TEC | Tecnológico de Costa Rica

Proyecto 1

MultiDisplay Animator

Septiembre, 2022

Objetivo

Realizar una re-implementación de algunas de las funciones de la biblioteca de pthreads en Rust del Sistema Operativo GNU/Linux.

Datos Generales

- **Fecha de Entrega:** Jueves 20 de Octubre de 2022 antes de las 23:59:59 GMT-6.
- **Fecha de Revisión:** Martes 25 de Octubre de 2022.
- **Lenguaje:** Rust para GNU/Linux
- **Recurso Humano:** Grupos de 2
- **Valor de la asignación:** 20 %

Profesor

Kevin Moraga
kmoragas@ic-itcr.ac.cr
Escuela de Computación

Introducción

Es posible realizar la implementación de la biblioteca de pthreads en el espacio de usuario, esto permite al estudiante comprender como se puede programar un subsistema del sistema operativo, sin tener que hacer cambios en el kernel. Para ello se re-implementará pthread y se probará utilizando un programa que utiliza la biblioteca creada por el estudiante.

Requerimientos Funcionales

MyPthreads

Se realizará la re-implementación de la biblioteca de pthreads, llamada **mypthreads**, de las siguientes funciones:

- `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 es necesario implementar los siguientes métodos:

- `my_thread_chsched`: Se encarga de cambiar el tipo scheduling del hilo.

Schedulers

Los schedulers se debe de establecer al momento de crear los threads. O sea, es necesario agregar un parámetro a la función `my_thread_create` que establezca el scheduler que utilizará ese hilo. Además este método debe ser compatible con una definición de pthreads estándar, o sea. Es posible compilar cualquier código que utilice phtreads, y que éste utilice la nueva biblioteca.

El scheduler por defecto en caso que no se especifique será *RoundRobin*.

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 inciales.

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.

Animación

El sistema de animación consiste en una especie de Flash para ASCII que permite crear animaciones que corren en un canvas que estará distribuido en varios monitores.

Lenguaje

Se deberá de describir un lenguaje común que permita la descripción de cualquier tipo de animación. En este lenguaje se tendrán conceptos como:

- Restricciones de tiempo de inicio y final
- Establecer el tamaño del canvas
- Descripción de Objetos
- Límite de espacio de Objetos
- Descripción de movimiento de Objetos

Monitores de despliegue

La aplicación permitirá correrse en distintas PCs. Cada computadora aportará su monitor como mecanismo de despliegue. La comunicación entre ellos se realizará utilizando sockets y un protocolo de comunicación definido por el estudiante.

Además es importante establecer el orden de los monitores.

Tamaño del canvas

Se deberá de poder establecer el tamaño del canvas, este tamaño incluye: que sección del canvas corresponde a cual de los monitores. Estos monitores se debieron de asociar previamente a través de la aplicación.

Descripción de Objetos

Tipos Se establecerá un mecanismo para definir el tipo de objeto. El tipo de objeto está directamente relacionado con el tipo de Scheduler a utilizar en este caso se dispondrá de **RoundRobin** y de **Sorteo**.

Forma El lenguaje permitirá la creación de nuevas formas, basadas en ASCII art, que se animarán. Un ejemplo de una posible forma es como se muestra a continuación.

```
| | |x| | |
| |x|x|x| |
|x|x|x|x|x|
| |x|x|x| |
| |x|x|x| |
```

Restricciones de tiempo de inicio y final

Esta descripción consiste en el momento en que el objeto entra en escena y cuando es el momento máximo para que el objeto salga de la escena. En caso que el objeto no respete el tiempo máximo de salida. Este deberá de **explotar**.

Descripción de movimiento de Objetos

Los objetos podrán moverse en cualquier dirección en el canvas, además se podrán realizar rotaciones de 0, 90, 180, 270 grados. Los movimientos consisten en una posición inicial y una posición final. Además de un ángulo inicial y un ángulo final.

Límite de espacio de Objetos

Ningún objeto puede utilizar el espacio, en el canvas, que otro objeto ya posea. Para ello se deben de implementar mutex o semáforos. En caso que el espacio se encuentre ocupado, el objeto deberá esperar hasta que el espacio esté libre.

Creación de una animación de prueba

Se debe de crear una animación de prueba que permita mostrar el funcionamiento de todos los requerimientos anteriores.

Otras consideraciones

Además de las definiciones anteriores tome en cuenta:

- Todo el sistema de Animación es gobernado por el Scheduler de TiempoReal, que hace que se cumplan los límites de tiempo. Este le da más prioridad a un hilo cambiándolo temporalmente de scheduler a uno más eficiente. Por ejemplo se deberá de elegir el sorteo y darle más tiquetes al hilo.
- Todos los animadores corren en modo listening.
- La arquitectura de la federación de monitores es distribuida formando un grafo. Este grafo puede ser: una lista enlazada, en forma de estrella o cualquier otra forma.

Extra

1. Establecer un lenguaje común donde dos proyectos funcionen como si fuera uno. O sea que sea posible la federación de dos proyectos diferentes. (5%)
2. Adaptar la tarea anterior para que utilice mypthreads en lugar de pthreads. (5%)

Requerimientos Técnicos

- El desarrollo se debe de realizar utilizando el lenguaje de programación Rust para GNU/Linux.
- Todo el proyecto debe de funcionar utilizando GNU/Linux.

La sintaxis del animador es:

```
$ animar -c configuración \\
        [-m monitor1,monitor2,...,monitorN]
```

Nótese que el cliente debe de recibir los parámetros desde la terminal. El signo de dolar representa el shell del SO.

Aspectos Administrativos

Entregables

- Código fuente del programa que cumpla los requerimientos funcionales y técnicos.
- Binario del programa, compilado para una arquitectura x86.
- Fuente de la documentación en Markdown o en Latex y luego a PDF.
- PDF con la documentación.

Kick-off

Se deberá realizar un documento para la clase siguiente, con el siguiente contenido:

1. **Introducción**: Debe presentar el problema, utilizando una redacción propia. Incluyendo un pequeño esquema definiendo la estrategia para solucionar el problema asignado.
2. **Ambiente de desarrollo**: Indicar las herramientas que se utilizarán durante la elaboración de la tarea. Incluyendo entorno de desarrollo, forma de debugging, flujo de trabajo en un sistema de control de versiones.
3. **Control de Versiones**: Se debe de incluir y compartir el enlace al repositorio que se utilizará en el control de versiones seleccionado.
4. **Diagrama UML**: diagrama de clases en UML, una explicación de las decisiones de diseño.

Documentación

Las siguientes son las instrucciones para la documentación. NO LA IMPRIMA. Además la documentación se debe de realizar utilizando MD con Latex.

1. **Introducción**: Debe presentar el problema, utilizando una redacción propia, discutido y redactado en el kick-off de la asignación.
2. **Ambiente de desarrollo**: Indicar las herramientas usadas para implementar la tarea.
3. **Estructuras de datos usadas y funciones**: Se debe describir las principales funciones y estructuras utilizadas en la elaboración de esta asignación.
4. **Instrucciones para ejecutar el programa**: Presentar las consultas concretas usadas para correr el programa para el problema planteado en el enunciado de la tarea y para los casos planteados al final de esta documentación.
5. **Actividades realizadas por estudiante**: Este es un resumen de las bitácoras de cada estudiante (estilo timesheet) en términos del tiempo invertido para una actividad específica que impactó directamente el desarrollo del trabajo, de manera breve (no más de una línea) se describe lo que se realizó, la cantidad de horas invertidas y la fecha en la que se realizó. Se deben sumar las horas invertidas por cada estudiante, sean concientes a la hora de realizar esto el profesor determinará si los reportes están acordes al producto entregado.
6. **Autoevaluación**: Indicar el estado final en que quedó el programa, problemas encontrados y limitaciones adicionales. Adicionalmente debe de incluir el reporte de commits de git. Por otro lado, también debe incluir una calificación con la rúbrica de la sección "Evaluación" con cada ítem evaluado de 0 a 10 y "Autoevaluación" con cada ítem evaluado de 1 a 5.
7. **Lecciones Aprendidas** del proyecto: Orientados a un estudiante que curse el presente curso en un futuro.

8. **Bibliografía** utilizada en la elaboración de la presente asignación.
9. Es necesario documentar el código fuente.

Evaluación

- MyPthreads:
  - Scheduler RoundRobin: 5%
  - Scheduler Sorteo: 5%
  - Scheduler en Tiempo Real: 5%
  - Cambio de Scheduler: 5%
  - Funciones de la biblioteca pthreads: 10%
- Documentación utilizando Markdown o Latex-PDF: 20%
- Diseño de lenguaje: 10%
- Implementación de la animación: 20%
- Funcionamiento en Multiples Displays: 20%
- Extra: 5%
- Kick-off: 5% extra o -20% si no se entrega

Si se utiliza la biblioteca **pthreads** en cualquier parte del código, la tarea **no** se evaluará. Para la implementación de hilos se deberá de utilizar necesariamente la biblioteca **myphtreads**

Autoevaluación

Realice una autoevaluación utilizando una escala [*1*] *Muy Malo,* [*2*] *Malo,* [*3*] *Regular,* [*4*] *Bueno y* [*5*] *Muy bueno*, de los siguientes rubros:

[1] [2] [3] [3] [5]	Aprendizaje de Round Robin.
[1] [2] [3] [3] [5]	Aprendizaje de Tiempo Real.
[1] [2] [3] [3] [5]	Aprendizaje de Cambio de contexto.
[1] [2] [3] [3] [5]	Aprendizaje de Sorte.

Aspectos Adicionales

Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones:

1. Sí no se entrega documentación, automáticamente se obtiene una nota de 0.
2. Sí el código no compila se obtendrá una nota de 0, por lo cuál se recomienda realizar la defensa con un código funcional.
3. El código debe ser desarrollado en el lenguaje especificado en los Datos Generales, en caso contrario se obtendrá una nota de 0.
4. Sí no se siguen las reglas del formato del envío a través de Google Drive se obtendrá una nota de 0.
5. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto.
6. Cada grupo tendrá como máximo 20 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cuál se recomienda tener todo listo antes de ingresar a la defensa.
7. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final de la presente asignación.
8. Cada grupo es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberá avisar al menos 2 días antes de la revisión al profesor para coordinar el préstamo de estos.
9. Durante la revisión podrán participar asistentes, otros profesores y el coordinador del área.
10. Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.