Trabajo Práctico Final SO I

Bobe Julio y Grau Marianela

April 21, 2021

1 Introducción

En este informe se explicará todas las experiencias vividas durante todo el desarrolo del trabajo práctico y su funcionamiento.
El proyecto consiste en la resolución del Juego de la Vida de Conway.

2 Ejecución del programa

El programa cuenta con un archivo makefile. Para poder utilizarlo hay que escribir el comando "**make**" creando así el archivo ejecutable.
Luego, para la ejecución del programa debemos poner: **./simulador path/archivo.game**. Esto generará un archivo **path/archivo.final** con la resolución del juego.

3 Estructuras de datos

En esta sección se explicarán las estructuras implementadas durante el desarrollo del trabajo práctico.

3.1 SList

Se implementaron listas simplemente enlazadas (SList) con el fin de guardar las células vivas y sus respectivas células vecinas.
Cada nodo de la SList cuenta con las coordenadas en donde se encuentran en el tablero y su estado en la generación "presente" (X = DEAD, 0 = ALIVE). Estos datos mencionados son representados mediante un char, 'X' o 'O'.
Creamos la funcion SList push con el fin de agregar elementos como si fuese una pila: meto el elemento al principio de la lista simplemente enlazada con la meta de no tener que recorrerla. Esto fue pensado así con el objetivo de que tenga un costo lineal. Misma idea con SList pop. Utilizando las SList evitamos tener que recorrer todo el tablero cada vez que se quisiera analizar que pasaria en una siguiente generacion.

4 Resolución del juego

Explicaremos cómo se resolvió el juego mediante el uso que nosotros le dimos a los hilos, a las barreras y los mutexs.

4.1 Hilos

Usando la devolución de la funcion get_nprocs() y con la ayuda del manual, comprendimos que lo devuelto de get_nprocs() eran los núcleos disponibles sin utilizar del procesador, en nuestro caso, eran 4. Por lo tanto, decidimos crear get_nprocs() hilos para resolver el juego. Este numero nos pareció correcto ya que si se generaran muchos más hilos, no habría "lugar" para poder ejecutarlos a la vez, esto haria que se retrase la ejecución del programa. Caso contrario, con menos hilos tal vez "desperdiciabamos" núcleos del procesador, puesto que cada hilo se ejecutaría en uno de estos, dejando algunos sin "trabajar".
Dichos hilos llevarían a cabo el análisis de las células vivas y las células vecinas a esta, para luego con esa información, ser capaces de crear un nuevo tablero para la siguiente generación.

4.2 Barreras

Al pensar cómo ibamos a hacer para resolver el juego, se nos planteo un problema: si yo luego de analizar el futuro de una célula cambiaba el tablero implicaba que cuando viniese otro hilo a analizar, por ejemplo, una célula cercana a la anterior, el tablero ya estaría modificado y eso no estaba bien. Por lo tanto, decidimos usar barreras.
Dos fueron creadas: una de ella fue para esperar que todos los hilos terminasen de analizar las casillas correspondientes y la otro para esperar a que los hilos terminasen de cambiar el tablero para la siguiente generación.

4.3 Mutex

En este trabajo práctico, los mutex fueron necesarios debido a que en varias partes de nuestro programa podía ocurrir que varios hilos se metiesen en diversas zonas criticas, ocasionando errores en la ejecución, por ejemplo, un mal estado en casillas del tablero.

5 Problemas

Tuvimos varios problemas a lo largo del trabajo práctico. Uno de los que más nos apareció y se nos hizo difícil fue con la barrera. De hecho, no lo pudimos solucionar porque no logramos ver que es lo que falla.
Nosotros implementamos las funciones de las barreras (ver carpeta Barreras) y utilizamos esas mismas en un principio en nuestro programa. Sin embargo, como dijimos arriba, se nos presentaron diversos problemas. Algunos creemos que los fuimos solucionando pero hay algunos que persisten, lo notamos ya que en algunas ejecuciones nos ocurria un deadlock. No logramos encontrar una solucion. Por eso mismo adjuntamos la carpeta con el .c y .h de nuestra implementación para ver si ustedes pueden encontrar el error.
Debido a este problema, usamos las funciones de barreras ya creadas.

6 Bibliografía

A la hora de realizar el trabajo práctico consultamos distintos tipos de bibliografías, estas son las que nos resultaron de ayuda:

- https://linux.die.net/man/3/pthread_cond_init
- https://linux.die.net/man/3/pthread_cond_signal
- https://stackoverflow.com/questions/61647896/unknown-type-name-pthread-barrier-t