Sistemas Operativos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico Número 2

`pthreads` - "SOScrabel"

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	`gonzalo.ciruelos@gmail.com`
Maddonni, Axel Ezequiel	200/14	`axel.maddonni@gmail.com`
Thibeault, Gabriel	114/13	`gabriel.eric.thibeault@gmail.com`

Índice

1. Read-Write Lock

2. Backend Multiplayer

Para realizar la implementación del backend multithreaded se utilizó como base el código del monoplayer y el Read-Write Lock implementado por nosotros.

El primer problema que aparece es al atender las conexiones. Para eso, se debió delimitar un máximo de jugadores que pueden estar jugando al juego en simultaneo. Para eso, vamos a escuchar como máximo esa cantidad de conexiones.

Para el tablero de letras usamos un rwlock para cada casillero, permitiendo asi que dos usuarios escriban simultaneamente 2 casilleros distintos. Por eso, debimos utilizar una matriz de rwlock's, en la que cada posición de la matriz se corresponde con la misma posición del tablero de letras.

Sin embargo, para el tablero de palabras la situacion es distinta, y tomamos la decisión de tener un único lock para todo el tablero, dado que si tenemos un lock por casillero, pueden pasar cosas como que un jugador haga update mientras otro esta escribiendo una palabra y que al que hizo update le llegue solo media palabra.

Haciendo lock del tablero entero nos aseguramos que cuando un jugador escribe una palabra, estamos seguros de que lo va a terminar de hacer antes de que otro jugador lea los contenidos del tablero de palabras. Debido a esto, además debimos tener en cuenta que el lock se debe hacer durante toda la escritura de una palabra, es decir, que la escritura de una palabra entera debe verse como *atómica* desde afuera.

Otra modificación –muy menor– que hubo que realizar, fue la de la finalización del atendedor. Ahora, como cada jugador se atiende con threads distintos, la finalización de un thread debe tratarse de manera particular para cada thread, por lo que cuando un jugador cierra su conexión o termina su partida, se debe llamar a `pthread_exit`, en vez de a `exit` simplemente, como está en la versión monojugador.