Lab3.md

Lab 3

Objectives

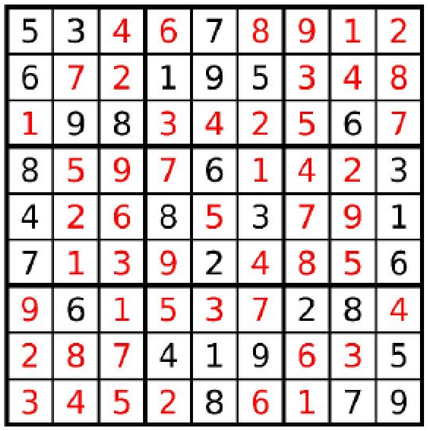
- Implement a multithreaded program that solves a Sudoku board.
- Gain experience solving a problem with parallel code.
- Gain experience using locking mechanisms like mutex locks.

Important Notes

- Work in your designated lab group
- This is the **assignment** version of the lab created for the 2019 Winter term due to a lab cancellation.

Lab Activity

A Sudoku puzzle uses a 9 × 9 grid in which each column and row, as well as each of the nine 3 × 3 subgrids, must contain all of the digits 1 ... 9. The Figure presents an example of a solved Sudoku puzzle.



5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

A solved Sudoku puzzle needs to have each of the digits 1 through 9 in:

- each row of the sudoku board
- each column of the sudoku board
- each of the smaller 3x3 sub-grids (shown in the picture using darker lines)

There are several ways of multithreading this application. A good starting point is to define three functions:

- A function which solves a row, provided only one number is missing
- A function which solves a column, provided only one number is missing
- A function which solves a grid, provided only one number is missing

These functions may all be run as threads using `pthread_create`. You may want to create an additional function which checks every few seconds if the board is solved (has no missing numbers).

Remember, a valid Sudoku board only has one solution!

Your solution will be evaluated into one of the following categories:

Level 1 - up to 80%

```
The program can solve a Sudoku board where there is always enough information
in a single row/grid/column to find the next number; i.e. using only the
functions described above.
```

This solution is possible using only the functions described above.

Level 2 - up to 100%

```
The program can solve any valid Sudoku board.
```

Note that if you wish to achieve the Level 2 solution, you will need to have communication between threads. A good approach is to keep a record of which numbers are possible for each square, and initialize it to contain all possible values (1-9 inclusive) for empty squares. You may use a 2D or 1D array representing the grid, where each value is an additional array of integers or characters that the square may contain)

Passing Parameters to each Thread

You can pass arbitrary data to a thread by using the void pointer (`void *`) argument to `pthread_create`. This pointer can point to any data and is passed along to the thread function as the first parameter.

It is recommended to use this feature to tell your thread what section it is working on. (ex: `my_thread_row` gets passed `int* 2` to work on row 3)

For example, if you want to pass an integer to a thread, you can do something like:

```
int *x = malloc(sizeof(int));
*x = 1;
pthread_t thread;
if (pthread_create(&thread, NULL, my_thread_main, x) != 0) {
        // Handle pthread_create error...
}
```

Your thread would then be able to read the value and free the memory.

```
void *my_thread_main(void *x) {
        int value = *((int*)x);
        free(x); x = NULL;
        // Do something with value.
}
```

Returning Results to the Parent Thread

Threads of a process all have access to the same heap memory. This means threads can communicate their results by sharing memory.

Reading the Puzzle

Your Sudoku puzzle solver must read in a file called puzzle.txt which contains the Sudoku puzzle to solve. Entries that are `-` are empty squares.

Here's an example:

```
5 3 - - 7 - - - -
6 - - 1 9 5 - - -
- 9 8 - - - - 6 -
8 - - - 6 - - - 3
4 - - 8 - 3 - - 1
7 - - - 2 - - - 6
- 6 - - - - 2 8 -
- - - 4 1 9 - - 5
- - - - 8 - - 7 9
```

Deliverables

Notice Lab project will be due on the posted date on Blackboard and will contain the following:

1. All source files for your implementation, include a makefile so that the code can be compiled. The solution must make use of threading in order to receive marks.
2. Your program MUST read from an input file in the working directory called `puzzle.txt`, which will contain a valid Sudoku board.
3. Please **do not** include compiled binaries in your submission. If binaries are found, they will be removed before evaluation and you may lose marks.