Scheduling Ships: Calendarizador de Hilos

Gabriel Chacón Alfaro, Kendall Martinez Carvajal, Cristopher Moreira Quirós, Abraham Venegas Mayorga
gchacon8@estudiantec.cr, kendallmc@estudiantec.cr, cris.moreira@estudiantec.cr, abravenegas@estudiantec.cr
Área Académica Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—This project's goal is to implement a user-level thread library called CEThreads based on the PThreads library. The main objective is to manage and schedule threads efficiently based on several scheduling algorithms. By simulating a canal (CPU) through which ships of different priorities and characteristics must pass, the project represents a system where resource allocation and task scheduling are crucial. The scheduling policies include Round Robin (RR), Priority, Shortest Job First (SJF), First Come First Serve (FCFS), and Real-Time scheduling. This project will also include a hardware prototype and a graphical user interface to represent the development of the project to better comprehend what is happening. Through the development of this project we were able to prove the importance of the management of threads and the improvement that gives to multitasking and the benefits from it.**

***Palabras clave*—Calendarización, Canal, Hilos, Procesos, PThreads**

I. Introducción

En los sistemas operativos modernos, utilizar los hilos eficientemente es un aspecto importante de su funcionalidad. Los hilos se encargan de ejecutar simultáneamente múltiples tareas, mejorando el rendimiento y la utilización de los recursos del sistema, por lo qué los hilos deben funcionar impecablemente. Nuestro proyecto se centra en la implementación de una biblioteca personalizada de hilos, llamada CEthreads, la cual va a ser diseñada para realizar un programa que simule el paso de barcos a través de un canal en el cual se gestiona el flujo mediante distintos algoritmos de calendarización de hilos. El sistema que se debe crear requiere que los barcos puedan transitar por un canal sin colisiones, utilizando para ello algoritmos como Round Robin (RR), First-Come-First-Served (FCFS), Shortest Job First (SJF), y Prioridad, que se encargar de controlar la cola de barcos que están listos para cruzar el canal. Estos algoritmos optimiizan el uso del recurso del canal de la misma manera en que los sistemas operativos gestionan los recursos del CPU.Por otro lado, el flujo de los barcos en el canal será controlado por tres algortimos: equidad, letrero y tico, que deciden de que forma pueden cruzar los barcos por el canal.

El algoritmo de equidad consiste en establecer un parámetro W (idicado por el usuario) que indica cuantos barcos deben de pasar de cada lado, es decir, se inicia transportando W barcos de izquierda a derecha, después W barcos en sentido contrario. En caso de que alguno de los lados no haya embarcaciones se debe de garantizar el flujo de las mismas, en el lado en el que si hayan, el algoritmo de letrero consiste en un letrero (izquierda - derecha) que indica el sentido de vía del canal, cuando el letrero está en "izquierda" pasan los barcos del lado izquierdo, si estuviera en "derecha" entonces pasan los que están del lado derecho. El letrero cambia cada cierto tiempo, el cual es definido por el usuario al iniciar el programa y, por último, el algoritmo tico donde no hay control de flujo, sin embargo, no puede haber colisiones de barcos, también se debe de garantizar el flujo en caso de que hayan barcos solo de un lado.

Finalmente, este documento se organiza de la siguiente manera: en primer lugar, se explica el ambiente de desarrollo que donde se detalla que se ocupa para ejecutar el proyecto. A continuación, se describen los atributos que se reforzaron durante el desarrollo del proyecto, luego, se realiza un análisis del diseño del proyecto y como se llevo este acabo. Se continua con las instrucciones para la utilización del proyecto. Finalmente, se discuten las conclusiones y se presentan recomendaciones para futuros trabajos.

II. Ambiente de desarrollo

Para este proyecto es necesario utilizar una distribución de linux, para el desarrollo de este proyecto se utilizó y se recomienda Ubuntu, preferiblemente la versión LTS, esto es necesario para poder visualizar los hilos y el manejo de estos por el sistema operativo, para visualizar el código se recomienda un editor de código como Visual Studio Code y su extensión para C y C++ o bien CLion. En su distribución de linux es necesario tener git para poder clonar el repositorio del siguiente link: , para mayor facilidad del usuario se recomienda utilizar GitHub Desktop, se puede instalar con los siguientes comandos:

-sudo apt install wget apt-transport-https gnupg2 software-properties-common

-wget https://github.com/shiftkey/desktop/releases/download /release-3.1.7-linux1/GitHubDesktop-linux-3.1.7-linux1.deb

-sudo apt install -f ./GitHubDesktop-linux-3.1.7-linux1.deb

Y con el siguiente comando abre la aplicación y procede a iniciar sesión en su cuenta.

-githup-desktop

Además, es necesario instalar el IDE de arduino y poseer una placa Arduino UNO conectado a un circuito de la figura 3 conectado mediante una protoboard. Con todas estas herramientas será posible ejecutar el makefile que correrá el programa y deberá proveer la configuración deseada para su evaluación.

III. Atributos

Estrategias para trabajo individual y en equipo equitativo

Desde el inicio del proyecto, se definieron estrategias claras para asegurar una participación equitativa de todo el equipo, de tal forma que se aprovecharan las fortalezas individuales de cada miembro del grupo. Además, se estableció una comunicación abierta y directa en la cual todos los miembros podían expresar sus opiniones e ideas, para contribuir con el desarrollo de la tarea. Por otro lado, para asegurar la equidad y que hubiera verdadera inclusión, se distribuyeron las tareas de forma que todos pudieran trabajar en diferentes aspectos del proyecto, como la lógica del las funciones de CEThreads, los calendarizadores, los algoritmos del canal, los barcos, el hardware y la interfaz gráfica. Finalmente, las reuniones regulares fueron clave para ajustar el trabajo de acuerdo a las necesidades de cada miembro y resolver cualquier obstáculo que surgiera.

Planificación del trabajo

Para el proyecto, en un inicio se plantearon varios problemas principales: las funciones de CEThreads, los calendarizadores, los algoritmos del canal, los barcos, el hardware y finalmente la interfaz gráfica. Luego estos problemas principales se dividieron en tareas más pequeñas bajo la mentalidad de ”divide y vencerás” para ser finalmente divididas de manera equitativa entre ambos.

Acciones promueven la colaboración entre los miembros del equipo

Para fomentar la colaboración, se implementó un entorno de trabajo compartido mediante la herramienta GitHub, además de utilizar un grupo de WhatsApp para facilitar la comunicación y el intercambio de ideas entre todos los miembros del grupo. También se llevaron a cabo reuniones tanto presenciales como virtuales, que permitieron que los miembros del equipo mantuvieran un seguimiento constante del progreso del proyecto y resolvieran problemas de manera conjunta.

Ejecución de las estrategias planificadas para el logro de los objetivos

Las estrategias planificadas se ejecutaron siguiendo un cronograma lo suficientemente flexible para permitir ajustes en caso de que surgieran inconvenientes durante el desarrollo. Cada miembro cumplió con sus responsabilidades asignadas y hubo apoyo mutuo cuando se encontraron dificultades. Además, las reuniones regulares permitieron la revisión de los avances en tiempo real y la corrección de problemas que desviaran el proyecto de lograr los objetivos planteados.

Evaluación del desempeño individual y en equipo

El desempeño de cada individuo se evaluó a través de la revisión de los objetivos alcanzados por cada uno. Además, cada integrante recibió retroalimentación específica sobre su trabajo, lo que permitió implementar cambios y mejoras en los avances que se entregaban. Por otro lado, a nivel de equipo se revisaron las metas alcanzadas y se evaluó la calidad de la integración del código en un solo programa. Finalmente, para estas evaluaciones se utilizaron métricas como el cumplimiento de las fechas de entrega de los avances, la funcionalidad del código y la cantidad de errores encontrados durante las pruebas.

Evaluación para las estrategias utilizadas de equidad e inclusión

Las estrategias de equidad e inclusión se evaluaron a lo largo del proyecto asegurando que todos los miembros tuvieran un espacio equitativo para participar en la toma de decisiones entorno a la ruta del proyecto, así como el desarrollo del mismo. Básicamente se promovió un ambiente donde cualquier miembro podía sugerir mejoras o cambios sin importar si esa área era su responsabilidad.

Evaluación de las acciones de colaboración entre los miembros del equipo

La colaboración entre los miembros fue evaluada tanto en términos de comunicación como en la capacidad de trabajar juntos de manera efectiva. Para esto se tomaron en cuenta aspectos como la puntualidad en las reuniones, la capacidad para aceptar retroalimentación y la disposición a ayudar al compañero en caso de que enfrentara problemas de cualquier tipo.

IV. Diseño del software y hardware

Para el diseño del proyecto es importante recalcar las asociaciones que se realizan como representación del manejo de procesos, en este caso el canal juega el papel del CPU por el que pasan los barcos que representan los diferentes procesos (cada uno es un hilo) y sus respectivas posiciones son recursos computacionales, además, los calendarizadores son los algoritmos que se utilizan en un SO y los algoritmos de flujo son análogos a las políticas de un SO. De esta manera se puede comprender la figura 1

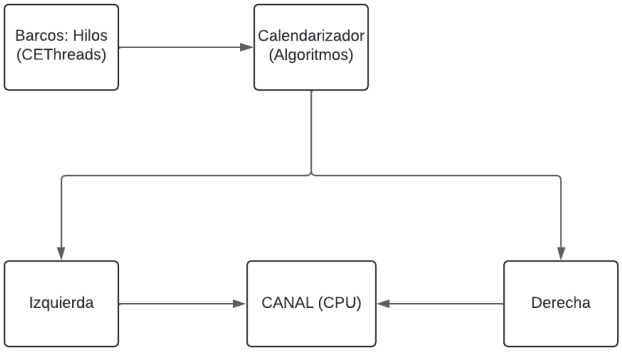

Figura 1. Diagrama de componentes

Para la representación del sistema mediante un circuito (hardware) se muestra a continuación el diagrama del circuito de la figura 2.

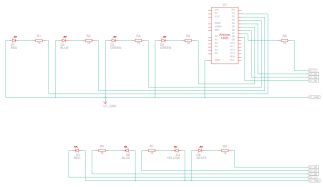

Figura 2. Diagrama del circuito de la lógica del canal

Además, su representación digitalmente, con sus respectivas conexiones en Arduino se muestra en la figura 3. Donde el color de los LED se representan de la siguiente manera: rojo = patrulla, azul = pesquero, verde = normal, amarillo = canal izquierdo y blanco = canal derecho.

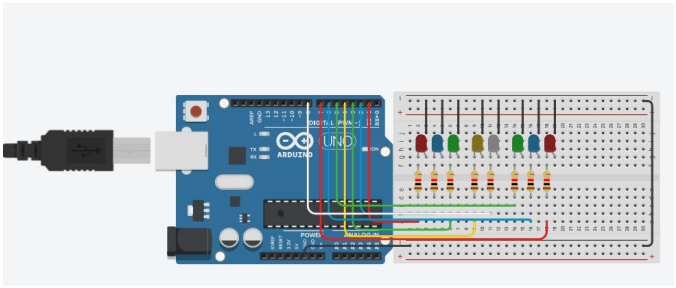

Figura 3. Circuito conectado a Arduino

El diagrama de arquitectura es sencillo, el usuario ingresa la configuración desesada en CEThreads y esto es comunicado de manera serial al arduino para transferirlo mediante sus pines a los pines y representar como se aprecia

Figura 4. Diagrama de arquitectura

El diagrama de secuencias en el caso del hardware se puede apreciar en la figura 5, así mismo, el diagrama de secuencia en el caso de la interfaz gráfica se aprecia en la figura 6

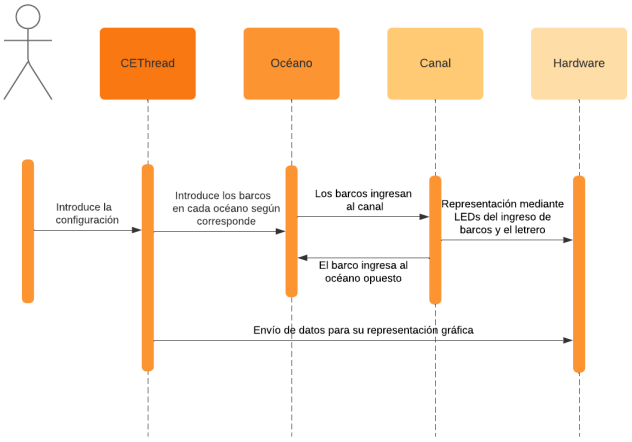

Figura 5. Diagrama de secuencia con representación gráfica mediante hardware

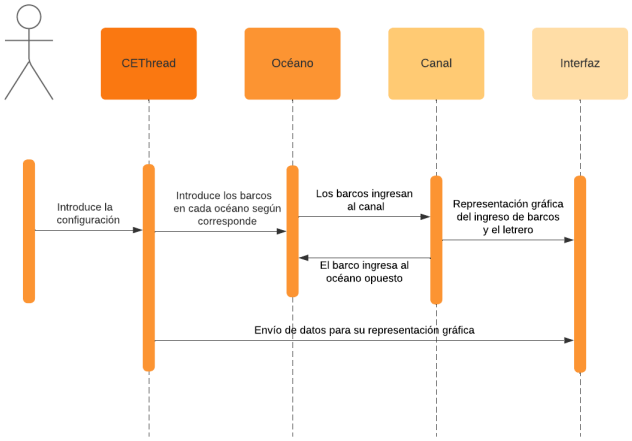

Figura 6. Diagrama de secuencia con representación gráfica mediante interfaz gráfica

A continuación se ve el diagrama de clases UML en el que se incluyen las clases CEthread, CEmutex, Node, Letrero, Tico, Equidad, Scheduler y finalmente TipoBarco, que consituyen a las clases principales que permiten el funcionamiento del programa.

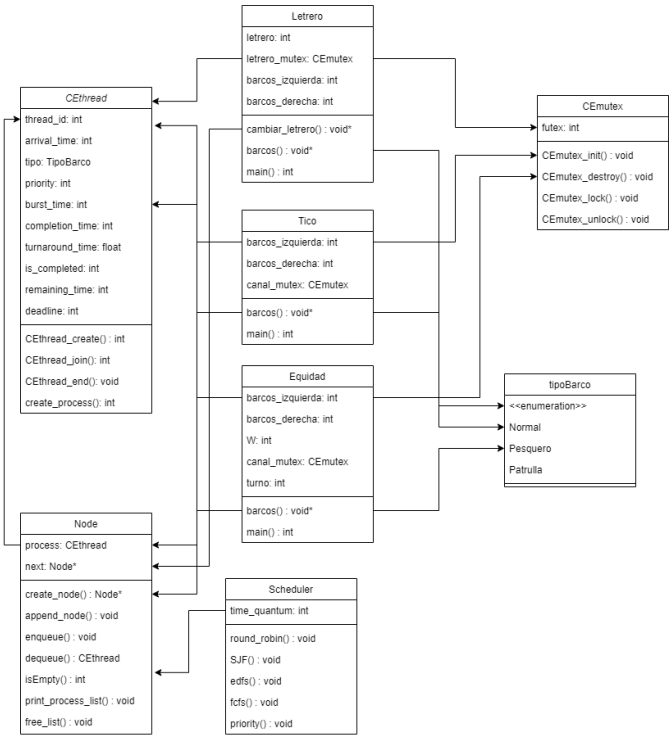

Figura 7. Diagrama de clases para el programa desarrollado

V. Instrucciones

Para este proyecto es necesario hacer uso de este en un entorno de linux con un editor de código como Visual Studio Code o CLion para visualizar el código en c, para la interfaz es necesario qt. Con el makefile se puede correr el programa y en consola se indican los barcos deseados en cada costado.

VI. Conclusiones

1. Se gestionó de manera exitosa múltiples hilos de manera eficiente mediante el uso de algoritmos de planificación.
2. Se elaboró de manera eficiente la librería CEThreads.
3. El diseño adecuado de los algoritmos de flujo es vital para sacar provecho al manejo simultáneo de hilos

VII. Recomendaciones

1. Sería útil realizar pruebas adicionales en entornos con más procesos para representar situaciones reales de alta concurrencia y cargas variables.
2. Para mejorar la usabilidad de la librería se recomienda añadir mecanismos de control de errores y seguridad en la biblioteca para garantizar la integridad de los hilos y evitar condiciones de carrera o bloqueos.
3. Es importante tener especial detalle en los algoritmos de flujo que son vitales para potenciar el rendimiento del manejo de hilos simultáneos.

Referencias

[1] Tanenbaum, A. S.y Bos, H. Modern operating systems. Pearson, 2022.