Proyecto 1: Semáforos en buzón circular

Emanuel Esquivel López, Roger Valderrama
ema11412@estudiantec.cr, roger.andres18@estudiantec.cr
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

I. Introducción

En el mundo informático en especial los sistemas operativos siempre existe un problema muy común presente en todos los sistemas el cual es el productor/consumidor, este problema se plantea ya que en muchos casos hay procesos que se encargan de producir en memoria mensajes o cualquier elemento almacenable, además de un consumidor que es el encargado de consumir estos mensaje y liberar memoria necesaria presente en el sistema. El problema empieza cuando existen varios consumidores y productores sobre una misma memoria compartida, lo que puede genera caos ya que varios productores pueden escribir en una misma posición del buffer lo que generaría un conflicto por perdida de datos. así como también los consumidores pueden caer en la misma posición a consumir inutilizado los demás perdiendo eficiencia e incoherencia entre datos, por lo que para la resolución de este es necesario un semáforo el cual se encargue de redirigir los consumidores y productores para que se pierda estas incoherencias y datos erroneos.

II. Ambiente de desarrollo

El proyecto fue desarrollado en linux, por lo que mucha de su implementación y elaboración fue con el uso de comandos en la terminal del mismo, esta fue desarrollada en el lenguaje de programación Assembly x86.

Se llevo acabo en una computadora de arquitectura 64 bits, sistema operativo Linux Kali linux 2020.2, 16 Gb memoria, procesador Ryzen 7 2700x, además de la edición de código en VSCode.

III. Atributos

III-A. Herramientas de Ingeniería (HI)

- Desarrollo de un buffer capaz de recibir mensajes por parte de productores y consumir esos mensajes por parte de los consumidores.
- Creación de un semáforo para el manejo de productores y consumidores.
- Desarrollo de makefiles para la compilación de los programas desarrollados.
- Modelado de un sistema de sincronización de uso de recursos compartidos para futuras aplicaciones en la industria.

III-B. Herramientas de Ingeniería (AC)

- Asignación de tareas de acuerdo a experiencia previa en actividades similares o que se deseen aprender.
- Agrupar ideas para generar una solución óptima que incluya las perspectivas de los tres miembros.
- Uso de la herramienta git para el manejo del código fuente de todo el grupo.

IV. Detalles de diseño

La vista general del sistema se aprecia en la figura 1, se aprecia que el usuario es el instanciador de los servicios realizados, como se ve esta compuesto por 5 componentes importantes.

IV-1. Inicializador:: Encargado de crear la memoria compartida, la cual recibe los parámetros del nombre y la memoria total $numero > 0$ el cual es el que se desea que tenga el buffer, además de instanciar los semáforos de accion que se utilizaran, esta información se ve en la figura 2 mas claro.

IV-2. Productor:: Encargado de enviar a memoria mensajes los cuales serán almacenados en el buffer de manera permanente hasta que sean consumidos o la memoria sea eliminada, si hay mas de un productor serán ordenados mediante un semáforo.

IV-3. Consumidor:: Se encarga de consumir los datos que están en la memoria compartida, se puede ver que si hay mas de 1 consumidor de igual manera se tiene un semáforo el cual se encarga de ordenar y tomar acción de quien consume en memoria.

IV-4. Finalizador:: Es el encargado de cerrar los procesos en memoria, eliminando así la memoria compartida y terminando el accionar de los demas procesos.

IV-5. Memoria compartida:: Almacena la información necesaria la cual es creada por los productores, y almacenada y ordenada por los index creados por el inicializador.

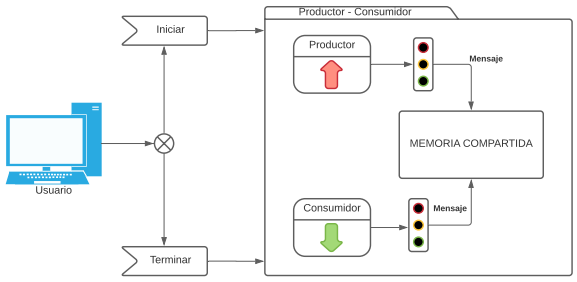

Figura 1. Diagrama de arquitectura del sistema

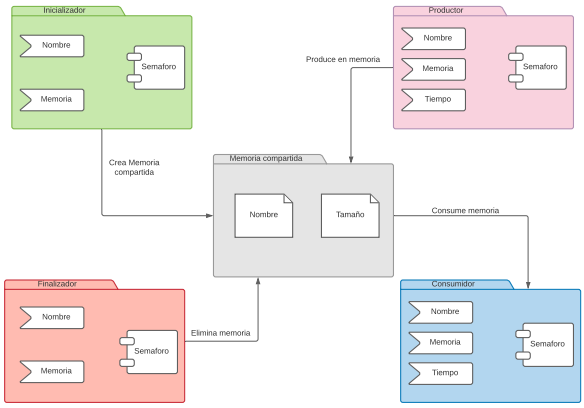

Figura 2. Diagrama de funcionalidades del sistema

V. Instrucciones de uso

V-1. Requisitos previos: Para el uso del sistema se realiza primero la compilación de los archivos para esto es necesario instalar los paquetes y compilador para C/C++ gcc mediante el comando.

```
sudo apt install build-essential -y
```

V-2. Compilación: Con esto podemos hacer make para compilar todos los archivos necesarios para la ejecución del proyecto, estos son guardados en la carpeta bin,

V-3. Ejecución: Para empezar es necesario un servicio **initializer**, el cual se encarga de realizar el buffer correspondiente para los datos, ya que sin este el productor consumidor y finalizador no funcionarían ya que no encontrarían un buffer en el cual operar, todos son llamados con un tamaño de buffer, un nombre, en caso de consumidor ademas un tiempo extra el cual es el de operacion, y el consumidor un modo, automático o manual.

```
/initializer <buffer_size> <buffer_name>
```

```
./producer <buffer_name> <time_medium>
```

```
./consumer <buffer_name> <time_medium>   <
    operation_mode>
```

```
./finalizer <buffer_name>
```

VI. Tabla de actividades

VI-A. *Emanuel Esquivel*

Actividad	Horas
Investigación sobre semáforos	3
Investigación de creación de memoria compartida	2
Implementación del Inicializador	8
Implementación del Productor	13
Documentación	2
Total	28

VI-B. *Roger Valderrama*

Actividad	Horas
Investigación sobre semáforos	1
Investigación de creación de memoria compartida	2
Implementación del Consumidor	15
Implementación del Finalizador	9
Documentación	2
Total	29

VII. Conclusiones

Se concluye que para el desarrollo del proyecto, se necesita amplios conocimientos teóricos sobre la dinámica de semáforos, la configuración requerida para un recurso compartido y sobre el flujo de datos de varios procesos sobre solo un recurso compartido.

Del mismo modo, con la ayuda de dicho desarrollo de semáforos, se profundiza la comprensión de la dinámica por debajo, del proceso de sincronización para procesos que utilizan recursos compartidos entre ellos; siempre considerando administrar los accesos dichos recursos con la mayor eficiencia posible.

VIII. Sugerencias y recomendaciones

Debido a que el mecanismo de sincronización por semáforos es un algoritmo relativamente viejo (creado en 1965), existe mucha documentación sobre su implementación. Por lo que, se sugiere la creación de una biblioteca para tener dicho algoritmo en un lugar, y así facilitar su uso para nuevos sistemas.

Referencias

[1] Generating random numbers of exponential distribution. (s. f.). Stack Overflow. Recuperado 22 de abril de 2021, de https://stackoverflow.com/questions/34558230/generating-random-numbers-of-exponential-distribution

[2] Memoria compartida en C para Linux. (s. f.). chuidiang. Recuperado 20 de abril de 2021, de http://www.chuidiang.org/clinux/ipcs/mem_comp.php

[3] Use C language to generate Poisson distribution random number instance source code. (s. f.). Alibabacloud. Recuperado 22 de abril de 2021, de https://topic.alibabacloud.com/a/use-c-language-to-generate-poisson-distribution-random-number-instance-source-code_1_31_20012031.html