Thread City

Instituto Tecnológico de Costa Rica
Escuela de Computación

Proyecto 1
Sistemas Operativos

Profesor:
Kevin Moraga

Estudiantes:
Luis Alejandro Salas Rojas/2014010742
Luis Alpizar Morales/2015099452

I Semestre - 2018

1 Introduccion

En este proyecto, se nos pide implementar la biblioteca pthreads,con el nombre de myThread, la cual debe contener una serie de funcionalidades que se hacen parecer a la biblioteca de C llamada pthreads, como lo son myThreadCreate,myThreadEnd, myThreadYield, myThreadJoin, myThreadDetach, myMutexInit, myMutexDestroy, myMutexLock, myMutexUnlock y por ultimo la de myMutexTrylock, además de esto, hay que implementar tres tipos de schedulers(RoundRobin, sorteo, tiempo real) los cuales haran referencia a la prioridad en que se va ejecutar cada funcionalidad del programa.

Este proyecto tiene la intención de comprender el funcionamiento de hilos en el Sistema Operativo Linux para asi poder crear cada una de las funcionalidades que estos tienen en una biblioteca nuestra.

Basados en la creacion de estas bibliotecas debemos crear una ciudad basada en hilos, esta debe contener carros, ambulancias, barcos y plantas nucleares para su correcto funcionamiento, tambien se debe utilizar un algoritmo para que los movimientos de cada una de las piezas se de de una manera mas rapida gracias a que se encuentra el camino mas corto. Los movimientos se daran de manera random y gracias a los hilos estos se daran al mismo tiempo todos.

2 Ambiente de desarrollo

Para realizar esta tarea se utilizara un lenguaje de programación, como C. En C se implementaron las funciones de la biblioteca pthread de una manera muy similar a la original, tambien se realizo una matriz de adyacencia la cual es para que el algoritmo dijkstra sepa donde se puede realizar un movimiento y donde no, tambien se crearon los .c de myMutex, Scheduler y startThread que son importados al main para ser utilizado en ligar de la biblioteca pthread.

Tambien se creo un makefile para compilar el proyecto y asi verificar si su estructura se encuentra desarrollada de una manera correcta.

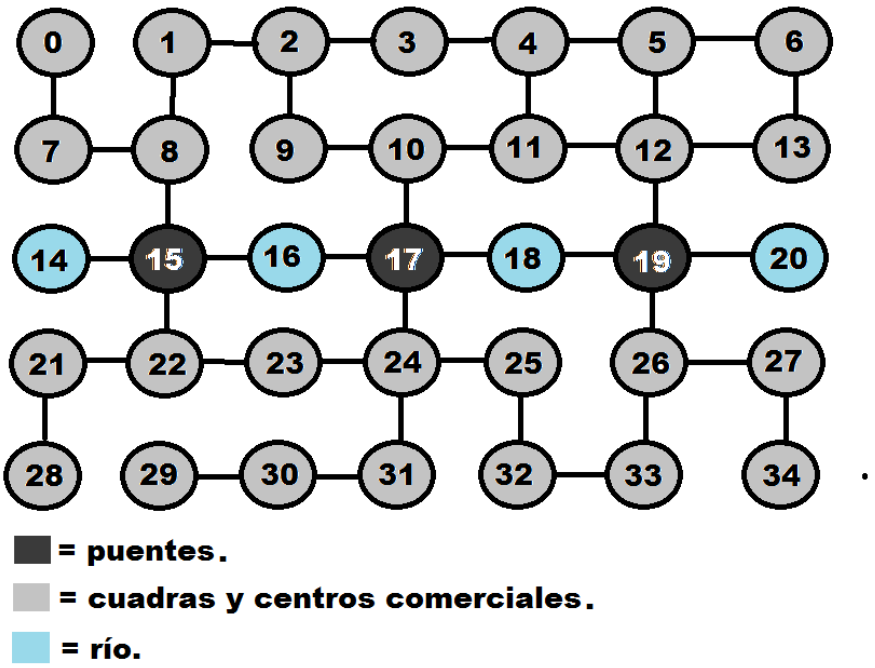
3 Estructuras de datos usadas y funciones

Para realizar esta tarea, se utilizaron algunas estructuras, como un arreglo de hilos para que se creen los carros que se deben mover por la ciudad y que estos lo hagan al mismo tiempo, de la misma forma se hizo para las ambulancias y los barcos.

Entre las funciones utilizadas, las más importantes son: la de createThread que lo que hace es ejecutar el hilo, tambien esta la de dijkstra que lo que hace es encontrar el camino mas corto para realizar un movimiento y por ultimo la que imprime la matriz que esta valida las diferentes posiciones y le da color a cada una de las letras que se utilizaron.

Ademas de esto se manejan dos matrices las cules son, una que es la matriz de adyacencia la cual lo que hace es validar el movimiento que desea hacer cada carro, ambulancia o barco y esta se puede observar en la imagen que se encuentra abajo.

Y en la otra matriz es la que se imprime para que se vea la ciudad represemtada con 0 cuando el lugar esta vacio, con A cuando es una ambulancia, B cuando es un barco y C cuando es una carro.



4 Instrucciones para ejecutar el programa

Para ejecutar el código se presenta un archivo makefile, por lo que se debe ejecutar:

1- Ingresar al directorio donde se encuentra los archivos del proyecto.
2- make
3- ./main

5 Actividad realizadas por estudiante

5.1 Actividades realizadas por estudiantes Luis Alpizar y Alejandro Salas:

- 16 de abril: Se comienza a leer sobre la implementación de los hilos. Horas: 7
- 17 de abril: Se termina y pone a prueba la funcion de create para ver si realiza lo mismo que la de la biblioteca. Horas: 4
- 18 de abril: Se hacen pruebas con la biblioteca context. Horas: 4
- 19 de abril: Se agregan las primeras funciones de la biblioteca. Horas: 6
- 20 de abril: Se investiga funcionamiento del scheduler. Horas: 4
- 21 de abril: Se realiza el scheduler RoundRobin. Horas: 4
- 21 de abril: Se realizan los schedulers de tiempo real y sorteo. Horas: 7
- 22 de abril: Se investiga sobre el funcionamiento de los mutex. Horas: 6
- 23 de abril: Se implementan los mutex. Horas: 5
- 24 de abril: Se resuelven errores en el código de la biblioteca creada y se prueba. Horas: 3
- 21 de abril: Se crea y se pone a prueba el funcionamiento de dijkstra y su matriz de adyacencia. Horas: 4
- 21 de abril: Se investiga sobre el error que nos da el sleep. Horas: 2
- 22 de abril: Se crean los carros y su movimiento. Horas: 2
- 26 de abril: Se crean las ambulancias y su movimiento. Horas: 2
- 27 de abril: Se crean los barcos y su movimiento. Horas: 2
- 28 de abril: Se crea la matrix que se imprime y su respectiva validacion. Horas: 5
- 29 de abril: Se incorpora la biblioteca creada en el juego y los hilos se ponen a prueba. Horas: 5
- 30 de abril: Se termina la funcionalidad de los semaforos y los ceda. Horas: 2
- 01 de mayo: Se termina la documentacion interna y externa. Horas: 8
- Total de horas: 82 horas

6 Comentarios finales

Se logro concluir todo el proyecto con una exepcion de que el sleep no sirve del todo bien, logramos la creacion de todas las funciones que se piden para este, tambien se pudo finalizar con el algoritmo dijkstra, No se logro implementar los metodos ThreadJoin y ThreadDetach principlamente porque no son indispensables.

7 Conclusiones y Recomendaciones

En este proyecto, la parte de la implementación de los hilos fue muy pesada, por lo que se recomienda empezar por ahi. El tema es complicado para encontrar ejemplos sencillos, por lo que se recomienda tener paciencia, leer mucho, ver muchos ejemplos y tratar de entender todos estos.

Tambien se encontraron problemas con diversas bibliotecas que se necesitaban como la del sleep, por lo que se debe de analizar y buscar bastantes ejemplos para lograr lo que se quiere.

Por ultimo se puede recomendar que en trabajos como estos lo mejor es utilizar la biblioteca pthreads de C y cuando la nuestra esta lista y probada solamente cambiar las funciones por las nuestras.

.

8 Bibliografia

- Dadannagari, N., Dadannagari, N. and profile, V. (2018). Creating your own Thread Library using ¡i¿¡u¿Context Switching¡/u¿¡/i¿. [online] Nitish712.blogspot.com. Recuperado el 01 de mayo de: http://nitish712.blogspot.com /2012/10/thread-library-using-context-switching.html

- Anon, (2018). [online] Recuperado el 01 de mayo de: https://www.quora.com/How-do-I-print-a-colored-output-on-cutm_mediumōrganic&utm_sourcēgoogle_rich_qa utm_campaignḡoogle_rich_qa

- Implementing a Thread Library on Linux (evanjones.ca). (2017). Evanjones.ca. Recuperado el 01 de mayo de: http://www.evanjones.ca/software /threading.html

- Multitasking using setjmp, l. (2017). Multitasking using setjmp,longjmp. Stackoverflow.com. Recuperado el 01 de mayo de: https://stackoverflow.com /questions/2560792 /multitasking-using-setjmp-longjmp

- Algoritmo de Dijkstra. (2018). Es.wikipedia.org. Recuperado el 01 de mayo de: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

- Dijsktra's algorithm. (2018). GeeksforGeeks. Recuperado el 01 de mayo de: https://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/

- IBM Knowledge Center. (2018). Ibm.com. Recuperado el 01 de mayo de: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxbd00/rmctxt.htm

- IBM Knowledge Center. (2018). Ibm.com. Recuperado el 01 de mayo de: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxbd00/rscntx.htmrscntx