Proyecto 1: Scheduling Ships

Ignacio José Carazo Nieto, Luis Fabián Crawford Barquero, Saul Gómez Rámirez, Kevin Rodríguez Lanuza
email:
nachocarazo18@estudiantec.cr,fabian152195@estudiantec.cr,saulgora@estudiantec.cr,kjrodriguez@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—Este proyecto tuvo como objetivo crear una biblioteca de hilos (CEThreads) que implementa varios algoritmos de calendarización para manejar múltiples hilos que utilizan un recurso. Desarrollamos y probamos la biblioteca en un entorno Linux, logrando una gestión eficiente de hilos y recursos. La principal contribución es la exitosa calendarización de hilos utilizando hardware para su visualización.**

***Palabras clave*—Hilos, Calendarización, Mutex.**

I. Introducción

En los sistemas operativos, la correcta gestión de procesos y la calendarización de tareas son aspectos cruciales para asegurar el uso eficiente de los recursos disponibles. La capacidad de calendarizar adecuadamente las tareas dentro del sistema permite a los desarrolladores optimizar el rendimiento, al seleccionar de forma apropiada qué procesos deben ejecutarse en cada momento.

El propósito de este proyecto es crear una biblioteca personalizada de hilos llamada CEThreads, la cual reimplementa las funciones principales de la biblioteca Pthreads. A través de esta implementación, se busca controlar la ejecución de los hilos mediante distintos algoritmos de calendarización, tales como Round Robin (RR), Shortest Job First (SJF), First Come First Serve (FCFS) y por prioridades. Cada uno de estos algoritmos se encargará de ordenar la çola de listos", asegurando que el hilo más adecuado sea ejecutado en el momento preciso.

Para este proyecto, se ha utilizado la analogía de un canal de navegación por el que transitan barcos, donde cada barco representa un proceso. Los barcos avanzan en una única dirección y se organizan de acuerdo con el algoritmo de flujo seleccionado, el cual determina en qué orden los barcos pueden cruzar el canal. Este modelo es útil para entender cómo los distintos algoritmos de calendarización gestionan los procesos dentro de un sistema operativo, evitando situaciones de ineficiencia o conflictos en el uso de recursos.

Además, el proyecto contempla la implementación de un prototipo de hardware que permitirá visualizar el comportamiento de los hilos a medida que avanzan a través del canal. Se ofrece también la opción de desarrollar una interfaz gráfica que ayude a representar visualmente el manejo de los hilos y los recursos involucrados.

II. Atributos

II-A. Estrategias individuales y de equipo

Durante el desarrollo del proyecto, se implementaron estrategias que garantizaron un enfoque equitativo e inclusivo tanto en el trabajo individual como en equipo, distribuyendo tareas y asegurando la participación de todos los miembros.

II-A1. Planificación:

- Distribución equitativa de tareas basada en las habilidades e intereses individuales de los miembros del equipo.
- Asignación de responsabilidades claras para evitar dependencia entre miembros y fomentar la autonomía.
- Inclusión de todos los miembros en la toma de decisiones iniciales sobre la estructura general del sistema.

II-A2. Ejecución:

- Trabajo individual en ramas separadas de GitHub para permitir autonomía en el desarrollo y minimizar conflictos.
- Comunicación constante a través de herramientas como para compartir avances y resolver dudas.
- Inclusión activa de todos los miembros en la discusión de decisiones técnicas importantes.

II-A3. Evaluación:

- Retroalimentación constructiva de todos los miembros del equipo sobre el desempeño individual y grupal.
- Evaluación conjunta de la calidad del código y el cumplimiento de los plazos establecidos.
- Revisión colectiva del trabajo final para garantizar que se respetaran los criterios de equidad e inclusión en todas las etapas del proyecto.

II-B. Planificación del trabajo

La planificación del trabajo en este proyecto se realizó asignando roles y estableciendo metas y reglas claras para asegurar un flujo de trabajo eficiente. Todos los miembros del equipo desempeñaron el rol de desarrolladores, contribuyendo directamente al desarrollo del sistema. Sin embargo, se asignaron roles adicionales para facilitar la coordinación y el progreso del proyecto:

- **Consultores**: Dos miembros del equipo fueron designados para comunicarse directamente con el profesor y aclarar cualquier duda o consulta generada durante el desarrollo del proyecto. Esto evitó confusiones y agilizó la toma de decisiones.

- **Coordinador de progreso**: Uno de los compañeros fue responsable de monitorear el avance de los diferentes miembros, asegurándose de que todos completaran sus tareas a tiempo. Si algún miembro enfrentaba dificultades, el coordinador se encargaba de buscar soluciones y brindar ayuda.
- **Encargado del repositorio**: El último miembro asumió la responsabilidad de crear y mantener el repositorio de GitHub, gestionando cualquier aspecto relacionado con el control de versiones y asegurando que todo el equipo tuviera acceso al repositorio.

II-B1. Metas y reglas de trabajo: Durante el desarrollo del proyecto, se establecieron las siguientes metas y reglas de trabajo:

- Plazos específicos: Se definieron plazos concretos para el desarrollo de partes clave del código, especialmente aquellas que eran fundamentales para que otros compañeros pudieran continuar con sus tareas.
- Avance continuo: Las metas establecidas ayudaron a garantizar un avance ordenado del proyecto.
- Uso de Git: Se implementó el uso obligatorio de Git como sistema de control de versiones, asegurando que todos los miembros realizaran pushes regulares.
- Pushes regulares: Los miembros debían realizar pushes especialmente al terminar una funcionalidad importante para mantener la integridad del código.
- Comunicación constante: Se promovió una comunicación constante y efectiva entre los miembros del equipo, insistiendo en la importancia de mantener informados a los demás sobre el estado de sus tareas.
- Resolución de problemas: Se alentó a los miembros a comunicar cualquier problema que surgiera, ya que una comunicación clara es esencial para el éxito del proyecto.

II-C. Acciones de colaboración entre los miembros del equipo durante el desarrollo del proyecto

- Uso de plataformas como un grupo creado en Whatsapp para mantener una comunicación constante, discutir dudas, compartir avances y coordinar tareas.
- Realización de reuniones regulares para revisar el estado de las tareas, identificar obstáculos y ofrecer apoyo entre los miembros.
- Realización de toma de decisiones conjunta en cambios importantes del sistema o de los algoritmos de calendarización.
- Implementación de flexibilidad y apoyo mutuo, redistribuyendo tareas si algún miembro enfrentaba dificultades.
- Uso de documentación del código para asegurar claridad en las contribuciones de cada miembro y facilitar el mantenimiento por parte del equipo.
- Establecimiento de estándares de codificación para asegurar la coherencia y calidad del código entre todos los miembros del equipo.
- Colaboración en la creación de pruebas y validación para garantizar la calidad del código, incluyendo pruebas unitarias y la revisión de los resultados.
- Implementación de feedback continuo, ofreciendo retroalimentación frecuente sobre el progreso y haciendo ajustes según las necesidades del proyecto.

II-D. Ejecución de estrategias planificadas para el logro de los objetivos

Durante la ejecución del proyecto, se llevaron a cabo diversas acciones para implementar las estrategias planificadas y alcanzar los objetivos establecidos:

- Monitoreo constante del progreso de las tareas mediante el uso de herramientas de gestión de proyectos, asegurando que cada miembro avanzara según lo planeado.
- Revisión periódica de las contribuciones individuales a través de GitHub para garantizar que el código estuviera alineado con los estándares acordados y facilitar la integración.
- Fomento de la comunicación activa durante las reuniones regulares, donde se discutían los avances, se abordaban obstáculos y se ajustaban las tareas según fuera necesario.
- Establecimiento de un ambiente colaborativo donde se brindara apoyo mutuo para resolver problemas técnicos y optimizar el desarrollo.
- Evaluación continua de las decisiones técnicas, asegurando que todos los miembros estuvieran informados y comprometidos con las modificaciones realizadas en el proyecto.
- Documentación sistemática de las decisiones y cambios realizados, lo que permitió un seguimiento claro de la evolución del proyecto y facilitó el trabajo en equipo. Para esto se creó un documento que se utilizó para mantener la idea general del proyecto. De esta forma al surgir una duda se acudía a dicho documento.

II-E. Evaluación del desempeño del trabajo individual y en equipo

La evaluación del desempeño durante el desarrollo del proyecto se realizó a través de las siguientes acciones:

- Revisión periódica del progreso individual, donde cada miembro presentaba sus avances y resultados en reuniones regulares.
- Uso de feedback constructivo entre los miembros del equipo para reforzar el aprendizaje y la colaboración, proporcionando sugerencias y reconocimiento por los logros alcanzados.
- Evaluación del cumplimiento de los plazos establecidos, analizando si las tareas se completaban dentro de los tiempos previstos y realizando ajustes cuando era necesario.
- Se realizó un proceso por el cual cada vez que un compañero terminaba su tarea, debía avisar al resto del grupo cuando realizaba su push y explicar que terminó y como utilizarlo, aún fuera de las reuniones regulares.

II-F. Evaluación de las estrategias utilizadas de equidad e inclusión

La evaluación de las estrategias de equidad e inclusión implementadas en el proyecto se llevó a cabo a través de las siguientes acciones:

- Evaluación del balance en la carga de trabajo, revisando si todas las tareas se distribuyeron de manera justa y si cada miembro del equipo tuvo la oportunidad de contribuir.
- Observación de la participación activa de todos los miembros en las discusiones y decisiones del equipo, asegurando que se escucharan todas las voces y perspectivas.
- Análisis de la efectividad de la comunicación, valorando si los canales utilizados fueron accesibles para todos y si fomentaron un ambiente de colaboración.
- Revisión de las decisiones tomadas en conjunto para garantizar que se consideraran las opiniones de todos los miembros y que se abordaran de manera equitativa.

II-G. Evaluación de las acciones de colaboración entre los miembros del equipo

La evaluación de las acciones de colaboración implementadas en el proyecto se realizó mediante las siguientes acciones:

- Reuniones de retroalimentación: Se llevaron a cabo reuniones para discutir el progreso del trabajo en equipo.
- Análisis de la comunicación: Se evaluó la efectividad de los canales de comunicación utilizados, considerando si facilitaron la resolución de problemas y el intercambio de información, en especial hablar siempre en el grupo en vez de manera individual entre compañeros, de esta forma se determinó que fue mas útil ya que todos estaban al tanto de lo que sucedía.
- Observación de la participación: Se observó el nivel de participación de cada miembro en las discusiones y actividades grupales para asegurar que todos estuvieran comprometidos.
- Evaluación del apoyo mutuo: Se discutió cómo los miembros del equipo ofrecieron ayuda y colaboración, especialmente en momentos críticos del desarrollo.

III. Diseño

En cuando al diseño, el grupo de trabajo se basó en los siguientes diagramas.

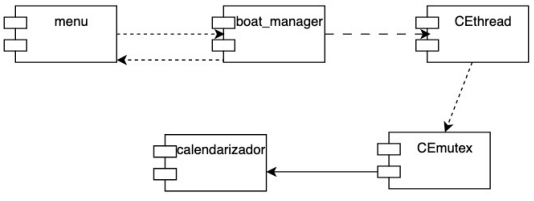

Figura 1: Diagrama de componentes

En la Figura 3 se observa el diagrama de componentes correspondiente al sistema desarrollado. Este muestra sus componentes separados. Primero el menú del sistema, el cual interactúa directamente con el **boat_manager**, este es el componente del sistema donde se define la lógica principal, todo el flujo del programa está aquí. Esta sección se comunica con **CEthread**, el cual es la biblioteca que representa la propia

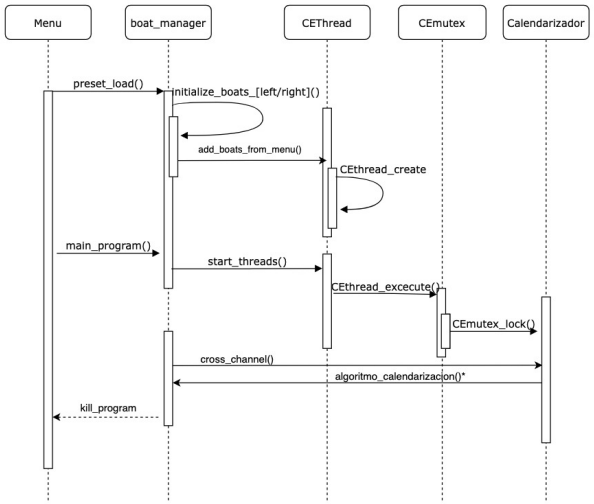

Figura 2: Diagrama de secuencia

biblioteca de hilos del lenguaje C, además del componente mutex, que se encarga de permitir el uso correcto de recursos que son necesariamente manipulados por varios elementos del sistema. Por último, existe el componente de calendarizador, que contienen los algoritmos para calendarizar los hilos (barcos) del sistema.

En el diagrama de de secuencia de la Figura 2, se observa el flujo que tiene todo el sistema a nivel de software. Primero, el sistema inicia con un interfaz sencilla de terminal Linux para que el usuario interactúe con el programa. Este representa el menú, que luego de hacer la carga de barcos y definir que barcos van a que lado del canal, este pasa al boat_manager que maneja el flijo del sistema, aquí se hace todo, solo se delega el trabajo a la biblioteca que posee las funciones de manejo de hilos, los mutex y el calendarizador de sistema, que permite implementar los algoritmos necesarios sobre colas de listo, todo esto cuando lo defina el boat_manager.

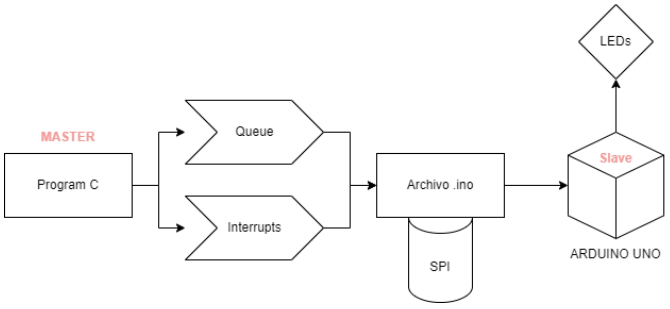

Figura 3: Diagrama de arquitectura hardware

- **Program C (Master)**: Es el programa que se ejecuta en el lado maestro, escrito en C. Este programa envía datos y controla el flujo hacia el Arduino UNO, que actúa como el esclavo.
- **Queue e Interrupts**: El flujo de datos desde el programa en C se gestiona a través de una **cola** (Queue), donde los datos se almacenan temporalmente, y también mediante

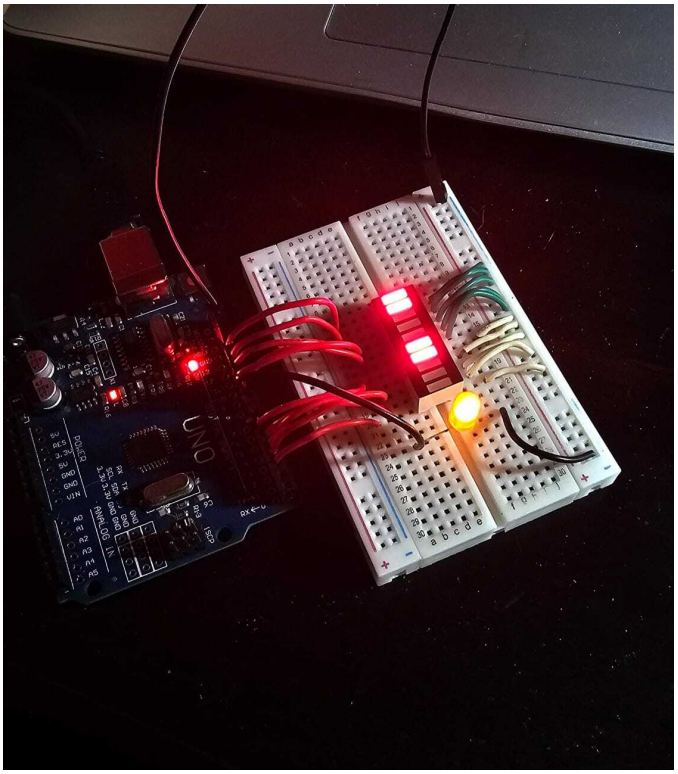

Figura 4: Foto del Hardware físico.

interrupciones (Interrupts), que permiten al programa manejar eventos asíncronos o cambios importantes en el estado del sistema.

- **Archivo .ino**: Representa el programa que corre en el Arduino UNO, escrito en el lenguaje de Arduino (.ino). Este archivo recibe los datos enviados desde el programa en C, los procesa, y controla las acciones a tomar con base en esos datos.
- **Arduino UNO (Slave)**: El Arduino actúa como esclavo en esta configuración. Recibe las instrucciones del programa en C a través de SPI y ejecuta las órdenes correspondientes.
- **LEDs**: El Arduino controla un conjunto de **LEDs** conectados, los cuales son activados o desactivados con base en las instrucciones que recibe del programa en C. Esto podría representar la activación visual de los barcos, como en el caso de un simulador de tráfico marítimo.

IV. Instrucciones de uso

1. Primero se debe abrir la terminal de Linux, y dirigirse a la carpeta ”src”que contiene el Makefile.
2. Luego, es posible modificar el archivo `config.tx`. Es importante destacar que, para los algoritmos de control de flujo, la variable `flow_control_method` puede tomar los siguientes valores:
   - `1`: Equidad
   - `2`: Letrero
   - `3`: Tico

   En cuanto a los métodos de calendarización, la variable `scheduling_method` puede tomar los siguientes valores:
   - `0`: Prioridad
   - `1`: Shortest Job First
   - `2`: First Come First Served
   - `3`: Round Robin
   - `4`: Tiempo Real
3. Posteriormente, ejecutar el Makefile con el comando ”make”. Esto compilará los ejecutable necesarios para la ejecución del programa.
4. Ahora, ingresar a carpeta program y ejecute el programa con el ejecutable del menú así ”./menu”. Esto desplegará un menú en consola.
5. Luego de definir lo que pide el menú, el programa funcionará automáticamente, una vez que el programa esté corriendo se pueden presionar las teclas para generar otros barcos de la siguiente manera.
   - q: normal/derecha
   - w: pesquero/derecha
   - e: patrulla/derecha
   - r: normal/izquierda
   - t: pesquero/izquierda
   - y: patrulla/izquierda

Figura 5: Demostración de ejecución en consola

La Figura 5 es una demostración de la ejecución del programa,

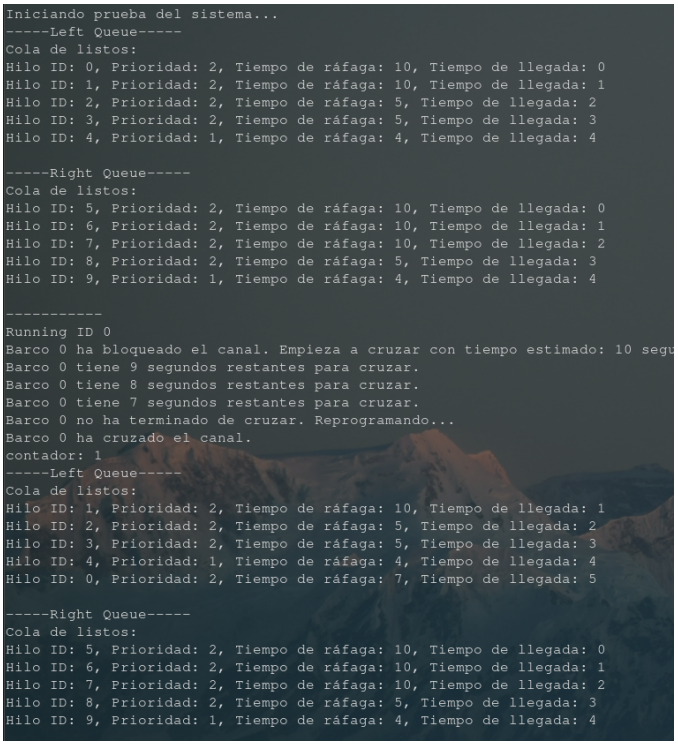

Figura 6: Funcionamiento del Round Robin.

y en la Figura 6 se observa algo del sistema corriendo las pruebas con el algoritmo Round Robin con quantum de 3 unidades de tiempo.

V. Conclusiones

El desarrollo de este proyecto permitió implementar exitosamente una biblioteca de hilos (CEThreads) que maneja de forma efectiva la calendarización de procesos utilizando diversos algoritmos, como Round Robin, Earliest Deadline Fisrt, Shortest Job First, First Come First Serve, y Prioridades. La estructura modular del sistema y su clara diferenciación entre componentes facilitó la implementación y depuración de cada sección, asegurando que el sistema funcionara de manera eficiente en un entorno Linux.

Uno de los mayores logros fue el correcto manejo de recursos compartidos mediante el uso de un mutex, lo que evitó condiciones de carrera y aseguró que cada hilo accediera de manera segura a los recursos necesarios. Además, la simulación de un entorno de navegación de barcos como procesos proporcionó una analogía sencilla y efectiva para visualizar la ejecución y calendarización de tareas dentro de un sistema operativo.

Este proyecto demostró la importancia de la planificación detallada, la colaboración en equipo y el uso eficiente de las herramientas de control de versiones. La correcta distribución de tareas y el seguimiento constante de los avances fueron factores clave en el éxito de la implementación.

VI. Recomendaciones y sugerencias

Optimización del código: Se sugiere realizar una optimización adicional de los algoritmos de calendarización para reducir la sobrecarga en sistemas con un número elevado de hilos. Algoritmos como SJF podrían mejorarse implementando una versión no expropiativa que evite el overhead de la reprogramación constante.

Ampliación de funcionalidades: Sería beneficioso expandir la funcionalidad de la biblioteca CEThreads, añadiendo soporte para más algoritmos de calendarización avanzados, como el algoritmo de calendario más justo (Fair Scheduling), o algoritmos orientados a sistemas en tiempo real.

Interfaz gráfica: La inclusión de una interfaz gráfica, aunque no fue requerida en este proyecto, podría ser una mejora significativa para visualizar de manera más intuitiva el comportamiento de los hilos y su gestión en tiempo real, facilitando así la depuración y comprensión del sistema.

Pruebas de estrés y casos de uso reales: Es recomendable someter el sistema a pruebas de estrés con un gran número de hilos y escenarios complejos. Esto permitiría evaluar su rendimiento bajo carga intensa y garantizar que sea escalable para aplicaciones más complejas.

Uso en sistemas embebidos: Como paso siguiente, se podría explorar la implementación de esta biblioteca en sistemas embebidos, donde la gestión eficiente de recursos es crítica, y los algoritmos de calendarización deben adaptarse a las limitaciones de hardware.

Referencias

[1] ”Pthreads (POSIX threads)”, MAN7, 2024. Disponible: https://man7.org/linux/man-pages/man7/pthreads.7.html