Proyecto 1: Consumidor-Productor-SharedMemory

Kevin Arce Sánchez, Francisco Murillo Morgan, Ricardo Molina Robles
kevin99fc@estudiantec.cr rjmolinazx@gmail.com
fmurillom@gmail.com
Área académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

I. Introducción:

En el presente documento se resolverá el problema ya conocido del consumidor y productor, el cual es un clásico problema de conflictos en la rama de los sistemas operativos. La idea de este proyecto es crear un buffer, el cual se define como un espacio en memoria en el que se almacenan datos [1], que funcione como un buzón de manera circular y de tamaño finito. Este buffer debe de inicializarse de manera independiente del consumidor y productor, brindándole además un nombre y el tamaño del mismo por medio de entradas por consola. A este buffer se le unirán consumidores y productores, de manera que los productores crearán mensajes y los consumidores los leerán. Cada uno de estos se inicializa de manera independiente y brindándoles el nombre del buffer al cual deseen unirse. Existe un programa llamado inicializador, el cual se encarga de crear el buffer y de inicializar las variables que serán utilizadas para solucionar todo el problema como el semáforo, máximo de consumidores y de productores y la bandera para cancelar todo el sistema de procesos.

II. Ambiente de desarrollo:

Para poder desarrollar el código en C, se utilizó el compilador GCC[2] y la herramienta de edición Visual Studio Code[3]. Esto permite editar y compilar los archivos en formato c, además, el código fue desarrolado y probado en ambientes Linux.

II-A. Incializador:

Para la implementación del inicializador, este primero consultará al usuario cual es el nombre del buffer que desea crear, así como la cantidad de mensajes que se desea que el buffer almacene. Utilizando los datos introducidos anteriormente, el inicializador procederá a reservar los espacios en memoria compartida para el buffer, el semáforo del buffer, las variables globales, y el semáforo para las variables globales. Una vez finalizado, se le indica al usuario la creación exitosa del buffer.

II-B. Productor:

Para la implementación del productor, primero se debe cargar al espacio de memoria del proceso, las direcciones de memoria compartidas del buffer para que este sea capaz de escribir mensajes en el buffer. Además de esto, también se debe cargar las direcciones de memoria correspondientes a los semáforos para el buffer y las variables globales. Una vez que se hallan cargado todas las direcciones de memoria necesarias, se procede a verificar que no se este excediendo el numero máximo de consumidores que puede soportar el buffer. De no estar excediendo el numero máximo de productores, se procede a disminuir la cantidad global de maximo de productores que se pueden crear, y luego se procede a aumentar el numero de productores que se encuentran activos, asi como aumentar también el historial de la cantidad de productores que fueron creados. Una vez que se ha terminado con el proceso de inicialización, el productor entrara en su ciclo de ejecución. En este ciclo, los productores primero verificarán, utilizando el semaforo del buffer, si este se encuentra en uso, y además si no se encuentra en uso, verificará si es su turno de utilizarlo. Si el buffer se encuentra en uso por otro proceso, el productor procede a suspenderse por un tiempo definido por una distribución exponencial, y vuelve a solicitar acceso al buffer. En el caso de que otro proceso no se encuente utilizando el buffer, y sea el turno respectivo de este productor, el productor procedera a marcar el uso del buffer en el semáforo, luego buscara si existe algun espacio vacío en el buffer, de encontrar un espacio vacío, procederá a cargar el mensaje en el buffer, y luego de esto marcará que ya el buffer no se encuentra en uso. Luego de verficar el estado del uso dle buffer, este procederá a revisar el estado del uso de las variables globales. Luego de esto, verificará si se ha activado la bandera para la terminación de los procesos. En caso de que esta bandera se encuentre activa, el productor terminará su eejcución, y disminuirá la cantidad de productores activos. Para el caso que el productor ni encuentre un espacio vació en el buffer, este procederá a suspenderse, hasta que encuentre un nuevo espacio en el buffer.

II-C. Consumidor:

Para la implementación del consumidor se realiza primeramente la configuración necesaria para obtener acceso a la memoria del buffer, las variables iniciales y el semáforo, antes de agregarse como consumidor al buffer comprueba que este tenga espacio en el número máximo de consumidores y si no es así espera a que se libere un espacio, este tiempo de espera se realiza de la misma manera que se realiza el tiempo de espera del consumidor para tratar de leer nuevamente. Luego se utiliza un loop infinito para las funciones del consumidor en el cual primeramente se comprueban dos cosas, que la bandera del semáforo permita entrar a leer mensajes y que el siguiente proceso sea igual al process id del proceso consumidor que

desea entrar a leer mensajes, si es así realiza una verificación para comprobar que existen mensajes para leer, si no existen simplemente avanza el index del siguiente proceso en el semáforo y libera el semáforo. Si existen mensajes por leer el consumidor verifica que el número mágico del mensaje a leer sea igual al módulo 6 del process id del proceso consumidor, si esta condición se cumple el consumidor realiza una serie de impresiones en consola que muestran la razón de la finalización de proceso y una serie de estadísticas de este como tiempos y cantidad de mensajes leídos, una vez realiza esto el proceso termina. Si el mensaje no tiene el valor del número mágico igual al módulo 6 del process id, el consumidor únicamente lee el mensaje, libera la bandera y aumenta el index que el semáforo continue de manera circular. El proceso de espera para volver a intentar entrar al semáforo lo realiza de la siguiente manera: primero crea dos hilos, uno para esperar a que se precione enter y así volver a intentar entrar al semáforo y otro hilo para llevar un tiempo de espera que es igual a un número random con distribución de Poisson generado, es decir, ya sea que pase el tiempo o que se precione enter, por cualquiera de esas dos razones el consumidor volverá a intentar entrar al semáforo.

II-D. Finalizador:

Para la implementación del finalizador, primero se deben cargar los espacios en memoria de todas las variables compartidas entre procesos, Luego de esto, primero se debe activar la bandera de finalización de los proccesos. Una vez que se ha activado la bandera, el finalizador procede a esperar a que el numero de productores y consumidores activos se disminuya a cero. Una vez que se cumpla esta condición, el finalizador procede a liberar los espacios de memoria compartida, y por último muestra en pantalla un acumulado de tiempos de ejecución ligado a todos los productores y consumidores utilizados.

III. Detalles del diseño del programa desarrollado,del software:

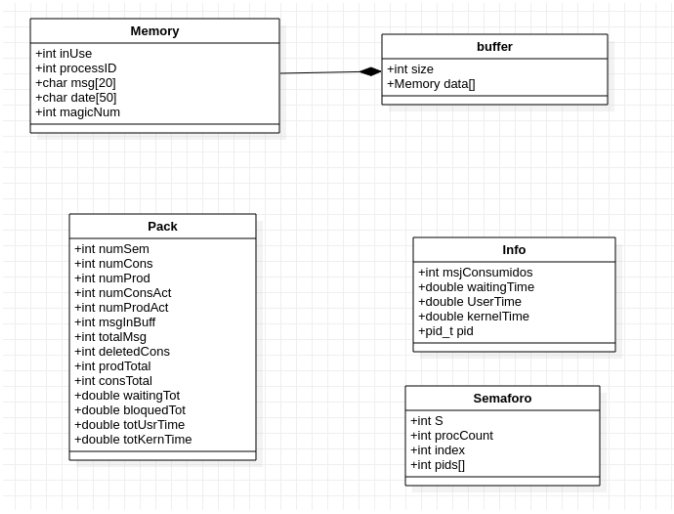

Figura 1. Diagrama de Clases del Sistema

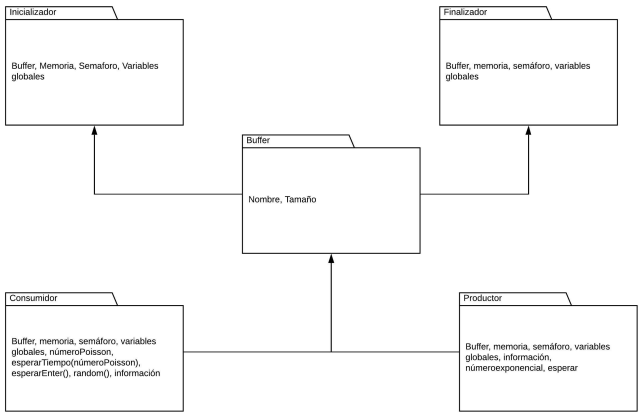

Figura 2. Diagrama de Paquetes

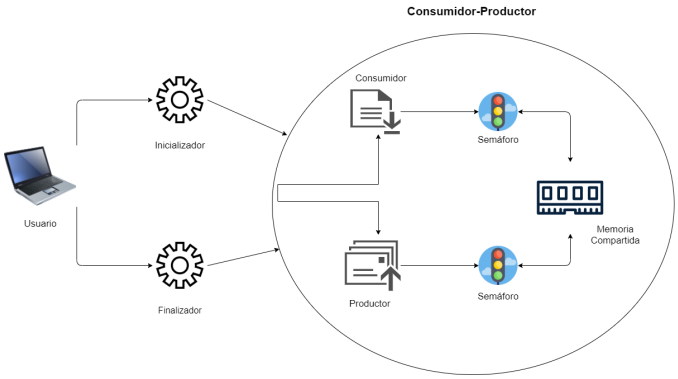

Figura 3. Diagrama de Arquitectura

IV. Manual de instrucción:

Par la utilización del programa, primero se debe descargar el código fuente, y ejecutar el comando ”make” para que este realice la compilación de los programas. Al realizar la compilación, se creará una carpeta ”excecutables” en el cual se podrán encontrar los programas compilados y listos para ser ejecutados. Los paso para la ejecución de los programas serán:

- Ejecutar inicializador: Se debe utilizar el comando ./inicializador, al ejecutarse se le solicitará al usuario que inserte el nombre del buffer que desea crear, y luego de esto se le solicitará que inserte el numero de mensajes que desea que se guarden en el buffer. El inicializdor mostrará en consola cuando este haya terminado de crear el buffer con el tamaño solicitado.
- Ejecutar productor: Se debe utilizar el comando ./producer [nombre del buffer] [media en segundos para la distribución exponencial] para comenzar con la ejecución de un productor. Al ejecutarse este comando, se mostrará en consola en color amarillo la cantidad de tiempo de espera que tendrá el productor, y en verde se mostrará cuando este inserte un mensaje en el buffer, así como el índice en donde inserto el mensaje. Si se desea crear otro productor, basta con ejecutar el comando ./producer [nombre del buffer] [media en segundos para la distribu-

ción exponencial] nuevamente en una nueva ventana de terminal.

- Ejecutar consumidor: Se debe utilizar el comando ./consumer [nombre del buffer] [media en segundos para la distribución de poisson] [modo de operación] para realizar la ejecución del consumidor. El modo de operación puede ser o 0 o 1. Si se seleciona el 0, el consumidor consumira un mensaje cada vez que se precione la tecla enter en el teclado. En el caso de utilizar la opcion 1, el consumidor consumira un mensaje con un tiempo de espera determinado por una distribución de Poisson. En caso de que se desee otro consumidor, baastará con volver a ejecutar el comando ./consumer [nombre del buffer] [media en segundos para la distribución de poisson] [modo de operación] en una nueva ventana de terminal. Si se llega a insertar en la opción de modo un número mayor a uno el modo de operación será el 1.
- Ejecutar finalizador: Una vez que se desea finalizar la ejecución del sistema, se debe ejecutar el comando ./terminator [nombre del buffer]. El programa procederá a activar la bandera de finalización, y los productores y consumidores mostrarán en consola información general de tiempo de ejecución de cada uno. Además el finalizador en pantalla mostrará estadísticas generales de tiempos de ejecución por parte de todos los consumidores y productores.

V. Tabla de actividades por cada estudiante:

Bitácora de Ricardo		
Actividad	Fecha	Horas invertidas
Investigación de la memoria compartida	20/6/2020	5 horas
Diseño del inicializador	21/6/2020	5 horas
Implementación de variables globales	23/6/2020	3 horas
Verificación de información y desarrollo del finalizador	27/6/2020	6 horas
Documentación	27/6/2020	6 horas
Horas totales		25 horas

Bitácora de Francisco		
Actividad	Fecha	Horas invertidas
Investigación del uso de semáforos	21/6/2020	2 horas
Investigación de la distribución exponencial	22/6/2020	2 horas
Implementación del productor	7/6/2020	7 horas
Integración productor-consumidor	23/6/2020	4 horas
Documentación	27/6/2020	6 horas
Horas totales		20 horas

Bitácora de Kevin		
Actividad	Fecha	Horas invertidas
Investigación de funcionamiento y desarrollo del semáforo	21/6/2020	2 horas
Implementación base del consumidor	23/6/2020	4 horas
Desarrollo de la conexión consumidor-buffer	24/6/2020	3 horas
Implementación de numero aleatorio de poisson y espera de tecla enter para el consumidor	25/6/2020	5 horas
Añadiendo variables globales y sprites al consumidor	26/6/2020	3 horas
Finalización del consumidor y pruebas finales junto con el productor, buffer y finalizador	27/6/2020	5 horas
Documentación	27/6/2020	4 horas
Horas totales		26 horas

VI. Conclusiones

Gracias a la implementación de semáforos, se entiende la dinámica detrás del proceso de sincronización para procesos sobre un recurso compartido entre ellos. Además de la responsabilidad que conlleva administrar ese acceso de manera eficiente.

Se concluye que el desarrollo de la aplicación requiere una curva de aprendizaje en la cual se abarca el flujo de datos de varios procesos sobre un solo recurso compartido, la configuración del recurso compartido y la dinámica de semáforos antes de comenzar con el desarrollo del código.

VII. Sugerencias y recomendaciones.

Durante el desarrollo de la solución del problema, se encontró una gran cantidad de código repetitivo en múltiples partes. Ante esta situación se recomienda la creación de bibliotecas personalizadas de tal manera que la presencia de esas secciones se reduzca a un solo archivo.

Referencias

[1] C. Gestor, "Buffer, memoria temporal", Ciset.es, 2020. Disponible en: https://www.ciset.es/glosario/417-buffer. [Accessed: 26- Jun- 2020]

[2] "GCC, the GNU Compiler Collection- GNU Project - Free Software Foundation (FSF)", Gcc.gnu.org, 2020. Disponible en: https://gcc.gnu.org/.

[3] V. Code, "Visual Studio Code - Code Editing. Redefined", Code.visualstudio.com, 2020. Disponible en: https://code.visualstudio.com/.