UNIVERSIDAD NACIONAL DE COSTA RICA

FACULTAD DE CIENCIAS EXACTAS

SISTEMAS OPERATIVOS

Proyecto 1

Elaborado por:
Giancarlo Alvarado Sánchez
David Villalobos González

Grupo:
06

Profesor:
Eddy Ramírez Jiménez

HEREDIA, I SEMESTRE 2020

PROYECTO 1

INTRODUCCIÓN

En el siguiente trabajo exploraremos el manejo de hilos y concurrencia mediante el lenguaje de programación C, de este modo intentaremos realizar una solución sobre cómo utilizar muchas de las funciones respectivas a los mismos hilos y todas sus virtudes o problemas al ser llevados a cabo en un sistema operativo óptimo.

Este trabajo llega a ser sumamente importante en el desarrollo y aprendizaje dirigido hacia la introducción de los estudiantes del curso de Sistemas Operativos en los aspectos básicos del manejo de hilos, que aunque previamente fueron explorados en el curso de Programación III, mantiene su utilidad como un claro repaso y es de buen referente para algunos estudiantes en tanto a un refrán que en varios años ha rondado en la cultura colectiva del ser humano occidental: *"Lo que bien se aprende nunca se olvida".*

El problema o ejercicio que se va a plantear se basa principalmente en la premisa referente a la sincronización, ya que en el mismo nos vemos en la necesidad de lograr que varios *ítems* o para este caso específico, automóviles, patrullas policiales y ambulancias puedan circular por un camino estrecho por el cual solo uno de ellos puede pasar por momento y sobre como al usar hilos y concurrencia logramos evitar deadlocks y busy waiting para el correcto "transito" de los mismos.

En términos generales, este trabajo ha sido de gran beneficio para nosotros como estudiantes y por ello deseo, mediante este trabajo escrito, mostrar y dar constancia de todos los aspectos que consideré beneficiosos en la investigación para el manejo de hilos y concurrencia en el lenguaje C y la librería Pthread.

DESCRIPCIÓN DEL PROBLEMA

El problema o ejercicio se basa en una simulación de un puente por el cual atraviesan automóviles provenientes de dos sectores, este y oeste, dado que el puente solo mantiene el ancho de un automóvil, provoca que en un punto solo puedan circular automóviles en un solo sentido, como cada automóvil es conducido por una hebra se deberá tener un manejo adecuado en la sincronización entre los automóviles, de manera que se logre evitar deadlocks y busy-waiting para el correcto "transito" de los mismos.

En términos generales, este trabajo ha sido de gran beneficio para nosotros como estudiantes y por ello deseo, mediante este trabajo escrito, mostrar y dar constancia de todos los aspectos que consideré beneficiosos en la investigación para el manejo de hilos y concurrencia en el lenguaje C y la librería Pthread.

MARCO TEÓRICO

En este proyecto que realizamos se vieron una serie de "problemas" los cuales tuvimos que tratar con diversos métodos de investigación que por métodos externos a nuestro control se debieron realizar por medios virtuales únicamente, tales como buscadores web, Google Scholar, bases de datos de distintas universidades e incluso guías videograbadas en distintas páginas alrededor de la web.

Consideramos que el manejo de la memoria, o específicamente, el tamaño de las variables en el procesador es indispensable para poder desarrollar un proyecto ya que la caída de este suele ser repetitiva si no se controla de una manera completa. El uso de la concurrencia da un entendimiento pleno de la concurrencia da paso a la solución de múltiples problemas en cuanto a la sincronización de datos se refiere. Evitar que se generen dos o más hilos al mismo tiempo que realicen la misma acción, en el mismo momento, en el mismo núcleo del procesador y utilizando los mismos recursos.

Formalizar los datos de cada variable es de gran ayuda para lograr el correcto funcionamiento del proyecto, se recomienda ir manejando la memoria al mismo tiempo que se van creando y manejando las variables. El resultado del proyecto fue favorable para nosotros como grupo ya que consideramos un trabajo final favorable y que nos consideramos considerablemente orgullosos del resultado de este.

Para implementar el proyecto, como ya se mencionó, se utilizó la librería de pthread.h para el manejo de hilos en POSIX. Los procesos trabajan por default trabaja con un único hilo de control que hace sólo una cosa a la vez. Con múltiples hilos podemos hacer más de una cosa a la vez donde cada hilo se hace cargo de una tarea o de la misma mediante sincronización. Para lograr esto se investigó las siguientes funciones de POSIX:
Con pthread_t tid podemos crear un hilo, pero no se va a ejecutar hasta invocar a la función, para correr un nuevo hilo con el método pthread_create

int pthread_create(pthread_t * restrict tidp, const pthread_attr_t * restrict attr, void * (* start_routine) (void *), void * restrict arg);

tidp: salida, puntero a id del hilo
attr: entrada, para definir atributos del hilo, null para default
start_routine: entrada, función a correr por el hilo
arg: entrada, argumento de la función del hilo.

Esta retorna un 0 si se creo correctamente sino retorna un numero de error

Para finalizar un hilo se debe colocar el metodo pthread_join la función rutina o la que está ejecutando el hilo debe retornar un * void, el cual es interpretado como el estatus de término de la rutina por pthread_join.

int pthread_join(pthread_t tid, void ** rval_ptr);

tidp: salida, puntero a id del hilo
ptr: retorno del método

Como un hilo retorna un Void*, no debemos colocar nada, solo finalizar llamando pthread_exit.
void pthread_exit (void * rval_ptr);

Ptr: valor de retorno

Para mantener una sincronización correcta de las variables en las llamadas zonas criticas podemos crear candados con pthread_mutex id y manejarlo con las siguientes funciones:
pthread_mutex_init(pthread_mutex id)
permite dar las condiciones iniciales a un candado mutex

pthread_mutex_destroy(pthread_mutex id)
Este destruye la variable (de tipo pthread_mutex_t) usada para manejo de exclusión mutua, o candado mutes
Si el mutex lo tenía otro hilo y éste es destruido, POSIX no define el comportamiento de mutex en esta situación.

pthread_mutex_lock
Permite solicitar acceso al mutex, el hilo se bloquea mientras otro lo esta utilizando

pthread_mutex_unlock()
Permute liberar un mutex bloqueado.

BITÁCORA

- **17/03:** Arreglamos el error cuando desaparecían autos. Con un if en el print del puente se identifican las direcciones y movimiento de los autos, arriba y abajo. Implementamos la modalidad 2, de semáforos, por el momento funcional, hasta que encuentre alguna pulga. Actualizamos el readme y organizamos un poco el main (creamos un método que tiene toda la simulación).

- **18/03:** Implementamos lo de leer un archivo de configuración, además nos dimos cuenta de que estabamos construyendo autos en ambos sentidos con la misma media pero es una media para cada sentido, otro día lo arreglaremos. A todo esto, nos está saliendo un error extraño, creemos que tiene que ver con la asignación de memoria y que no alcanzamos a resolver ahí. Despues lo resolveremos.

- **21/03:** Solucionamos el error anterior (el raro) después de depurar a pata durante mil años, notamos que un struct estaba mal inicializado. Surgieron miles de incógnitas cuando empezamos a hacer la modalidad 3, nos dimos cuenta que deberíamos desarmar el método choose car, porque el que elige los carros es el algoritmo, según el modo y tener un switch preguntando la modalidad a cada rato, aparte de ser busy waiting, puede que sea innecesario, debo armar un nuevo plan para solucionar eso, ya que ahora me di cuenta que hicimos busy waiting en todo lado. Hay un nuevo error ahora al liberar la memoria en las colas donde ya cruzaron los carros.

- **27/03:** Tuvimos que restructurar el programa y tratamos de meter todo en structs, para que sea más ordenado, ahora tengo que separarlo en diferentes archivos. En teoría ya funcionan los primeros dos algoritmos, el tercer tiene complicaciones que estamos intentando ver, no logramos comprender muy bien que pasa, por dicha aprendimos a depurar en C y ahora nos es más fácil corregir errores. Nos salieron más dudas en todo, ojalá podamos conversar con el profe. Pero ahí esta más o menos, ahí vemos cuando podemos hablar del proyecto.

- **09/04:** Separamos todo el programa en distintos archivos, .h y .c, ya bien. Configuramos el makefile para todo el proyecto, además dejamos activa la depuración. Le implementamos un tipo de priorización a cada modalidad, en caso de que llegue una ambulancia, por ejemplo, si está en fifo los carros de un sector le ceden el espacio a los del otro lado, aunque estos tengan la vía, lo mismo para las demás modalidades. Se arreglo el error relacionado con la liberación de la memoria, la verdad no sabemos cómo, solo se arregló. Le agregamos un tipo de estadísticas al programa para que se puedan ver al final de la simulación, no es un requisito, pero nos pareció interesante. El programa verifica que todas las configuraciones fueran cargadas sino no inicia excepto la modalidad que se le puede preguntar al usuario al inicio de la simulación. Por cierto, el programa ya está listo para iniciarse con un archivo de configuración que se mande en los parámetros del main y cambiarle el ID desde ahí, si no se configuró se pregunta, además si se digita "./start help" se muestra esas características en pantalla como un tipo de ayuda.

- **12/04:** Separamos la estructura global simulation, para que se cree desde el main, así se puede probar múltiples simulaciones con el código, además creamos una estructura llamada “argumentos” para el paso de argumentos hacia los métodos que son utilizados por hilos, que necesitan recibir múltiples parámetros e hicimos un back up en caso de que el proyecto se falle al restructurarlo después de las consultas que le hicimos al profe.

- **17/04:** Realizamos la restructuración y al parecer el código cada vez se hace más pequeño, lo que paso es que nos estábamos complicando mucho la vida.

CONCLUSIONES

- El manejo y uso correcto de los hilos y la concurrencia es fundamental para el completo entendimiento de nuestra carrera universitaria.

- Es fundamental lograr un manejo de la memoria de las variables para evitar problemas del proyecto.

- Los hilos son una secuencia de tareas muy pequeña que puede iniciar el sistema operativo.

- Sin los hilos, el uso de sistemas operativos como los conocemos actualmente sería imposible.

- Los hilos que comparten los mismos recursos agregados a recursos se denominan colectivamente como proceso.

- El hecho de que los hilos del mismo proceso compartan recursos significa que cualquiera de estos hilos puede modificar estos recursos.

- Cuando un hilo modifica datos en la memoria, otros hilos acceden instantáneamente a estos datos modificados.

OPINIÓN PERSONAL

En términos generales podemos definir este proyecto como un gran avance en nuestra formación universitaria que se representa en nuestro plan de estudio, por lo cual consideramos que el aprendizaje referente al manejo, creación y uso de los Hilos ha sido completamente gratificante. Este tipo de proyectos impulsan al estudiante a utilizar diversos métodos de aprendizaje, investigaciones pertinentes y sumado al material brindado por el profesor en cuanto los aspectos teóricos o académicos del mismo brinda una fuerte seguridad al estudiantado sobre lo que se realiza y da por completado un buen número detalles referentes a los que se deben ver en la carta al estudiante correspondiente al curso de Sistemas Operativos.

En cuanto a la experiencia personal correspondiente a los estudiantes Giancarlo Alvarado Sánchez y David Villalobos González, consideramos que el proyecto es realmente grato para el aumento del conocimiento, pero también creemos de que en cuanto posibles “mejoras” (en lo que cabe la palabra) se podría mostrar a los estudiantes ciertas preferencias del profesor en cuanto formas de presentación del proyecto. Por ejemplo, la recomendación de Overleaf u otro método de LaTeX, o un compilador que el profesor considere de mayor beneficio a la hora de desarrollar el proyecto.

Esto solo para aumentar en mucha mayor medida el conocimiento del estudiantado en cuanto aspectos no calificados en el curso se refiere. Por lo demás podemos considerar el proyecto como excelente.