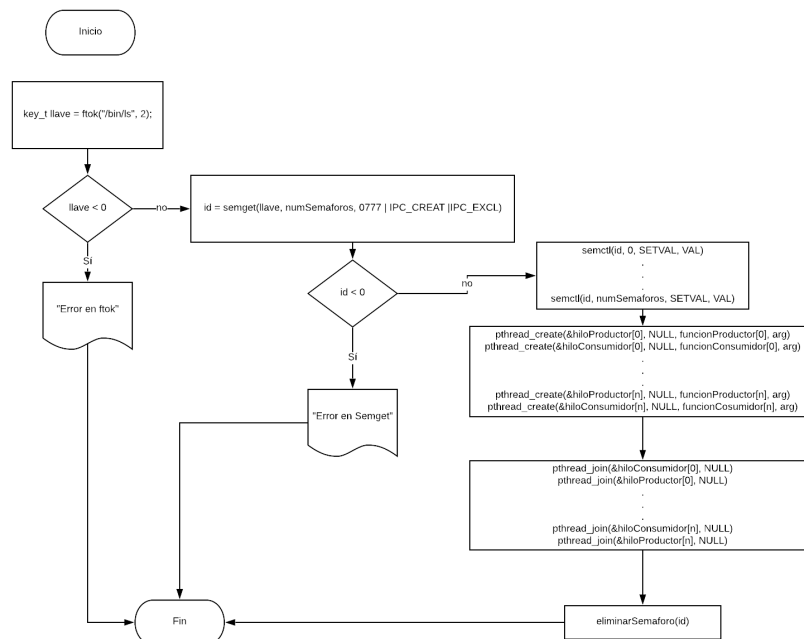


Figura 9: Función que realiza la creación de semáforos e hilos productores y consumidores.

En la *figura 9* se muestra la función encargada de crear la llave, que se realiza con la función `ftok` que recibe como parámetro un directorio, en nuestro caso se pasa el directorio del comando `ls`, para posteriormente preguntar si la llave se creó de manera exitosa.

Como siguiente paso se crean los semáforos con la función `semget`, en caso de que se creen de manera correcta los semáforos se inicializan de acuerdo a las características del problema que se está tratando, lo cual se indica más adelante.

Como penúltimo paso se crean los hilos productor y consumidor, para que finalmente se eliminen los semáforos, esto con el propósito de evitar errores en futuras ejecuciones del programa.

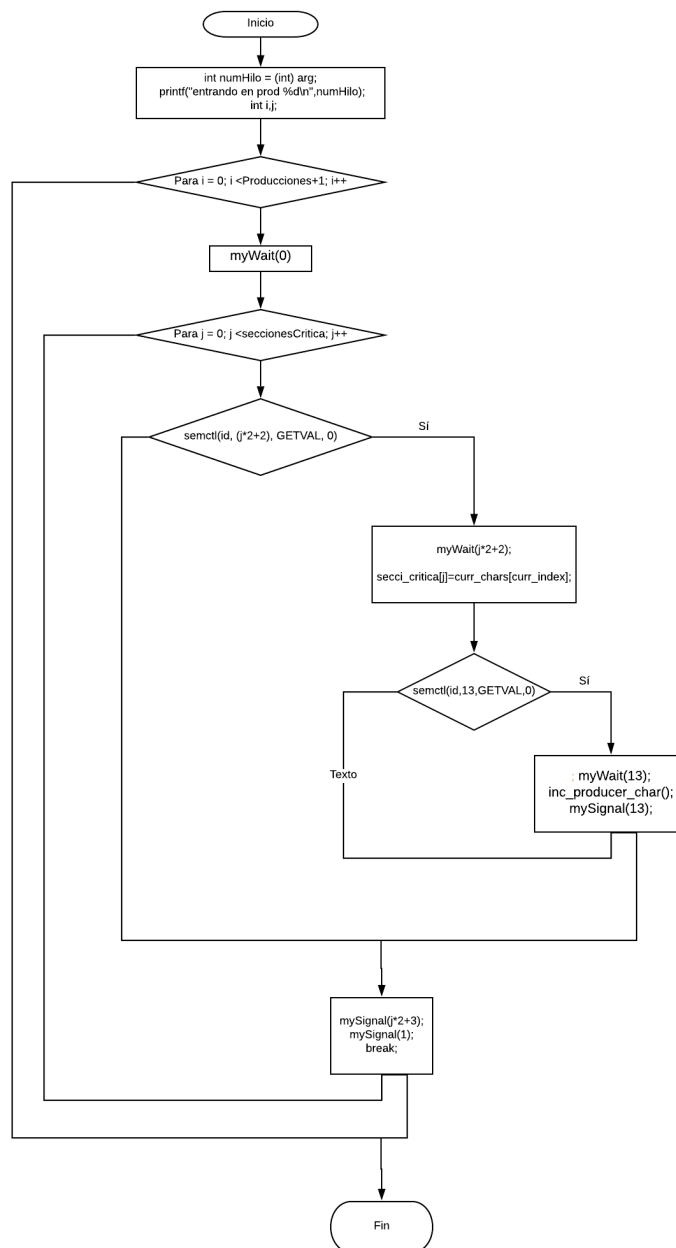


Figura 10: Función del productor.

Primeramente se obtiene el identificador de cada uno de los hilos para poder diferenciarlos, para proseguir a iterar de 0 al número de productores, posteriormente se decrementa el semáforo cero.

Como segundo pasa se itera de 0 al número de secciones criticas, con el fin de generalizar las condiciones de la disponibilidad de la i-ésima sección critica

En caso de que se encuentra disponible se decrementa el semáforo $j*2+2$ para llenar la j-ésima sección critica con los caracteres que se desean.

Como penúltimo paso se verifica que el semáforo id se encuentre disponible, en tal caso se decremente el semáforo del contador, se libera el recurso, además de obtener el carácter correspondiente.

Finalmente se libera $j*2+3$ y 1

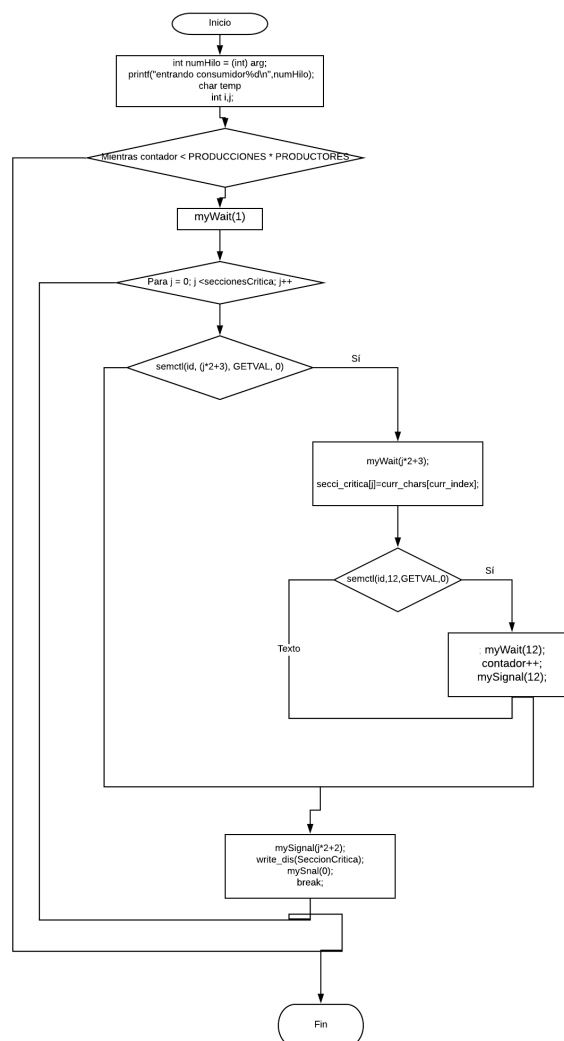


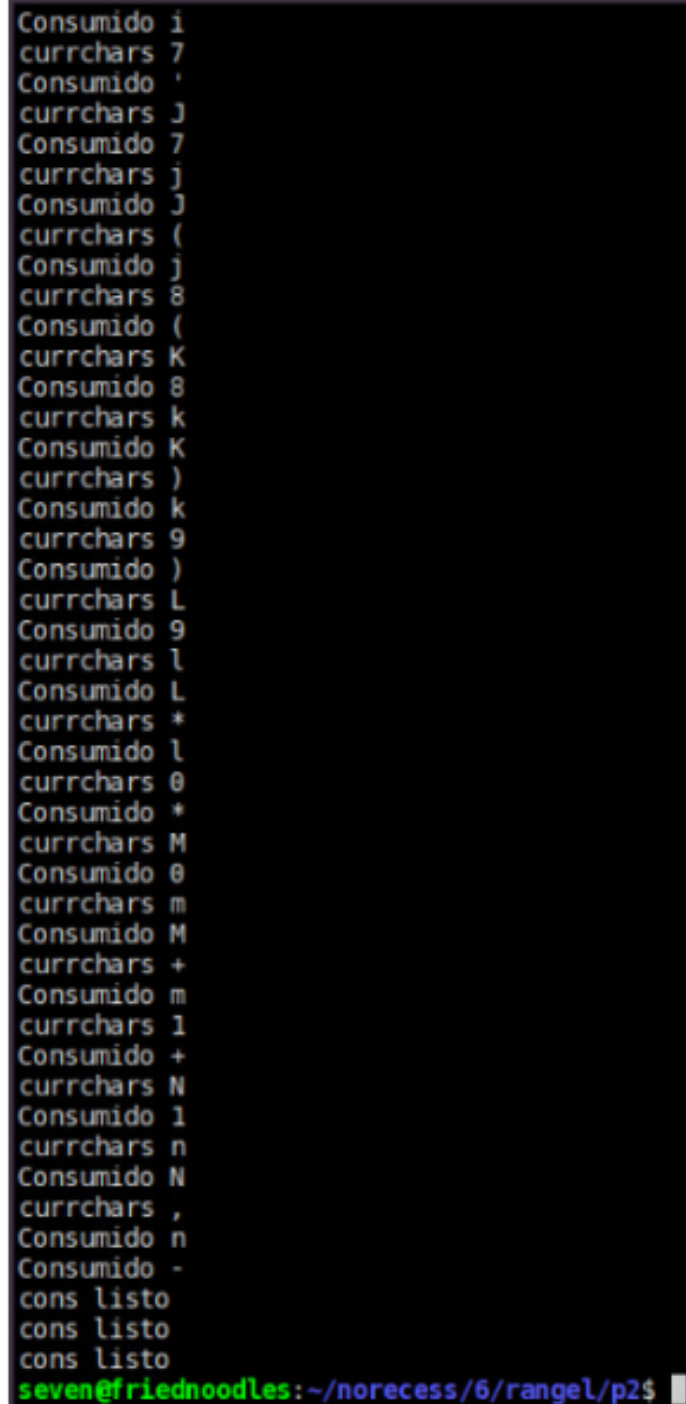
Figura 11: Función del consumidor

En la *figura 11* es posible observar la implementación de la función del consumidor, la cual primeramente muestra el identificador de cada uno de los consumidores, para posteriormente realizar un ciclo while que no se detendrá mientras el contador sea menor

al número de producciones por el número de productores, luego se decremente el semáforo uno.

Como segundo paso se itera el número de secciones críticas con el fin de preguntar si se encuentra disponible y poder consumir los datos.

4. Salida del programa.

A terminal window with a black background and orange text. The output shows a sequence of 'Consumido' and 'currchars' messages for characters i through n, followed by three 'cons listo' messages. The prompt at the bottom is 'seven@friednoodles:~/norecess/6/rangel/p2\$' in green.

```
Consumido i
currchars 7
Consumido '
currchars J
Consumido 7
currchars j
Consumido J
currchars (
Consumido j
currchars 8
Consumido (
currchars K
Consumido 8
currchars k
Consumido K
currchars )
Consumido k
currchars 9
Consumido )
currchars L
Consumido 9
currchars l
Consumido L
currchars *
Consumido l
currchars 0
Consumido *
currchars M
Consumido 0
currchars m
Consumido M
currchars +
Consumido m
currchars 1
Consumido +
currchars N
Consumido 1
currchars n
Consumido N
currchars ,
Consumido n
Consumido -
cons listo
cons listo
cons listo
seven@friednoodles:~/norecess/6/rangel/p2$
```

Figura 12: Salida del programa

5. Conclusiones.

Hernández Castellanos César Uriel

En conclusión los semáforos son una herramienta de sincronización y al mismo tiempo una variable protegida, esta solo puede ser modificada por la rutina inicialización. Son una herramienta que proporciona el programador que asegura la exclusión entre hilos concurrentes para acceder a un recurso compartido.

En el presente documento se trata el problema del productor consumidor el cual es un ejemplo clásico de problema de sincronización de multihilos, el cual consiste en que el productor no añada más productos que la capacidad del buffer y que el consumidor no intente tomar un producto si el buffer está vacío.

La idea para dar solución a este problema es la siguiente, ambos hilos se ejecutan simultáneamente y se "despiertan." "duermen" según sea el estado. De manera concreta el productor añade productos mientras quede espacio y en el momento en que se llene el buffer se pone a "dormir", ya cuando el consumidor toma un producto notifica al productor que puede empezar a trabajar nuevamente, en otro caso si el buffer se vacía, el consumidor pasará a dormir y en el momento que el productor agrega un producto crea una señal para despertarlo, para esto se emplean mecanismos de comunicación para el caso particular de la práctica se usaron semáforos.

Las dificultades que se presentaron en la práctica fue que al inicio no se entendía del todo bien, pero se logró solucionarlo documentándonos acerca del tema. El segundo problema se presentó cuando el número de consumidores aumentaba y otro problema fue la sincronización de los archivos.

Finalmente el problema pudo ser generalizado para múltiples consumidores y productores

Martínez Islas Mauricio Joel

El problema productor consumidor me dejó observar lo problemático que puede llegar a ser la programación concurrente.

¿A qué se parece la programación concurrente? La vida. El todo. El vivir. El ser. Esto nos lleva a algo tan simple como una taza de café. Gotas puras de una cosecha fresca que nos pueden animar y distraer de las dificultades del día a día.

Vivir por lo sencillo es algo que debemos perseguir en todos los días. Se puede extender a todo concepto que ha llegado a tocar la humanidad. Incluso la programación.

Pimentel González Carlos

Con esta práctica aprendimos lo importante que es saber sincronizar y utilizar de manera correcta los semáforos. A veces tener errores de lógica, puede ocasionar que los semáforos queden abiertos cuando no deben, por eso es importante saber colocar exactamente las instrucciones en el orden concreto.

Con la perfecta sincronización de productores y consumidores logramos crear el sistema. Con esto nos damos cuenta que el sistema operativo y las aplicaciones que hacen uso

de zonas críticas, deben de estar minuciosamente hechos para mantener esta sincronización sin romper el todo el sistema.