ThreadCity

Instituto Tecnológico de Costa Rica
Escuela de Computación

Principios de Sistemas Operativos
I semestre

Profesor:
Kevin Moraga

Alumnos:
Josue David Rodriguez Alfaro
Jose Daniel Rodriguez Huertas
Kevin Rodriguez Murillo

Carné:
2015033850
2015108864
2015097419

Abril 2018

1 Introducción

En la computación el termino thread es muy utilizado debido a que es una tarea que puede ser ejecutada al mismo tiempo que otra tarea. En el presente trabajo, se elaborarán diversos threads para carros, ambulancias, barcos o bien la planta nuclear. Pero, ¿en qué consiste el proyecto?

Se debe realizar una re-implementación de algunas de las funciones de la biblioteca de pthreads de C del Sistema Operativo GNU/Linux, respectivas funciones son:

- my thread create
- my thread end
- my thread yield
- my thread join
- my thread detach
- my mutex init
- my mutex destroy
- my mutex lock
- my mutex unlock
- my mutex trylock

Además de las funciones por defecto de phtreads se debe llevar a cabo la implementación del siguiente método:
- my thread chsched: Se encarga de cambiar el tipo scheduling del hilo.

Por otra parte, se abarca el tema de los schedulers, un scheduler cumple la función de repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución. A continuación, se definen los tipos de schedulers que soportará la biblioteca:

- Scheduler RoundRobin Se debe de realizar la implementación del scheduler siguiendo un algoritmo de RoundRobin.

- Scheduler Sorteo Se debe de realizar la implementación del scheduler siguiendo un algoritmo de Sorteo. Los hilos creados para este scheduler puede que necesiten parámetros extra, por ejemplo, cantidad de tiquetes iniciales.

- Scheduler de Tiempo Real Se debe de realizar la implementación del scheduler siguiendo un algoritmo de Tiempo Real. Los hilos creados para este scheduler puede que necesiten parámetros extra, por ejemplo, límites de tiempo.

Cada una de las funciones mencionadas anteriormente serán incorporadas en la funcionalidad de una ciudad llamada “Thread City”, en la cual habrá carros, plantas nucleares, comercios, ambulancias, barcos y un mínimo de 25 cuadras. Thread City tiene un río en la zona central, lo que hace que la ciudad se dividida

en zona norte y zona sur, la conexión entre las dos secciones de la ciudad es mediante 3 puentes.

Finalmente para la construcción de la ciudad, es importante mencionar las siguientes consideraciones que se deben tomar en cuenta:

• Las ambulancias tienen mayor prioridad que los carros.
• El puente 1 es de dos carriles, y cuenta con un semáforo en cada uno de sus extremos.
• Los barcos solo pueden navegar por el puente 1.
• Al no pasar vehículos cerca de la planta nuclear, se procederá a realizar una explosión, debido a que no se le proporcionó el suficiente material radiactivo.
• El puente 2 presenta una ceda.

2 Ambiente de desarrollo

Se utiliza el lenguaje de programación C, por medio de este lenguaje se lleva a cabo el desarrollo de las funciones de la biblioteca pthreads,para de esta manera poder visualizar la simulación de la ThreadCity. Entre las estructuras de datos más importantes se encuentra un arreglo para almacenarlos hilos, un arreglo para saber la prioridad de los hilos y un último para conocer la cantidad de tiquetes que tiene cada hilo.

La herramienta Git posibilita un mejor orden y mayor rendimiento de acuerdo a las versiones trabajadas del código.

La documentación se elabora en un editor de Látex en línea, el cual nos permite descargar el PDF y la carpeta zip con los archivos necesarios como lo es el .bib.

Para la compilación del programa, se requiere de un archivo makefile en el cual se crea el nombre del ejecutable, se compila ejecutando el gcc con los archivos correspondientes .c y .h.

3 Estructuras de datos usadas y funciones

La solución para la creación del NetNeutrality se realiza en el lenguaje C, las funcionalidades de los principales métodos se mencionan continuación:

• my_thread_create: La cual recibe como parámetros, función a ejecutar, parámetros necesarios, tiquetes iniciales y prioridad. Esta función crea un hilo y lo agrega a la cola del scheduler.
• my_thread_end: La cual se encarga de finalizar un hilo.
• my_thread_yield: Esta función activa el scheduler y cede el procesador.
• run_threads: Esta función activa el scheduler y pone a correr los hilos.
• my_mutex_init: Esta función inicializa el mutex, recibe como parámetro el puntero a la variable mutex.
• my_mutex_lock: Esta función recibe como parámetro el mutex a bloquear y se encarga de bloquear el recurso siguiente en el texto.
• my_mutex_destroy: Esta función finaliza el mutex.
• my_mutex_unlock: Esta función desbloquea el recurso y lo coloca como disponible.
• my_mutex_trylock: Esta función bloquea el recurso, pero si este no está disponible no se bloquea.
• my_thread_chsched: Función que se encarga de cambiar el scheduler activo.
• cityControler: Hilo del juego el cual se encarga de inicializar la matriz, y es quien se encarga de imprimir en pantalla el tablero.

El control de la ciudad es llevado a cabo en el cityController.c, donde se crean los hilos de ambulancias, carros, barcos, y planta nuclear, así como también la actualización de la parte gráfica y cálculo de la distancia mediante dijkstra.

En general se emplean arreglos [] y una matriz [42][42] de adyacencia para el almacenamiento de variables e hilos, que nos permiten el buen funcionamiento del programa.

4 Instrucciones para ejecutar el programa

Para la ejecución del programa se deben seguir los siguientes pasos:

Luego de acceder mediante terminal a la carpeta donde se ubican los archivos, se procede a hacer make, para la compilación de los archivos correspondientes al my_threads.c cityController.c my_mutex.c y scheduler.c. Posteriormente escribir en terminal: "./main" para la ejecución del programa.

Una vez realizados los pasos anteriores se cuenta con una simulación del comportamiento de la ciudad del ThreadCity, con ambulancias, carros, plantas nucleares, comercios y un mínimo de 25 cuadras.

5 Actividad realizadas por estudiante

5.1 Bitácora Josué David

• 16 de abril: Se comienza a leer sobre la implementación de los hilos.
Horas: 5

• 17 de abril: Se inicia investigando sobre context.
Horas: 4

• 18 de abril: Se realizan algunas pruebas con context para entender su funcionamiento.
Horas: 4

• 20 de abril: Luego de llevar a cabo la investigación se procede a hacer un código base utilizando context.
Horas: 5

• 21 de abril: Se continúa investigando sobre las funciones de pthread.
Horas: 6

• 22 de abril: Se realizan algunas funciones de la biblioteca específicamente pthread create.
Horas: 6

• 23 de abril: Junto con otro compañero se investiga los scheduler a implementar.
Horas: 5

• 24 de abril: Se realizan el scheduler de RoundRobin.
Horas: 7

• 26 de abril: Se crea la matriz de adyacencia a utilizar en la ciudad.
Horas: 2

• 27 de abril: Se investiga sobre el scheduler de tiempo Real y se crea Dijkstra.
Horas: 5

• 28 de abril: Se resuelven errores en el código.
Horas: 2

• 29 de abril: Se comienza con la creación del scheduler de tiempo Real y se inicia con la creacion de los hilos que utilizara la ciudad.
Horas: 4

• 30 de abril: Se implementan los hilos para la creacion de carros, plantas nuclerares, barcos y ambulancias, además de las consideraciones como el puente, semáforo y prioridad.
Horas: 9

• 1 de mayo: Se realiza una mejor presentación de la ciudad y se hace la documentación.
Horas: 6

Tiempo invertido en total: 71 horas ,0 minutos

5.2 Bitácora José Daniel

• 16 de abril: Se comienza a leer sobre la implementación de los hilos.
Horas: 3

• 17 de abril: Se inicia investigando sobre context.
Horas: 3

• 18 de abril: Se realizaron algunas pruebas con context para entender su funcionamiento.
Horas: 4

• 20 de abril: Se empezó a plantear un prototipo de la ciudad utilizando la biblioteca pthread.
Horas: 5

• 21 de abril: Se siguió implementando el prototipo y se creó una matriz simple para visualizar la ciudad.
Horas: 6

• 23 de abril: Se crean vehículos en la matriz y se investiga sobre Dijkstra.
Horas: 6

• 24 de abril: Se crean las funciones de crear hilo y detener hilos.
Horas: 7

• 26 de abril: Se prueban las nuevas funciones de hilos en el prototipo
Horas: 4

• 27 de abril: Se investiga un poco sobre GTK.
Horas: 5

• 28 de abril: Se decide no utilizar GTK y se inicia a crear una ciudad que se vea en terminal y se pueda representar con base en la matriz lógica.
Horas: 5

• 29 de abril: Se analizan las zonas donde serán ubicados los puentes y el ceda. Se implementa el ceda y los puentes.
Horas: 4

• 30 de abril: Se implementan los hilos para la creación de carros, plantas nucleares, barcos y ambulancias.
Horas: 9

• 1 de mayo: Se realiza una mejor presentación de la ciudad y se termina la documentación.
Horas: 7

Tiempo invertido en total: 68 horas

5.3 Bitácora Kevin

• 16 de abril: Se comienza a investigar sobre cómo manejar y crear los hilos sin hacer uso de la biblioteca pthread
Horas: 3

• 17 de abril: Se empieza a entender el funcionamiento de context y como se utiliza para el manejo de estado de los hilos.
Horas: 4

• 18 de abril: Se hacen pequeñas pruebas utilizando context.
Horas: 4

• 20 de abril: Se ayuda a compañero a realizar un código base usando context para el manejo de los hilos
Horas: 6

• 21 de abril: Una vez que se entendió bien lo de context, se procede a analizar y entender bien cómo funcionan las distintas funciones del pthread.
Horas: 3

• 22 de abril: Se empiezan a desarrollar algunas funciones de la biblioteca pthread, en específico sobre la de pthread yield
Horas: 6

• 23 de abril: Se continúa implementando las funciones del pthread
Horas: 5

• 24 de abril: Se empieza a investigar y realizar pruebas sobre los schedulers de sorteo.
Horas: 7

• 26 de abril: Se empieza a investigar sobre el algoritmo de dijkstra.
Horas: 2

• 27 de abril: Se ayuda a compañero con la elaboración del algoritmo de dijkstra.
Horas: 5

• 28 de abril: Se resuelven bugs o errores encontrados en el código de dijkstra.
Horas: 4

• 29 de abril: Se comienza a crear la ciudad, creando los hilos de los carros y ambulancias, así como su movimiento dentro de la ciudad.
Horas: 6

• 30 de abril: Se sigue trabajando sobre los hilos de los carros y ambulancias, trabajando sobre la prioridad a la hora de pasar por el puente.
Horas: 8

• 1 de mayo: Se corrigen algunos errores y se mejora la presentación, se empieza la documentación.
Horas: 5

Tiempo invertido en total: 69 horas ,0 minutos

6 Comentarios finales

Se logra satisfactoriamente programar en el lenguaje C las funciones relacionadas a la biblioteca pthreads:

- my thread create
- my thread end
- my thread yield
- my mutex init
- my mutex destroy
- my mutex lock
- my mutex unlock
- my mutex trylock

A excepción del my_thread_join y detach, principalmente porque no son necesarios para la implementación del juego.

Otro punto a tomar en cuenta que no fue posible implementarlo al cien por ciento, fue la calendarización de la llegada de los carros con el algoritmo de scheduling en tiempo real, al utilizar un scheduling distinto, aplicará para los carros también.

La animación de la ciudad en GTK, no se desarrolló por motivos de tiempo, además, porque utilizar la interfaz en el lenguaje de programación C es difícil de manipular y requiere de práctica para saber colocar los elementos en las posiciones adecuadas y hacer los efectos y cambios en la pantalla correctos.

La tarea abarcó bastante tiempo ya que no se tenía bien claro cómo trabajar con threads y schedulers, además no se contaba con mucho conocimiento sobre la forma de manejar una serie de threads simultáneamente.

Un aspecto a considerar es el esfuerzo realizado por cada uno de los compañeros, debido a que se facilitó la solución de la tarea, y cada uno logró comprender los diversos métodos empleados.

7 Conclusiones y Recomendaciones

En la elaboración de este proyecto se obtienen las siguientes conclusiones:

• Se ha logrado un mayor aprendizaje y conocimiento, sobre diversos temas que abarca el curso Principios de Sistemas Operativos, por ejemplo: entender el funcionamiento de los threads, schedulers, implementación de algoritmos como es el caso de RoundRobin, Sorteo, Tiempo Real y Dijkstra, etc.

• Se debe de investigar una gran parte ya que no se está acostumbrado a trabajar en C con hilos, tampoco a realizar un programa que tenga como fin llevar a cabo la simulación de una ciudad tomando en cuenta las diferentes consideraciones planteadas (puentes, semáforos, ceda, prioridad ambulancias).

Recomendaciones

Las recomendaciones para la persona que inicia este tipo de proyecto son las siguientes:

• Consultar una gran cantidad de fuentes bibliográficas para lograr un mayor conocimiento.
• Tener una buena organización de trabajo, posiblemente con un cronograma de actividades.
• Elaborar el trabajo día a día, para que no haya acumulación y las idead puedan fluir mejor.
• Contar con un espacio de trabajo apropiado.
• Pedir instrucción a personas que conozcan sobre el tema.
• Hacer anotaciones precisas.
• Dividirse el trabajo.
• Utilizar el git es de gran ayuda.
• Preguntarse siempre cuál es el objetivo que se quiere lograr y ser perseverante.

.

8 Bibliografia

• Barney, B. (2018). POSIX Threads Programming. [online] Computing.llnl.gov. Recuperado 16 abril, 2018 de: https://computing.llnl.gov/tutorials/pthreads/.

• Murphy, T. (2018). pthreads in C – a minimal working example timmurphy.org. [online] Timmurphy.org. Recuperado16 abril, 2018 de: http://timmurphy.org/2010/05/04/pthreads-in-c-a-minimal-working-example/.

• Multitasking using setjmp, l. (2017). Multitasking using setjmp,longjmp. Stackoverflow.com. Recuperado 17 de abril 2018, de https:stackoverflowċomquestions2560792multitaskingusingsetjmplongjmp

• mfiber: Simple User Level Thread Library. (2017). LID: Lost In Droid. Recuperado 18 de abril 2018, de https://machiry.wordpress.com/2012/01/02/mfibersimple-userlandthreadlibrary/

• Implementing a Thread Library on Linux (evanjones.ca). (2017). Evanjones.ca. Recuperado 18 de abril 2018, de http://www.evanjones.ca/software/threading.html

• Implementing a Thread Library on Linux (evanjones.ca). (2017). Evanjones.ca. Recuperado 18 de octubre 2018, de http://www.evanjones.ca/software/threading.html

• INI file. (2017, October 11). In Wikipedia, The Free Encyclopedia. Recuperado 18 de abril 2018, de https://en.wikipedia.org/w/indexp̀hptitleĪNI_file&oldid8̄04815470

GitHub:
– https://github.com/brianwatling/libfiber
– https://gist.github.com/DanGe42/7148946