Tecnológico de Costa Rica

Primer Proyecto

MultiDisplay Animation

Autores:
Joseph Loaiza Cruz
Yeison Cruz León

Profesor:
Ing. Kevin Moraga, MSc.

01 de noviembre de 2015

1. Introducción

La compresión y el tratamiento de hilos es conocimiento que en la actualidad cualquier programador debería tener, que sería de cualquier software actual sin ellos, sistemas operativos como Windows, iOS o cualquier versión de Linux serían imposibles de ejecutar, pues son muchas las actividades que se deben tratar a un mismo tiempo.

El siguiente programa trata de profundizar en eso mediante la creación de una biblioteca de hilos los cuales funcionarán a nivel de usuario y su posterior uso en un software encargado de animar figuras ASCII a lo largo de varios monitores conectados en red, para ello se utilizarán los siguientes requerimientos de software:

- Se realizará la re-implementación de la biblioteca de pthreads.
- Deben existir tres tipos de schedulers, a saber: **RoundRobin, Sorteo** y **Tiempo Real**.
- El scheduler de Sorteo y RoundRobin serán los que administren los hilos creados, sin embargo, estos serán administrados a su vez por el scheduler de Tiempo Real.
- Se debe crear un software de animación que utilice la biblioteca de hilos creada previamente.
- Las figuras animadas deben tener un tiempo máximo de ejecución, una vez que este se acabe y si la figura no ha llegado a su punto final esta debe explotar.
- Las figuras pueden tener la capacidad de rotar mientras se mueven en angulos de 0, 90, 180 y 270 grados.
- Una figura no puede invadir el espacio de otra, si en algún momento dado una figura debe pasar por donde otra se esta movimiento en el mismo instante de tiempo la figura debe esperar que la otra abandone el lugar.
- El movimiento de las figuras puede ser hacia cualquier dirección dada.

2. Ambiente de desarrollo

Para realizar lo anterior mencionado se utilizarón diversas herramientas que facilitaron su desarollo, entre ellas se encuentran las siguientes:

- **Sublime Text 2:** Esta herramienta es un editor de texto muy popular centrado principalmente en código, el mismo soporta Snippets, plugins varios y una amplia gama de sistemas de construcción de código, entre ellos algunos de los más conocidos como Python, Java, PHP, ASM, C++ y por supuesto C.
- **VMware:** Software de virtualización que simula un sistema físico con unas características dadas por el usuario, en el caso de este programa fue útil para emular un entorno Linux y hacer las pruebas del programa implementado ya que la computadora física no contaba con un entorno de este estilo instalado por defecto.
- **GNU C And C++ Compiler (GCC):** Compilador de los lenguajes C y C++ utilizado en este caso para compilar los archivos .c y .h que fueron creados en el desarrollo del programa.

3. Estructuras de datos usadas y funciones

3.1. Estructuras de datos y bibliotecas externas

En el desarrollo del programa se utilizaron las siguientes estructuras de datos:

- **Arreglo:** Un arreglo es un conjunto de datos que se almacenan en memoria de manera continua, para diferenciar sus elementos se utilizan índices donde cada dato del arreglo tiene asignado un índice distinto iniciando generalmente en cero.
- **Lista Simple Enlazada:** Una lista simple enlazada son un conjunto de nodos unidos por una referencia, donde el primer nodo apunta al siguiente y así de forma consecutiva.
- **Lista Circular:** De igual forma que una lista simple enlazada este tipo de lista contiene un conjunto de nodos enlazados entre sí por una referencia con la salvedad que el nodo inicial también sera el nodo final de la lista.

En el desarrollo del programa se utilizaron las siguientes bibliotecas externas:

- **Ncurse:** Biblioteca que permite de una forma sencilla la impresión de caracteres en pantalla, brinda una terminal especial que puede ser manipulada de una forma sencilla incluyendo la adición de colores y el movimiento por la misma de forma amigable al usuario, para el proyecto actual fue utilizada para la impresión de las figuras ASCII que se movian de un lado al otro así como la impresión del menu en pantalla.
- **LibXML2:** Biblioteca para la manipulación de archivos XML dentro del lenguaje C, presenta funciones predefinidas para el control de los datos así como el acceso a los mismos de una forma sencilla y amigable.

3.2. Funciones principales

Dentro del código del programa se pueden encontrar las siguientes funciones o rutinas principales:

3.2.1. Biblioteca de Threads

- **myThreadInit:** Función encargada de inicializar todas las estructuras que serán utilzadas para el manejo de los hilos, recibe un parámetro el cual corresponde al quantum que utilizará la biblioteca internamente, es importante recalcar que esta función debe ser llamada **SIEMPRE** antes de utilizar cualquier otra función de la biblioteca.
- **myThreadCreate:** Función encargada de la creación de un nuevo hilo dentro del sistema, la misma recibe un apuntador a un parametro de tipo *thread_t* el cual debe ser creado antes de llamar a la función, un apuntador a la función que el hilo debe ejecutar, en el caso que esta función reciba algún parametro este también se puede pasar de lo contrario se puede colocar en NULL, un tiempo límite máximo de ejecución de hilo, sin embargo, sino se quiere que el hilo tenga un máximo tiempo de ejecución se puede pasar un valor de cero y esta funcionalidad será ignorada y por último el tipo de Scheduler que se desea administre el hilo, es decir, RoundRobin o Sort.
- **myThreadSelf:** Esta función devuelve el índice que fue asignado al hilo al crearse para el hilo que la llama.
- **myThreadYield:** Función que pausa la ejecución del hilo que la llama y devuelve el procesador.
- **myThreadExit:** Función que termina la ejecución del hilo que la llama, generalmente esta función es llamada por el mismo hilo.
- **myThreadChsched:** Función encargada de cambiar de scheduler un hilo en tiempo real, es decir, si actualmente un hilo es administrado por el scheduler Sort puede ser cambiado a RoundRobin.
- **myThreadSleep:** Función que duerme el hilo que la llama por una cantidad de tiempo dado, este tiempo debe ser pasado en **milisegundos**.
- **myThreadCancel:** Función que termina la ejecución del hilo pasado por parámetro, es importante recalcar que este hilo cancelado **NO** podrá ser recogido por un join ya que la función que estaba ejecutando no pudo devolver ningún valor.
- **myThreadJoin:** Función que es llamada por otro hilo para esperar la finalización del hilo que es pasado por parámetro así como recoger su valor de retorno.

- **myThreadDetach:** Esta función indica a la biblioteca que no queremos que nos guarde el valor del hilo indicado en el parámetro, es decir, el hilo finalizará pero no podrá ser posible recoger su valor de finalización mediante un join.
- **myThreadGetTimeExecution:** Esta función devuelve el tiempo que lleva ejecutandose el hilo que es pasado por parámetro.
- **myThreadMutexInit:** Esta función recibe un parámetro de tipo *thread_mutex_t* el cual debe ser creado con anticipación, su funcionalidad es la inicializar el mutex que es pasado por parametro creando las estructuras necesarias para su control.
- **myThreadMutexLock:** Esta función bloquea un mutex y coloca a los demás hilos en espera que este sea liberado.
- **myThreadMutexUnlock:** Esta función desbloquea un mutex que fue previamente bloqueado.
- **myThreadMutexTryUnlock:** Esta función bloquea un mutex en caso que el mismo no haya sido bloqueado con anterioridad.
- **myThreadMutexDestroy:** Esta función destruye un mutex que fue previamente inicializado.

3.2.2. Animación

- **animateObject:** Función encargada de la animación de cualquier figura, es la función que ejecutan los hilos que son creados dentro del módulo de animación.
- **setASCIIObject:** Función que carga todas las figuras que serán animadas en memoria, lee los archivos que las contienen y las coloca en una estructura para su posterior utilización.
- **freeAllMemory:** Función que libera toda la memoria que fue utilizada en la ejecución de la animación.
- **printFigure:** Función que envía una figura por red para que esta sea recogida por un servidor y la imprima en pantalla.
- **verifyFigureExplosion:** Función que es ejecutada por cada hilo una vez que haga cualquier movimiento en la pantalla, revisa si el tiempo

máximo que tenía la figura para ejecutarse ha expirado y de ser así inicia la animación de explosión.

4. Instrucciones para ejecutar el programa

Nota: Se asume que el usuario ya tiene instalado el compilador GCC, además de encontrarse en un ambiente Linux.

1. En caso que no tenga instaladas las bibliotecas **ncurse y libxml2** deberá ejecutar los siguientes comandos en orden secuencial: *apt-get install ncurses, apt-get install ncurses-dev, apt-get install libxml2, apt-get install libxml2-dev.*

2. Copie el contenido de todos los archivos fuente a alguna carpeta específica.

3. Muevase por consola hasta esa carpeta específica y ejecute la instrucción **make**.

4. Ejecute la instrucción **sudo ./MultiDisplayAnimation** y continue en el menú.

5. Actividades realizadas por los estudiantes

- **15/09/15 - 10 horas:** Iniciamos la investigación del funcionamiento de los hilos a nivel interno, que tipos de hilos existen (kernel, ULT) así como el funcionamiento de los diferentes schedulers que se deben implementar, Sorteo, RoundRobin y Tiempo Real.
- **16/09/15 - 5 horas:** Investigación sobre como realizar cambios de contexto entre un hilo y otro, como lo hace el sistema y mecanísmos sobre como podríamos hacerlo nosotros, llegamos a la conclusión que existian dos maneras, usando sigsetjmp() / siglongjmp() o de la segunda manera, usando makecontext(), getcontext(), setcontext() y swapcontext().
- **18/09/15 - 15 horas:** Se inicia la construcción de la biblioteca de hilos.
- **19/09/15 - 10 horas:** Se continua la construcción de la biblioteca de hilos junto con el desarrollo del primer scheduler, RoundRobin.
- **20/09/15 - 20 horas:** Se continua la construcción de la biblioteca de hilos y se termina el desarrollo del primer scheduler, la biblioteca funciona como debe ser pero sólo utilizando RoundRobin.
- **22/09/15 - 8 horas:** Se inicia la construccción del scheduler Sorteo, se corrigen errores en la biblioteca de threads que estaban impidiendo el funcionamiento bajo algunos casos de prueba utilizados.
- **23/09/15 - 8 horas:** Se termina la contrucción del scheduler Sorteo y se integra a la biblioteca de threads, se detectan errores críticos en la biblioteca que impedian la integración con el segundo scheduler, la biblioteca funciona como debería de ser utilizando sólo el scheduler Sorteo.
- **25/09/15 - 15 horas:** Se inicia la construcción del sistema de animación utilizando la biblioteca de threads previamente creada, se comienza por la investigación sobre como imprimir caracteres ASCII en pantalla, se toma la decisión en conjunto de utilizar la biblioteca *ncurses* para la impresión, se analiza que opciones existen para el manejo de archivos de configuración, se toma la decisión en conjunto de utilizar el formato XML y se escoje la biblioteca *libxml2* para ello por tener una curva de aprendizaje rápida, se hace un gran avance en materia de animación.

- **26/09/15 - 15 horas:** Hoy Yeison esta cumpliendo años, sin embargo, un día de cumpleaños programando... Se continua la implementación de la biblioteca de animación, se encuentrana algunos problemas que pudieron ser solucionados de forma rápida, se corrigen algunos errores en la biblioteca de threads que estaban bloqueando las animaciones y no fueron detectados en iteraciones anteriores, se logra animar hasta tres figuras al mismo tiempo con rotación pero en local, aún no por red.

- **29/09/15 - 15 horas:** Se inicia el sistema de comunicación red para poder imprimir figuras en varios monitores conectados entre sí, se encuentran errores graves en la lógica de animación los cuales son corregidos.

- **30/09/15 - 8 horas:** Se termina el sistema red así como la animación, el sistema en general cumple con los requerimientos solicitados, se termina el desarrollo del software.

- **01/10/15 - 4 horas:** Se termina la documentación previamente iniciada en el proceso de desarrollo y se integra a los demás archivos.

6. Comentarios finales

6.1. Estado final del programa

El programa cumple con el 100 % de los requerimientos técnicos y funcionales requeridos.

6.2. Problemas encontrados

Se encuentran una diversidad de problemas, se exponen algunos cuantos de ellos, principalmente los más significativos.

- **Uso de estructura compartida o estructuras individuales:** Dado que para el desarollo de este programa se debían implementar dos schedulers diferentes los cuales administraran los hilos se podía tomar dos caminos, implementar una sola estructura de datos y que ambos schedulers la compartieran (esta estructura guardaría todos los hilos, sean administrados por RoundRobin o Sorteo) o bien crear una estructura de datos separada por cada scheduler y que la misma contuviera sólo los hilos correspondientes a cada schedulers, sin embargo, ambas tenian sus ventajas y desventajas, para el primer enfoque se tenía la ventaja que a la hora de cambiar de scheduler no era algo complicado, pues el hilo se encontraba ya en la misma estructura por ende sólo se debian de hacer cambios menores, pero tenía la desventaja que lo que hiciera un scheduler dentro de la estructura podría afectar el trabajo del otro, es decir, tener problemas de concurrencia, en el segundo enfoque era todo lo contrario, la ventaja es que lo que hiciera uno no afectaba al otro pues los hilos estaban totalmente separados, pero hacer un cambio de scheduler se volvia tedioso, luego de pensar un rato y comentar entre ambos se llegó a la conclusión que el primer enfoque era el más adecuado pues optimizaba memoria y trabajar la concurrencia no era un problema tan complicado.

- **Implementación del Join:** Implementar una solución para el join fue algo complicado, pues encontrar una forma de hacer que un hilo espere la muerte de otro es un poco complejo, más aún, poder capturar el valor de retorno del hilo esperado, para lograr esto se tomó la decisión de implementar una estructura de datos la cual correspondía a una lista simple en la cual una vez que un hilo terminara su ejecución guardará los valores de retorno y de esta forma poder consultarlos.

- **Dormir un hilo:** Quizás el problema más complejo de resolver en toda la construcción de la biblioteca de hilos, no se podía usar la función ya existente llamada *sleep* o bien *usleep* porque bloqueaba todos los hilos lo cual no es lo deseado, para resolver este problema se implemento una función que indica cuando tiempo lleva un hilo en ejecución con base en cuantos quantums han pasado de ejecución, esto permite hacer una validación entre el tiempo dado y el tiempo transcurrido, mientras el hilo está esperando realmente ejecuta una función de tipo *busy waiting* que hace que el hilo espere.

- **Choque de animaciones:** Dado que una animación no podía ocupar el campo que alguna otra estuviera usando, se implemento un motor de impacto el cual suspendiera un hilo mientras la otra animación se seguía movimiento, para esto se utilizo un arreglo el cual mapea toda el canvas de animación completo en una matriz, en el momento en que una figura ocupa un espacio este pixel en la matriz es ocupado por la figura de esta forma cuando otra figura necesita ocupar este espacio primero verifica si se encuentra disponible y sino espera.

- **Manejo de posiciones:** Una consola general de animación sólo dispone de 80x24 pixeles únicamente, sin embargo, al tener un canvas que puede estar compuesto por varias consolas el canvas general de animación podria ser de 240x24 por ejemplo, dado que una figura sólo ve un canvas gigantesco esta podría ubicarse en la posición 160 por ejemplo, pero a la hora de hacer una impresión en consola no existe la posición 160 sino sólo de 0 a 80, por ende había que hacer la conversión de este número a su equivalente a una sola consola, este problema fue complicado de resolver pero se logro haciendo cálculos matemáticos de posicionamiento.

- **Explosión de una figura:** El detectar la explosión de una figura fue un trabajo complicado pues se debía de hacer básicamente cada vez que una figura se moviera pues es en ese momento que podía explotar, para hacer esto cada vez una figura termina de moverse y antes que se mueva a la siguiente posición valida si su tiempo de ejecución consumido supero al tiempo establecido, si es así inicia una animación de explosión y sino continua su movimiento.

6.3. Limitaciones adicionales

- No se puede utilizar el caracter **$** para constituir una figura pues el mismo es un caracter reservado del software.
- El caracter | es utilizado para darle estructura a las figuras que serán animadas y es ignorado a la hora de imprimirlas en pantalla, por ende no puede ser utilizado para constituir una figura.
- El software sólo funciona bajo el protocólo IPV4, no puede ser utilizado el protocólo IPV6.
- Las figuras deben tener un tamaño inferior al canvas de animación completo que se esté utilizando.
- No puede existir un sleep de hilo menor al quantum asignado a los hilos en general, la unidad menor de sleep es el quatum mismo, si se llama a la funcion myThreadSleep con un valor inferior a este el hilo solo esperará un quatum.

7. Conlusiones y recomendaciones del proyecto

7.1. Conclusiones

- Los threads a nivel de usuario tiene la ventaja de poder ejecutarse en cualquier sistema operativo con capacidad para compilar lenguaje C.
- El intercambio de los hilos no necesita los privilegios del modo kernel ya que todas las estructuras de datos están en el espacio de direcciones de usuario, por lo tanto, el proceso no debe cambiar a modo kernel para gestionar hilos, es decir, se evita la sobrecarga de cambio de modo y con esto el sobrecoste u overhead
- Una simple llamada al sistema para escribir en un archivo por ejemplo, bloquea todos los hilos que se estan ejecutando.
- Los Mutex son sumamente importantes en el manejo de la concurrencia.
- Un thread mal sincronizado puede hacer que todos los demás fallen.
- Ncurses y LibXML2 son bibliotecas excelentes para impresión en pantalla y el manejo de XML respectivamente.

7.2. Recomendaciones del proyecto

- No pierda el tiempo, empiece a trabajar desde el primer día que le entregaron la asignación o no le dará tiempo suficiente para terminarlo o bien, terminará muy ajustado.
- Tenga cuidado con respecto a donde escribe los datos de sus archivos .bin, una mala asignación del dispositivo puede ocasionar que los escriba en su disco duro principal corrompiendo el MBR.
- Lea mucho código ajeno, entiendalo hasta la última línea, si a nosotros nos funcionó puede funcionarle a usted.
- Guarde todas las páginas que visita las cuales tengan información valiosa que le fue de utilidad, puede que la necesite luego una vez más y así no perderá tiempo volviendola a buscar.
- Raye, y raye mucho, tome lápiz y papel y haga ejecuciones *a pie*, es una de las mejores maneras de entender por donde se está yendo su flujo de datos.

8. Bibliografía

Averroes. (None de None de None). Portal de la Junta de Andalucía. Obtenido de Portal de la Junta de Andalucía: http://www.juntadeandalucia.es/averroes/ 41009822/inf/dfs

Diego. (25 de 03 de 2010). Instituto Politénico Superior. Obtenido de Instituto Politénico Superior: http://usuarios.fceia.unr.edu.ar/ diegob/taller2/presenta/02-Punteros.pdf

Droid, L. I. (02 de 01 de 2015). Machinary. Obtenido de Machinary: https://machiry.wordpress.com/2012/01/02/mfiber-simple-user-land-thread-library/

Fernández, G. (06 de 06 de 2012). Poesía Binaria. Obtenido de Poesía Binaria: http://totaki.com/poesiabinaria/2012/06/leer-un-archivo-xml-en-c-con-libxml2-con-todos-los-nombres-atributos-y-contenidos/

Javier. (19 de 03 de 2007). CR y SOL. Obtenido de CR y SOL: http://crysol.org/es/node/621

McGraw-Hill, E. (21 de 10 de 2008). MailxMail. Obtenido de MailxMail: http://www.mailxmail.com/curso-informatica-procesos/procesos-c-funcion-sigprocmask

None. (01 de 01 de 2008). universidad Nacional de Santo Antorio Abad del Cusco. Obtenido de universidad Nacional de Santo Antorio Abad del Cusco: http://myslide.es/documents/documentacion-planificacion-por-loteria.html

None. (22 de 07 de 2009). Ejemplos de Java y C Linux. Obtenido de Ejemplos de Java y C Linux: http://www.chuidiang.com/clinux/senhales/senhales.php

None. (01 de 11 de 2009). Sistema Operativo. Obtenido de Sistema Operativo: http://sistemaoperativo.wikispaces.com/Hilos

None. (15 de 01 de 2010). Universidad Nacional de la Plata. Obtenido de Universidad Nacional de la Plata: http://www.ing.unlp.edu.ar/electrotecnia/soyr/material/Pthreads.p

Perepelitsa, C. (22 de 12 de 2013). Quora. Obtenido de Quora: https://www.quora.com/How-do-you-write-a-C-program-to-split-a-string-by-a-delimiter