Productor-Consumidor

David Román Muñoz, Juan Esquivel Rojas, Gerardo Zeledón Martínez
romux09@gmail.com, juanesro1012@gmail.com,gerardo.zeledon14@gmail.com
Área académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—En este documento se mostrará el desarrollo y funcionamiento de un sistema que hará uso de una memoria compartida, la cual será accedida por procesos productores encargados de escribirla y por procesos consumidores encargados de leer los mensajes escritos.**

***Palabras clave*—Share Memory, Semaphores, Buffer, Process**

I. Introducción

En el mundo de la computación se puede dar el caso de que dos procesos completamente independientes puedan compartir datos por medio de mensajes haciendo uso de un espacio de direccionamiento por aparte del que ya ellos ya poseen, a este espacio se le puede llamar memoria compartida. Como esta memoria puede ser accedida por dos procesos simultáneamente debe existir algún protocolo que controle dicho acceso, a este protocolo se le llama sincronización de procesos. En pocas palabras la sincronización de procesos es controlar la transmisión y recepción de señales que tiene por objeto llevar a cabo el trabajo de un grupo de procesos cooperativos.

II. Ambiente de Desarrollo

A continuación se detallan las herramientas utilizadas para el desarrollo e implementación del Sistema Productor - Consumidor.

II-A. GCC y Make

Debido a que uno de los requerimientos del proyecto es que debe ser desarrollado en el lenguaje de programación C, se usará el compilador gcc para compilar tanto el código fuente de los diferentes procesos, como las diferentes bibliotecas personalizadas que serán desarrolladas para la simplificar el desarrollo del problema. Como el proyecto es bastante robusto, el paquete make facilita la compilación haciendo uso de diferentes makefiles dedicados a compilar cada una de las bibliotecas y archivos con el código fuente con un solo comando.

II-B. Visual Studio Code

Este es un editor de código fuente desarrollado por Microsoft para Windows , Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias.

Además, para el desarrollo de esta aplicación, se utilizó un plugin llamado Live Share. Este paquete extra permite la programación dentro de VS code de varias personas de manera simultánea. ESto permite programar a todos los miembros del grupo en un mismo proyecto, lo cual ayuda a llevar un mejor control del repositorio del proyecto.

III. Detalles del diseño del programa desarrollado

Como parte de la solución al problema del productor-consumidor por desarrollar, se diseñan una serie de diagramas que facilitan un mejor entendimiento y desarrollo de la solución por implementar.

Estos diagramas se diseñan teniendo en cuenta el formato básico UML, en conjunto con un formato de diseño modular en diversas capas de abstracción para la solución a desarrollar. Por lo tanto, para este caso se diseñan el diagrama de arquitectura del sistema, diagrama de paquetes y de dominio. A continuación se adjuntan dichos diagramas.

III-A. Diagrama de arquitectura

Como arquitectura general del sistema, se muestra el uso de un inicializar y un finalizador, los cuales son los encargados de iniciar y finalizar respectivamente, el funcionamiento del programa. El inicializador es el encargado de validar todos los parámetros de inicio, se encarga de la creación del búffer a utilizar como abstracción de la memoria compartida, e iniciar las variables y demás elementos necesarios para el funcionamiento del sistema. Y el finalizador termina con el funcionamiento del sistema y todos los procesos involucrados.

Y en cuanto al sistema del productor-consumidor como tal, se muestra el uso de una memoria compartida, productores y consumidores, y los semáforos los cuales verifican el tráfico de los datos de una entidad a otra.

A continuación, en la Figura 1 se observa el Diagrama de Arquitectura empleado para la solución del proyecto.

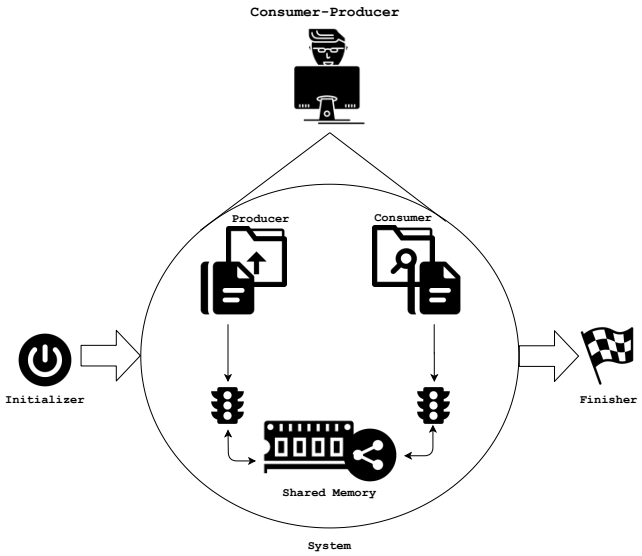

Figura 1: Diagrama de arquitectura del sistema.

III-B. ***Diagrama de paquetes***

Seguidamente, en la Figura 2 se observa el diagrama de paquetes empleado para la solución del proyecto.

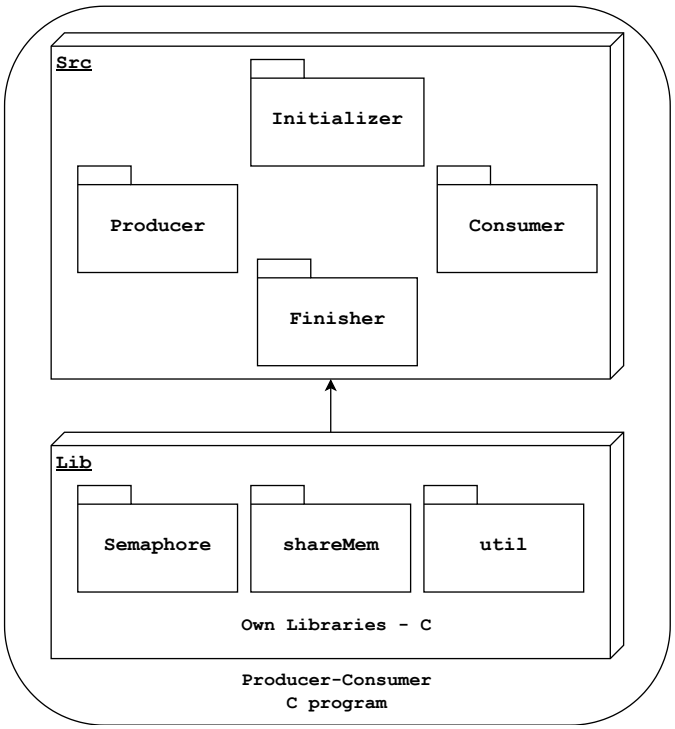

Figura 2: Diagrama de paquetes del sistema.

En este diagrama se presentan todos los paquetes utilizados para la solución del sistema: En el bloque lib, se encuentran las bibliotecas creadas para el funcionamiento de cada entidad dentro del sistema. La biblioteca semáforo es la encargada de controlar los dichos semáforos utilizados para el funcionamiento del proyecto, la biblioteca shareMem es la encargada de cumplir con las funciones necesaria para el tratamiento de la memoria compartida y la util es la contenedora de todas las funciones auxiliares utilizadas a lo largo del sistema y las distribuciones de probabilidad lineales utilizadas (De Poisson y Exponencial).

Mientras que en el bloque Source, se encuentran los paquetes de los entes encargados del sistema: Inicializador, finalizador, produtor y consumidor.

III-C. ***Diagrama de Dominio***

En el Diagrama de la figura 3 se muestra como van a interactuar cada uno de los procesos entre si, el finalizador al ser el encargado de terminar todos los procesos debe interactuar con todos pero por otro lado el inicializador solo interactúa con el semáforo y la memoria porque los productores y consumidores se inician independientemente

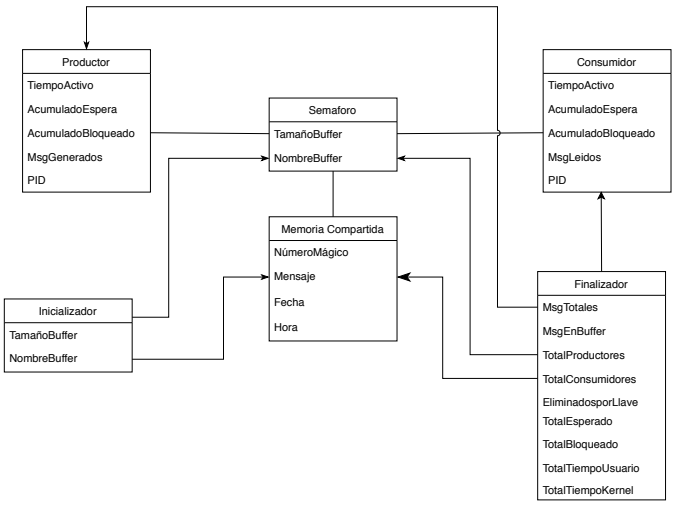

Figura 3: Diagrama de Dominio del sistema.

IV. RESULTADOS

IV-A. ***Inicializador***

A continuación, en la imagen de la figura 4 se puede observar inicializador en funcionamiento, el cual tiene como objetivo aceptar los parámetros iniciales para la creación de los recursos y crear todas las variables globales así como los semáforos. También, este modulo se encarga de inicializar todos los espacios de memoria para evitar que se lean datos no deseados.

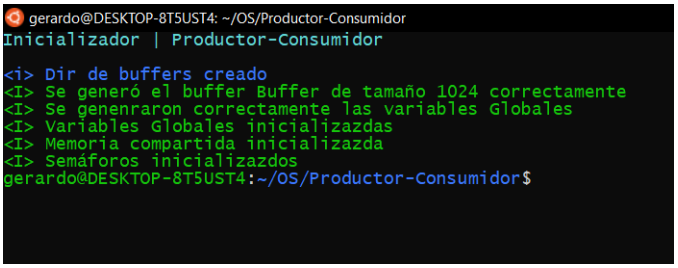

Figura 4: Inicializador del sistema

IV-B. ***Productor***

El productor se encarga de escribir mensajes en el buffer definido en el inicializador. Dichos mensajes almacenan la fecha, hora, número mágico y una palabra aleatoria. El acceso a la memoria para escribir los datos sera controlada por medio de un semáforo el cual se encargara de velar porque sin importar el número de productores que estén activos simultáneamente, solo uno tenga acceso a la memoria. En la imagen de la figura 5 se observa dicho funcionamiento.

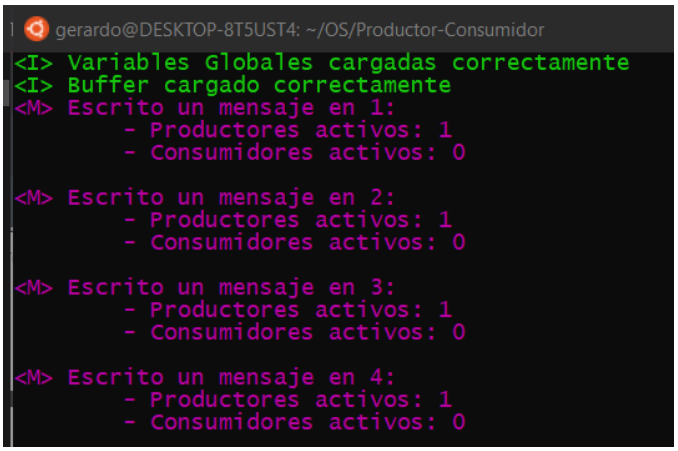

Figura 5: Productor del sistema

IV-C. ***Consumidor***

El consumidor se encarga de leer mensajes en el buffer y extraerlos de memoria, los cuales fueron escritos por algún productor. Si se le aplica la operacion módulo al identificador de proceso del consumidor y este coincide con el número mágico que posee el mensaje leído se activa una rutina de finalización, de no ser así, se muestra de manera amigable el mensaje leído. En la imagen de la figura 6 se observa dicho funcionamiento.

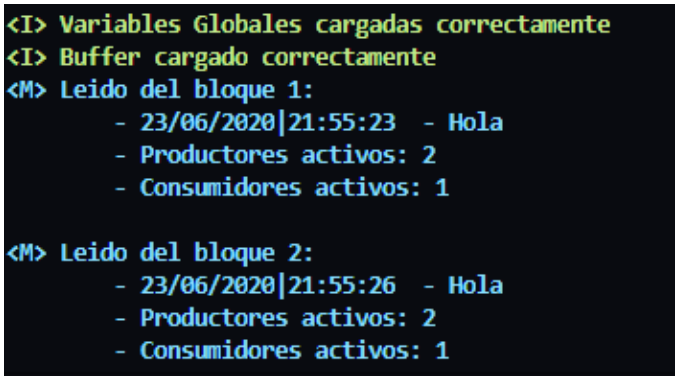

Figura 6: Consumidor del sistema

IV-D. ***Finalizador***

Por último se tiene al finalizador que tiene como objetivo cancelar todo el sistema de procesos y mostrar de manera elegante todas las variables globales con las estadísticas de los productores y consumidores.Para finalizarlo primero se cargan las variables globales y por medio de una señal se le indica primero a los consumidores que deben cerrarse y luego se le indica a lo productores. Cuando se finalizan todos libera la memoria compartida y los semaforos para dar por finalizado el sistema.

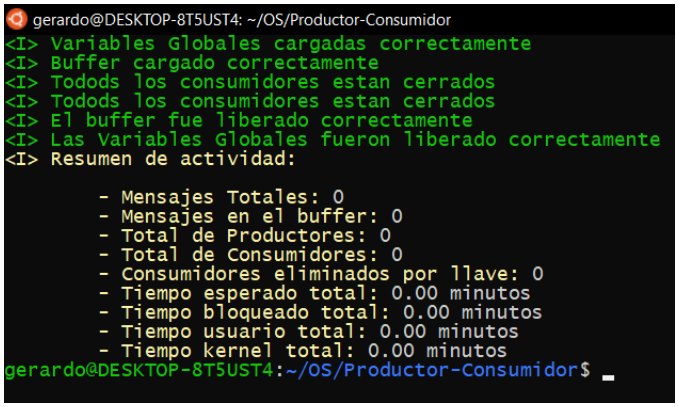

Figura 7: Finalizador del sistema

V. INSTRUCCIONES DE COMO SE UTILIZA EL PROYECTO

Antes de empezar a utilizar el sistema, es necesario el compilar todos los archivos necesarios dentro del mismo. Por lo tanto, para compilar el programa es sólo de ejecutar el comando:

```
Make
```

Para ejecutar dicho comando se requiere hacerlo dentro de la carpeta del proyecto. Por ejemplo: juand@juand-Inspiron-5458: /CE-TEC/Semestre9/SO/Proyecto1/Productor-Consumidor.

A continuación se muestra como ejecutar y utilizar cada uno de las partes correctamente:

1. **Inicializador:** Para ejecutar el inicializador es necesario estar dentro de la carpeta bin del proyecto y ejecutar el comando:
   ./incializador (Nombre del Búffer) (Tamaño del Búffer)

2. **Productor:** De la misma forma que el inicializador, para poner en funcionamiento el productor, además de estar en la ubicación de la carpeta bin, se requiere el comando:
   ./productor (Nombre del Búffer) (Media de tiempos)

3. **Consumidor:** Al igual que el resto de archivos, se requiere estar en la carpeta bin y ejecutar el comando:
   ./consumidor (Nombre del búffer) (Media de tiempos) (Modo de operación)

4. **Finalizador:** Para el caso del finalizador, este se puede ejecutar en cualquier momento ya que al ejecutarse el sistema se termina inmediatamente. Para ejecutar este ente del proyecto es necesario estar ubicado dentro de la carpeta bin y ejecutar el siguiente comando:
   ./finalizador (Nombre del búffer)

VI. BITÁCORA: TABLA DE ACTIVIDADES POR CADA ESTUDIANTE

Fecha	Descripción
19/06/2020	Obtención de requerimientos División de trabajo
20/06/2020	Investigación sobre la memoria compartida Inicio de biblioteca dedicada a memoria
21/06/2020	Finalización de la biblioteca Realización de pruebas.
22/06/2020	Creación del productor y consumidor Realización de pruebas
23/06/2020	Corrección de errores mínimos

Cuadro I: Bitácora Gerardo Zeledón Martínez.

Fecha	Descripción
19/06/2020	Obtención de requerimientos División de trabajo
20/06/2020	Investigación sobre el manejo de semáforos Creación del inicializador
21/06/2020	Implementación de biblioteca de semáforos Pruebas con semáforos.
22/06/2020	Integración de productor y consumidor Realización de pruebas
23/06/2020	Corrección de errores mínimos

Cuadro II: Bitácora David Román Muñoz.

Fecha	Descripción
19/06/2020	Obtención de requerimientos División de trabajo
21/06/2020	Investigación e implementación de distribuciones de probabilidad lineales
21/06/2020	Análisis y pruebas con la biblioteca de semáforos creada
22/06/2020	Creación del finalizador y realización de pruebas de funcionamiento.
23/06/2020	Corrección de errores mínimos Inicio de la documentación del proyecto
24/06/2020	Diseño de diagramas del sistema Creación de diagrama de arquitectura y paquetes

Cuadro III: Bitácora Juan D. Esquivel Rojas.

VII. CONCLUSIONES

El lenguaje C, es sumamente útil en proyectos que requiere una abstracción de bajo nivel (o intermedio), por el hecho de que permite definir porciones de memoria, la cual puede compartirse y ser utilizada por múltiples procesos.

Los semáforos, permiten sincronizar el acceso a memoria y de esta manera se evita el implementar busy waiting, por lo que genera una mejora considerable en rendimiento y previene problemas como lo es el segmentation fault.

VIII. SUGERENCIAS O RECOMENDACIONES

Una forma fácil de probar el funcionamiento correcto de los semáforos consiste en tener mínimo 2 procesos y colocar un ”sleep.ªntes de poner en alto un semáforo, de esta manera se puede observar si funciona el semáforo implementado.

REFERENCIAS

[1] Üse C language to generate Poisson distribution random number instance source code”, Topic.alibabacloud.com, 2020. [Online]. Available: https://topic.alibabacloud.com/a/use-c-language-to-generate-poisson-distribution-random-number-instance-source-code$_{1}3_{1}2$0012031.$html$.[$Accessed$: 21 − Jun − 2020].

[2] G. distribution and W. Vane, ”Generating random numbers of exponential distribution”, Stack Overflow, 2020. [Online]. Available: https://stackoverflow.com/questions/34558230/generating-random-numbers-of-exponential-distribution. [Accessed: 25- Jun- 2020].

[3] S. Mitra and I. Bangalore, ”Producer-consumer Problem Solution in C using Semaphore and Mutex”, Cricket,Coding and Life, 2020. [Online]. Available: https://shivammitra.com/c/producer-consumer-problem-in-c/. [Accessed: 18- Jun- 2020].

[4] ”Problema del productor-consumidor”. www2.infor.uva.es, 2020. [Online]. Available: https://www2.infor.uva.es/ cllamas/concurr/-pract97/rsantos/index.html. [Accessed: 18- Jun- 2020].