CP386: Assignment 3 – Fall 2024

Due on Nov 4, 2024 (Before 11:59 PM)

This is a group (of two) assignment, and we will practice the concept of multi-threading, process scheduling, and some related system calls.

General Instructions:

- Teamwork is encouraged, but you can complete this assignment individually, too.
- For this assignment, you must use C language syntax. Your code must compile using make without errors. You will be provided with a Makefile and instructions on using it.
- Test your program thoroughly with the GCC compiler in a Linux environment.
- If your code does not compile, then you will score zero. Therefore, ensure you have removed all syntax errors from your code.
- GitHub Classroom-based repository will be used to track your work and its split with your partner daily. Follow the steps described at the end of the document to create your repository.
- Gradescope platform would be used to upload the assignment file(s) for grading. The link to the Gradescope assignment is available on Myls course page. The link to the Gradescope assignment is available on Myls course page. For submission, connect the GitHub Classroom repository and select the branch master to submit on Gradescope. Ensure your file name is as suggested in the assignment; using a different name may score Zero. If working in the group, make sure only one person from the group is uploading the files to Gradescope; if both have submitted, your files will be flagged for plagiarism. Kindly upload the files with caution.
- Please note that the submitted code will be checked for plagiarism. By submitting the code file(s), you would confirm that you have not received unauthorized assistance in preparing the assignment. You also ensure that you are aware of course policies for submitted work.
- Marks will be deducted for any questions where these requirements are not met.
- Multiple attempts will be allowed, but only your last submission before the deadline will be graded. We reserve the right to take off points for not following directions.

Question 1

Write a C program to validate the solution to the Sudoku puzzle. A Sudoku puzzle uses a 9 × 9 grid in which each column and row, as well as each of the nine 3×3 sub-grids, must contain all the digits 1 ··· 9. Figure below presents an example of a valid 9 x 9 Sudoku puzzle solution. This program implements a multithreaded application that reads a Sudoku puzzle solution from a file ***("sample_in_***sudoku***.txt")*** and validates it.

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

There are several different ways of multi-threading this application (students are encouraged to explore). 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 sub-grids contain 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 problem. 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.

1. 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 would appear as follows:

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

   Pthreads 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's create()` (Pthreads) function, which in turn will pass it as a parameter to the function that is to run as a separate thread.

2. 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 indicates 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.

The expected output for this question is:

```
Sudoku Puzzle Solution is:
2 7 6 3 1 4 9 5 8
8 5 4 9 6 2 7 1 3
9 1 3 8 7 5 2 6 4
4 6 8 1 2 7 3 9 5
5 9 7 4 3 8 6 2 1
1 3 2 5 9 6 4 8 7
3 2 5 7 8 9 1 4 6
6 4 1 2 5 3 8 7 9
7 8 9 6 4 1 5 3 2
Sudoku puzzle is valid
```

Note: When submitting source code file for this question, name it like:

- `sudoku.c`

Question 2

Write a C program to find the average waiting time and turn-around time for the pre-defined set of tasks. The program will read a predefined set of tasks from file "sample_in_schedule.txt" and will schedule the tasks based on the First Come, First Served scheduling algorithm (non-preemptive). The columns in "sample_in_ schedule.txt" present the thread ID, its arrival time, and its burst time, respectively, with the following example format:

```
1,        0,        3
2,        1,        2
3,        2,        1
4,        3,        4
```

Thus, thread T2 has arrived at 0 milliseconds and a CPU burst of 3 milliseconds, and so forth. There are several different ways of developing this program (students are encouraged to explore). You can use array's to hold the information and compute the results. But the easiest approach may be to create a data structure using a struct that may appear as follows:

```
struct threadInfo{
        int p_id;
        int arr_time;
        int burst_time;
        int waiting_time;
        int turn_around_time;};
```

Once the arrival time and burst times of threads have been read from the file, the turn-around time and the threads' waiting time are calculated using the following formula.

- `Turn-around_time = thread_completion_time - thread_arrival_time`
- `Waiting_time = Turn-around_time - Thread_burst_time`

The average waiting time for all threads is determined by summing all the threads' respective waiting times and dividing the sum by the total number of threads. Similarly, the average turn-around time is calculated summing the respective turn-around time of all the threads and divided the sum by the total number of threads.

The expected output for this question is:

```
Thread ID       Arrival Time    Burst Time      Completion Time Turn-Around Time        Waiting Time
1                       0               2               2               2               0
2                       1               6               8               7               1
3                       2               4               12              10              6
4                       3               9               21              18              9
5                       6               12              33              27              15
The average waiting time: 6.20
The average turn-around time: 12.80
```

Note: When submitting source code file for this question, name it like:

- `fcfs.c`

GitHub Repository Creation Instructions:

We will use a GitHub Classroom-based repository to keep track of the assignment work. To submit a group assignment project on GitHub Classroom, follow these steps:

- Make sure that each member of the group has a GitHub account.
- Use the link https://classroom.github.com/a/Ete2W38i to access the assignment.
- Once you have accessed the assignment, you will see a green button that says, "Accept this assignment."

Click on the button to create a repository for your group. Carefully select your name.

- Next, you will be prompted to choose a team for your group. If your team already exists, select it. Otherwise, click the "Create a new team" button to create a new team for your group. Include your teammate in the new team (only one of you will be creating the team). The team's name must be your first name and your teammate's first name, e.g., “firstname1-firstname2”. Only one of you will create a team (named as mentioned), and the other will join it. Caution: Do note create a random name of your team.
- After creating the repository (by default, Private and keep it as is), each group member must clone it to their local machine using Git. To do this, navigate to the repository's page and click the green "Code" button. Copy the HTTPS link and use it to clone the repository. You can use GitHub Desktop on your local OS to manage the GitHub repository better.
- Work together to complete the assignment, ensuring everyone contributes and regularly pushing your work onto GitHub. Your contributions and division of the work will be graded based on the repo activities. If your GitHub account is missing from the commits/contributions list, you will get zero for the assignment.
- Keep pushing your changes daily to GitHub so I can check your progress. Once the assignment is complete, push the changes to the repository using Git.
- When you are ready to submit the assignment, navigate to the assignment page on Gradescope and click the "Submit assignment" button. This will prompt you to select the repository that you want to submit. Select the repository that your group created.
- Now, confirm that you want to submit the assignment. You will be asked to provide a comment explaining your submission. Provide relevant details and click on the "Submit assignment" button.
- In the final step, visit Gradescope and select the project assignment. It will ask you to connect your GitHub Account. Once authorized, you can select the repository you were working with and select the branch master to submit on Gradescope. Do not forget to add your assignment partner to the Gradescope Assignment.
- That's it! Your group assignment has now been submitted to GitHub Classroom. I and marker will be able to review your submission and provide feedback.