DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2: `pthreads`

Escape en Sistemas

13 de noviembre de 2014

Sistemas Operativos

Integrante	LU	Correo electrónico
Rodriguez, Pedro	197/12	pedro3110.jim@gmail.com
Benegas, Gonzalo Segundo	958/12	gsbenegas@gmail.com
Barrios, Leandro Ezequiel	404/11	ezequiel.barrios@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Contents

Parte I Introducción

En el presente trabajo práctico empleamos técnicas de programación paralela. En particular, nos centramos en el diseño y la implementación de un servidor que a través del uso de múltiples hilos de ejecución (`multithreading`) se encarga de atender concurrentemente las peticiones a distintos clientes. Esto lo logramos siguiendo el modelo de *workers* propuesto por la cátedra en el enunciado del TP.

En el TP se nos propone resolver el problema de coordinar adecuadamente la evacuación de un edificio. En el modelo provisto por la cátedra, contamos con un sistema que procesa los pedidos de los clientes de forma secuencial. Y nuestro objetivo es usar técnicas de multiprogramación para procesar estos pedidos de forma paralela. En nuestro modelo, los clientes son personas que están en un espacio rectangular dividido en metros cuadrados. Cada persona está en algún metro cuadrado y cada baldosa tiene un límite predefinido en cuanto a la cantidad de personas que pueden estar en ese espacio al mismo tiempo.

Parte II Desarrollo

Para implementar el `servermulti` nos basamos en el código de `servermono` provisto por la cátedra. Las tareas que tuvimos que realizar para lograr la administración paralela de los pedidos por parte de los clientes fueron:

1. para cada cliente que pida entrar a nuestro sistema, largamos un *thread* que se encargue de la comunicación entre `servermulti` y dicho cliente. Todos estos *threads* comparten el acceso a un mismo bloque de memoria. Es esta la razón por la cuál tuvimos que emplear `mutex` y `locks` para administrar correctamente el acceso a estas posiciones de memoria.

2. cada vez que un cliente pide desplazarse de una casilla del aula a otra, manejar correctamente el acceso a dichas celdas y actualizar correctamente las variables globales que determinan el estado actual del sistema en cada instante

3. manejar correctamente la salida definitiva del edificio de las personas que ya están afuera del mismo. Para esto tuvimos que tener cuidado con, para cada persona que logra salir de la habitación, determinar si hay algún rescatista libre en ese momento para ponerle la máscara y así poder posteriormente (cuando hayan llegado otras cuatro personas junto a ella) salir definitivamente del edificio y salvarse. Si no lo hay, la persona tiene que esperar a que llegue un rescatista y le ponga una máscara antes de poder salvarse definitivamente (podrían haber más de cinco personas fuera del aula pero esperando a que lleguen rescatistas para que le pongan la máscara y recién ahí, cuando hayan cinco personas con máscara, puedan salir).

Un problema importante que nos surgió fue el de evitar *deadlock* cuando dos a y b que están en las casillas A y B piden acceso a B y A respectivamente. Para arreglar este potencial problema, lo que hicimos fue...

Para manejar correctamente el tema de los rescatistas utilizamos dos variables: una para contabilizar la cantidad de personas fuera del edificio **con máscara** y por otro lado la cantidad de personas fuer del edificio pero **esperando máscara**.

Cuando testeamos nuestro código, vimos que el tiempo que tardaban los rescatistas en ponerle la máscara a cada una de las personas era muy corto. Entonces, para que la existencia de rescatistas tenga sentido y su existencia sea considerable cuando corremos nuestro `servermulti`, agregamos un tiempo de tardanza determinado para que cada rescatista tarde en colocar una máscara a una persona.

Parte III Conclusión