Assignment #3

Hide Assignment Information

Instructions

Assignment overview

A Sudoku puzzle uses a 9 × 9 grid which follows the following three rules:

1. Each column must contain all of the digits 1-9 with no repetition
2. Each row must contain all of the digits 1-9 with no repetition
3. Each 3 × 3 subgrids must contain all of the digits 1-9 with no repetition

Here is an example of a valid Sudoku puzzle:

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

Here is an example of an invalid Sudoku puzzle (Error is highlighted):

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

For this assignment, write a C program which will accept a command-line parameter indicating a file that contains a 9 × 9 grid of digits. Your program will read this file and use multithreading to quickly verify to the user whether or not this is a valid Sudoku puzzle.

Purpose

The goals of this assignment are the following:

- Learn how to use multi-threading and data parallelism to accomplish tasks faster
- Get experience with the `pthread_create()` and `pthread_join()` system functions
- Gain more experience with the C programming language from an OS perspective

Computing platform

You are welcome to develop your program on your own workstation if you wish, but you are responsible for ensuring that your program compiles and runs without error on the Gaul computing platform. Marks will be deducted if your program fails to compile, or your program runs into errors on Gaul.

- Gaul tips

Instructions

You will write a program called `assignment-3.c`. This program will:

- Accept a command-line parameter
    1. The command-line parameter indicates the name of the file to open
    2. Here is a sample. You can assume that the 9 x 9 grid only contains the digits 1-9 and are separated by whitespace.

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

- Your program will then employ data parallelism to divide the task of checking each row, column, and subgrid
    1. You must use 28 threads (27 "child" threads and the "parent" process represents the 28th thread)
        1. Nine threads will check that all 9 rows are valid
        2. Nine threads will check that all 9 columns are valid
        3. Nine threads will check that each of the 9 subgrids are valid
- Your program will then gather the results from all 27 threads and report back whether or not this is a valid Sudoku solution. If it is an invalid solution, your program will indicate where the error was found.

Output

For example, executing `./assignment-3 valid-sudoku.txt` should produce the following output (Order may not be sequential):

```
Thread # 1 (subgrid 1) is valid
Thread # 2 (subgrid 2) is valid
Thread # 3 (subgrid 3) is valid
Thread # 4 (subgrid 4) is valid
Thread # 5 (subgrid 5) is valid
Thread # 6 (subgrid 6) is valid
Thread # 7 (subgrid 7) is valid
Thread # 8 (subgrid 8) is valid
Thread # 9 (subgrid 9) is valid
```

```
Thread # 10 (row 1) is valid
Thread # 11 (row 2) is valid
Thread # 12 (row 3) is valid
Thread # 13 (row 4) is valid
Thread # 14 (row 5) is valid
Thread # 15 (row 6) is valid
Thread # 16 (row 7) is valid
Thread # 17 (row 8) is valid
Thread # 18 (row 9) is valid
Thread # 19 (column 1) is valid
Thread # 20 (column 2) is valid
Thread # 21 (column 3) is valid
Thread # 22 (column 4) is valid
Thread # 23 (column 5) is valid
Thread # 24 (column 6) is valid
Thread # 25 (column 7) is valid
Thread # 26 (column 8) is valid
Thread # 27 (column 9) is valid
valid-sudoku.txt contains a valid solution
```

Executing `./assignment-3 invalid-sudoku.txt` should produce the following output (Order may not be sequential):

```
Thread #  1 (subgrid 1) is valid
Thread #  2 (subgrid 2) is valid
Thread #  3 (subgrid 3) is valid
Thread #  4 (subgrid 4) is valid
Thread #  5 (subgrid 5) is valid
Thread #  6 (subgrid 6) is valid
Thread #  7 (subgrid 7) is valid
Thread #  8 (subgrid 8) is valid
Thread #  9 (subgrid 9) is valid
Thread # 10 (row 1) is valid
Thread # 11 (row 2) is INVALID
Thread # 12 (row 3) is INVALID
Thread # 13 (row 4) is valid
Thread # 14 (row 5) is valid
Thread # 15 (row 6) is valid
Thread # 16 (row 7) is valid
Thread # 17 (row 8) is valid
Thread # 18 (row 9) is valid
Thread # 19 (column 1) is valid
Thread # 20 (column 2) is valid
Thread # 21 (column 3) is valid
Thread # 22 (column 4) is valid
Thread # 23 (column 5) is valid
Thread # 24 (column 6) is valid
Thread # 25 (column 7) is valid
Thread # 26 (column 8) is valid
Thread # 27 (column 9) is valid
invalid-sudoku.txt contains an INVALID solution
```

Helpful hints

- To read from a file, you should use the relevant functions in the stdio.h library such as `fopen`, `fscanf`, and `fclose`

Submitting

When you are finished your assignment, follow these steps:

1. **Cite your work:** Did you use code that you did not write yourself? Just like an essay, if you did not write it yourself, you must properly attribute the author. Leveraging existing code for your assignment is acceptable but it should represent a minor portion of your submission. Specifically, any code that fulfills the learning outcomes detailed in the Purpose section above should be almost entirely your own. Contact the course instructor if you are unsure.

    1. If you used sample code from the course notes or the textbook, in the comments, state which sample program you used or which page(s) from the lecture notes you used.
    2. If you used code from an online source (eg. StackOverflow, GeeksForGeeks, etc.), in the comments, provide the link where the code was found.
    3. If you used an LLM such as ChatGPT, in the comments, provide the link to the prompts you used to generate the code.
    4. Using code from another student is a violation of the policy on academic integrity and is not permitted.
2. Verify your code runs on Gaul. Create screenshots demonstrating your program and place them in the `Assignment-3` directory. Remove any spurious files (eg. the executable version of your program). The directory should have a single file called `assignment-3.c` and any screenshots. Supporting files like `Makefiles` or `README`s are acceptable.
3. If necessary, run the following command to get out of the `Assignment-3` directory: `cd ..`
4. Package your assignment into a tarball: `tar -cvf Assignment-3.tar Assignment-3`
5. Verify the contents of your tarball (`tar -tvf Assignment-3.tar`) (`du -sh Assignment-3.tar`). The tarball should have your `assignment-3.c` file and all screenshots. **If your tarball is 10kb in size** you have an empty tarball and you made an error on this step. Make sure you are properly creating your tarball with the right files in it.
6. Use an SFTP program to download the tarball to your local workstation and then upload it to OWL.

Due on Mar 4, 2025 11:55 PM

Available on Feb 12, 2025 12:00 AM. **Access restricted before availability starts.**

Available until Mar 6, 2025 11:55 PM. **Access restricted after availability ends.**

Show Rubrics

Submit Assignment

Allowed File Extensions

tar

Files to submit

(0) file(s) to submit

After uploading, you must click Submit to complete the submission.

Add a File **Record Audio** **Record Video**

Comments

Submit **Cancel**