



Assignment 2

[20 marks]

Submission Deadline: **FRIDAY, OCTOBER 26, at 11:59PM**

ASSIGNMENT TASK

Design a multithreaded Sudoku Solution Checker program that accepts a Sudoku solution from the user as input, determines whether the solution is valid or invalid. If invalid, it also specifies which rows, columns or 3x3 subgrids have errors.

You may complete this assignment individually or in groups of 2 students. Only one submission per group is required.

[Note: While Sudoku is fun, you do not need to be a Sudoku player to complete this project]

ASSIGNMENT DETAILS:

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 $1 \dots 9$. Figure below presents an example of a valid Sudoku puzzle.

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

You need to accept the user input from a file. A sample input file sud_sol.txt is provided.

There are several different ways of multithreading this application. One suggested strategy is to create threads that check the following criteria:

- A thread to check that each column contains the digits 1 through 9
- A thread to check that each row contains the digits 1 through 9
- Nine threads to check that each of the 3×3 subgrids contains the digits 1 through 9

This would result in a total of eleven separate threads for validating a Sudoku puzzle. However, you are welcome to create even more threads for this project. For example, rather than creating one

thread that checks all nine columns, you could create nine separate threads and have each of them check one column.

Passing Parameters to Each Thread

The parent thread will create the worker threads, passing each worker the location that it must check in the Sudoku grid. This step will require passing several parameters to each thread. The easiest approach is to create a data structure using a struct. For example, a structure to pass the row and column where a thread must begin validating might appear as follows:

```
/* structure for passing data to threads */
typedef struct
{
    int row;
    int column;
} parameters;
```

Pthreads programs will create worker threads using a strategy similar to that shown below:

```
parameters *data = (parameters *) malloc(sizeof(parameters));
data->row = 1;
data->column = 1;
/* Now create the thread passing it data as a parameter */
```

The data pointer will be passed to the pthread_create() function, which in turn will pass it as a parameter to the function that is to run as a separate thread.

Returning Results to the Parent Thread

Each worker thread is assigned the task of determining the validity of a particular region of the Sudoku puzzle. Once a worker has performed this check, it must pass its results back to the parent. One good way to handle this is to create an array of integer values that is visible to each thread. The i^{th} index in this array corresponds to the i^{th} worker thread. If a worker sets its corresponding value to 1, it is indicating that its region of the Sudoku puzzle is valid. A value of 0 would indicate otherwise. When all worker threads have completed, the parent thread checks each entry in the result array to determine if the Sudoku puzzle is valid.

SUBMISSION DETAILS:

Upload a **single zip** file containing:

1. your code (consisting of your c file),
2. an image file (.jpg or .png) showing a screen capture of your program in action, and
3. a readme.txt file containing names of the group members, specific contribution by each group member, and the compile string including any instructions to compile your program)

to the Assignment 2 dropbox on Brightspace. Do NOT submit the executable.

SUBMISSION DEADLINE:

Friday, 26-October-2018, 11:59PM

GRADING CRITERIA:

- Code compiles using gcc on Bluenose. Assignment instructions are properly followed.
 - Note that you will get a zero in this assignment if your code does not compile
 - Also note that marks will be deducted for any warnings generated during compilation.
- Your application uses 11 or more threads – at least nine for the subgrids and one each for rows and columns. It may use more than 11 threads. [8]
- Sudoku solution is correctly checked as being valid or invalid. [2]
- In case of invalid solution, rows, columns and subgrids with errors are displayed. [4]
- Return values for system calls are checked and errors handled (if any). [2 marks]
- Clean up is done, i.e., threads are properly joined, and any memory allocated is properly deallocated (if required). [2 marks]
- Code is well written, properly formatted and commented. Readme.txt file is provided with the correct compilation string and specific contribution made by group members (if applicable). [2 marks]

Note that in the absence of a compilation string in the Readme.txt file, your submission will be compiled using the standard string `gcc -o A2 A2.c` (or your filename), and you will receive a zero grade if it fails to compile.