

K213086

OS ASSIGNMENT 3

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>


#define ROWS 10

#define COLS 10


int grid[ROWS][COLS]; // global grid of cells

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER; // mutex for grid access

pthread_cond_t cond = PTHREAD_COND_INITIALIZER; // condition variable for synchronization


// helper function to print the grid

void print_grid() {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            if (grid[i][j] == 0) printf("0");
            else printf("1");
        }
        printf("\n");
    }
    printf("\n");
}


// helper function to count the number of live neighbors for a given cell

int count_neighbors(int i, int j) {
```

```

int count = 0;
for (int x = i-1; x <= i+1; x++) {
    for (int y = j-1; y <= j+1; y++) {
        if (x == i && y == j) continue;
        if (x < 0 || x >= ROWS || y < 0 || y >= COLS) continue;
        if (grid[x][y] == 1) count++;
    }
}
return count;
}

```

// thread function to update the grid for a range of rows

```

void* update_grid(void* args) {
    int start_row = *((int*)args); // start row for this thread
    int end_row = start_row + ROWS/2; // end row for this thread
    while (1) {
        // wait for the main thread to signal that it's safe to update the grid
        pthread_mutex_lock(&mutex);
        pthread_cond_wait(&cond, &mutex);
        pthread_mutex_unlock(&mutex);

        // update cells for this range of rows
        for (int i = start_row; i < end_row; i++) {
            for (int j = 0; j < COLS; j++) {
                int neighbors = count_neighbors(i, j);
                if (grid[i][j] == 1 && (neighbors < 2 || neighbors > 3)) {
                    grid[i][j] = 0;
                }
                else if (grid[i][j] == 0 && neighbors == 3) {

```

```

        grid[i][j] = 1;
    }
}
}
}
}

```

```

int main() {
    // initialize the grid with random values
    srand(time(NULL));

    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            grid[i][j] = rand() % 2;
        }
    }

    // create four worker threads to update the grid
    pthread_t threads[2];
    int thread_args[2];
    for (int i = 0; i < 2; i++) {
        thread_args[i] = i * ROWS/2;
        pthread_create(&threads[i], NULL, update_grid, &thread_args[i]);
    }

    // main loop to print the grid and update it
    int iterations = 0;
    while (iterations < 20) { // exit loop after 100 iterations
        print_grid();
    }
}

```

```

// signal the worker threads to update the grid

pthread_mutex_lock(&mutex);

pthread_cond_broadcast(&cond);

pthread_mutex_unlock(&mutex);

usleep(100000); // sleep for 100ms

iterations++;

}

return 0;
}

```

```

k213086@k213086-virtual-machine:~$ gcc -o a assi
k213086@k213086-virtual-machine:~$ ./a
00110100000011111111
00110011111110011001
00111011100000100011
10011000100101100011
01101001110010101010
01110010000011010001
11111110100110000101
10110000001010111010
10000000011111011011
10101010111011011111
01110111000110100101
01110000111100000110
00110010110111011010
01101011101100001110
11000010110100110011
10111100110000001100
01101001110011101100
01001010110001010001
11110011001011101011
11101111000111011011

00111110101011101111
01000000001010001000
01011111111010110101
10000000000100101001
10100001100100101111
01110010000011010001
11111110100110000101
10110000001010111010
10000000011111011011
10101010111011011111
10001010000010110001
01011000101000000111
00001010000010010001
10000010101111001101
10010010000101110000
10111100110000001100
01101001110011101100
01001010110001010001
11110011001011101011
11101111000111011011

```

```
k213086@k213086-virtual-machine:~$ ./ass1.sh
```

```
Running with 1 processes
```

```
real    0m0.004s
```

```
user    0m0.004s
```

```
sys     0m0.001s
```

```
Running with 2 processes
```

```
real    0m0.004s
```

```
user    0m0.004s
```

```
sys     0m0.000s
```

```
k213086@k213086-virtual-machine:~$ gedit ass1.sh
```

