Sistemas Operativos I (Juego de la Vida)

por Francisco Alcacer y Jose Cuesta

April 2020

1 Descripción del funcionamiento

La función principal para la ejecución del juego con hilos es congwayGoL que, teniendo en cuenta que guardamos el tablero en un arreglo unidimensional, hace principalmente lo siguiente:

- Trabaja con dos tableros. Esto es, asignamos espacio para un tablero adicional con el objetivo de que los hilos puedan ir actualizando estados de un juego de un tablero en otro, para evitar race conditions o inconsistencias en un estado de juego (que se pueden producir cuando un hilo modifica alguna célula primero, y después viene otro y lee esa célula con un valor que puede ser incorrecto).
- Una vez que tenemos los dos tableros, los hilos comienzan a trabajar sobre el tablero inicial, cada uno en su espacio asignado en el tablero. La funcion con la que trabajan los hilos es work, y la idea es la siguiente:
    1. Leen la información del tablero original y escriben en el tablero adicional la actualización;
    2. Luego, se detienen ante una barrera, esperando a que todos los hilos terminen de escribir en el segundo tablero;
    3. Una vez pasada la barrera, todos los hilos intercambian las referencias de los dos tableros y vuelven a actualizar (ir a 1).
- El tablero resultante puede estar el tablero que se paso como argumento o en el tablero creado en la función, dependiendo de la paridad de la cantidad de ciclos.

El mecanismo de control son entonces las barreras. Sin poner las barreras se producen race conditions, dejando estados incorrectos en cada generación. Algo que también es importante resaltar es que la barrera utilizada es reutilizable. Esto quiere decir que una vez que todos los hilos la pasan, se "vuelve a bajar". No volver a bajar la barrera (i.e. no resetear el contador de hilos esperando a 0) produciría el mismo efecto que no tener barrera después del primer ciclo (o "dejarla levantada"). En cuanto a deadlocks, no nos encontraríamos con ninguno en caso de no implementar ningún mecanismo de control para la sincronización de hilos (sólo ocurrirían race conditions).

2 Estructuras

Utilizamos diferentes estructuras para resolver el problema:

i)

```
struct _board {
    size_t row;   <- Cantidad de filas del tablero
    size_t col;   <- Cantidad de columnas del tablero
    char *tab;    <- Representacion del tablero por medio de
                     un array uni-dimensional
};

Ordenamos los elementos de una fila i y columna j de la forma
tab[col * i + j].
```

ii)

```
typedef struct cond_barrier {
    int count;              <- Limite para la cantidad de hilos
                               en espera
    int waiting;            <- Cantidad de hilos esperando en
                               la barrera
    pthread_mutex_t mutex;  <- Mutex para acceso asincronico a
                               la barrera
    pthread_cond_t cond;    <- Variable de condicion para esperar
} barrier_t;
```

iii)

```
typedef struct _Game {
  Board board;          <- puntero a la estructura _board donde
                           se guarda el tablero
  unsigned int cycles;  <- cantidad de ciclos que se ejecutaran
                           sobre el tablero
} * Game;
```

3 Funciones

Las cabeceras de todas las funciones se encuentran definidas en:
`Board.h`
`Barreras.h`
`Game.h`
Mientras que las implementaciones las podemos encontrar en:
`Board.c`
`Barreras.c`
`Game.c`

4 Uso

Para utilizar el programa, primero ejecutar make. Después, simulacro Ejemplos/Ejemplo.game para ejecutar con algun ejemplo.