Programming Project 4

OS2025-SWU-软工中外 34 班

Spring 2025

Experimental topic

Programming Projects in Chapter 4

Page P-25 to P-26 of Operating System Concepts (10th) or this pdf

Experimental resources

/resources/code_for_project/code_for_project4

Experimental report naming format:

StudentID-name-Project-X (e.g., 2220183211-张三-Project-1)

Experimental objectives

1. (Course Objective 2) Master the basic concepts, design ideas and status of the process. Understanding the basic principles and ideas of process synchronization and mutual exclusion. Use the basic means of realizing process synchronization and mutual exclusion to analyze and solve the basic problems of process synchronization and mutual exclusion.
2. (Course Objective 3) Master the basic concepts and design ideas of the computer file system. Have the ability to write simple codes to read or write a file.

Experimental report submission

> **Attention**
>
> Please upload your report with naming format to the ftp server within **two weeks** (Deadline:).

1 Sudoku Solution Validator

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 $1 \cdots 9$. `fig-p4-1` presents an example of a valid Sudoku puzzle. This project consists of designing a multithreaded application that determines whether the solution to a Sudoku puzzle is valid.

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

Fig. 1.1: Solution to a 9 × 9 Sudoku puzzle.

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.

2 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;
```

Both Pthreads and Windows 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 either the `pthread_create()` (Pthreads) function or the `CreateThread()` (Windows) function, which in turn will pass it as a parameter to the function that is to run as a separate thread.

3 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.

4 提交说明

需要提交程序，能够从 *puzzle/* 文件夹下读取 *puzzle/puzzle-[1-1000].txt* 文件，对于每个文件，都是一个需要验证的矩阵。

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

需要通过你的程序给出每个文件对应的合法 (输出：1) 或者非法（输出：0)。

输出到 *学号-output.txt* 文件夹中，格式为 *序号,1* `或者`*序号,0* 。

例如: *2220183211-ouput.txt* , 输出为

```
1,1
```