COMP 3500 Introduction to Operating Systems
Project 3 – Banker's Algorithms

Points Possible: 100 due: 11:59 pm Thursday Oct 12th, 2023

There should be no collaboration among students. A student shouldn't share any project code with any other student. Collaborations among students in any form will be treated as a serious violation of the University's academic integrity code.

Objectives:

1. Complete the source code to simulate producer/consumer problem.
2. Understand the basics of the POSIX thread library.
3. Build a binary program on a Linux machine.
4. Run the program and record the result of the simulation.

Requirements:

- Each student should **independently** accomplish this project assignment. You may discuss with other students to solve the coding problems.
- To embark on this project, you may choose one of the following four options.
    - Important! Option 1: For Mac and Linux users, use SSH to connect to a remote Linux server. Please read files in "tutorial" on Canvas for details.
    - Important! Option 2: For Window 10 users, Please read "enviro_tutorial", "Putty Tutorial" and "Win subsystem Linux" for details.
- Important! Read the I/O format specification carefully and follow. It is very important to your final grade of this project!
- You are highly recommended to use a Linux operating system.

1. Introduction to banker's algorithm

Banker's Algorithm is a resource allocation and deadlock avoidance algorithm. This algorithm test for safety simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

In simple terms, it checks if allocation of any resource will lead to deadlock or not, OR is it safe to allocate a resource to a process and if not then resource is not allocated to that process. Determining a safe sequence(even if there is only 1) will assure that system will not go into deadlock.

Banker's algorithm is generally used to find if a safe sequence exist or not. But here we will determine the total number of safe sequences and print all safe sequences.

The data structure used are:

- Available vector
- Max Matrix
- Allocation Matrix
- Need Matrix

Here gives the example input :

Process	Max			Allocation			Available			Needed		
	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
P0	7	5	3	0	1	0	3	3	4	7	4	3
P1	3	2	2	2	0	0				1	2	2
P2	9	0	2	3	0	2				6	0	0
P3	2	2	2	2	1	1				0	1	1

In the above case, we have 4 process(p0, p1, p2, p3) and three types of resources. The needed matrix is

Needed[i][j] = Max[i][j] – Allocation[i][j]

The “Available” column shows current remaining available resources. As you can see, if we can not start P0 or P2 as our next task because not enough availale resources. A possible safe sequence should start P1 or P3. In fact, we have four possible safe sequences:

1. P1–> P3–> P0–> P2
2. P1–> P3–> P2–> P0
3. P3–> P1–> P0–> P2
4. P3–> P1–> P2–> P0

2. Example walk through

When you run this program, you should have the following results if you use the input of example from section 1:

m: input the number of processes
4
n: input the number of types of resources
3
Max Matrix: input the value of each element left to right, top to bottom
7
5
3
3
2

2
9
0
2
2
2
2
Allocation Matrix: input the value of each element left to right, top to bottom
0
1
0
2
0
0
3
0
2
2
1
1
Available Vector: input the value of each element left to right
3
3
4
The system is safe
The safe sequence is:
1-->3-->0-->2
A process start to request resources. The process number is:
1
Each type resource requested by this proess
1
2
2
The system is safe
The safe sequence is:
1-->3-->0-->2
Request approved!
Do you want to try again? Yes press y/Y, otherwise press any other key to quit
y
A process start to request resources. The process number is:
0
Each type resource requested by this proess
7
4

3
Requested more than available, please re-try!
Requested more than available, please re-try!
Requested more than available, please re-try!
The system is not safe
Your request is declined!
Do you want to try again? Yes press y/Y, otherwise press any other key to qui

As we expected, if we let process 1 be the next task, allocate all resource it needed, 1, 2, 2, then the request will be approved by the banker, but if we use process 0, and request resources 7, 4, 3, then the request will be declined.

3. Follow the Format Specification (10 Points)

In the source file "Firstname_Lastname.cpp", you will find first four comment lines:

```
// #0#BEGIN# DO NOT MODIFY THIS COMMENT LINE!
// Firstname
// Lastname
// #0#END# DO NOT MODIFY THIS COMMENT LINE!
```

Your first task is modifying the two lines **between** the beginning line and end line. Change them into your first name and last name. Remember the strings are case-sensitive, so capitalize the first letter of the word and follow exactly the syntax of the example.

You can see lots of similar code blocks in the file. You are free and supposed to fill your answer between those special beginning and ending comment lines. You can insert and edit multiple lines between special comment lines in anyways, however(Important!), as the comment indicated, do not modify the special begin and comment lines **themselves**!

Let's do the second task. Scroll down to the bottom of the file and find those lines (press "shift + g" if you are using vi/vim):

```
// #8#BEGIN# DO NOT MODIFY THIS COMMENT LINE!
banner_id = 903900281;
// #8#END# DO NOT MODIFY THIS COMMENT LINE!
```

Look at your student ID card, check your banner ID. Again, change the "banner_id" value to your own ID. Your unique student id will be compiled into the program, and the input of the experiment also uniquely depends on your ID.

Warning: Since every student has a unique id number, the later compiled binary file is also unique. Copy binary file from other students will be easily detected!

3. Complete Source Code (70 Points)

Read the source code, try to understand the function of each line of code. Follow the instructions in the comments Firstname_Lastname.cpp and insert proper code into the rest 7 blocks to implement a producer/consumer model.

4. Run and Record Simulation Result (10 Points)

Compile your source code into a binary program. For example, use following command to include the pthread library:

```
$ g++ Firstname_Lastname.cpp -o Firstname_Lastname
```

After you compile the c source code successfully, please use the script command to record the running result of the program Firstname_Lastname:

```
$ script Firstname_Lastname.script
Script started, file is Firstname_Lastname.script
./Firstname_Lastname
```

After you run the program, you will have the following results:

```
Banner id: 903900281
...
```

You should have similar result and difference in the banner ID. Then exit recording and save it into typescript file "Firstname_Lastname.script" by the following command:

```
$ exit
exit Script done, file is Firstname_Lastname.script
```

Warning: Since every student has a unique id number, the result of the simulation is also unique. Copy simulation results from other students will be easily detected!

5. Deliverables (10 Points)

Since you have generated multiple script files, please save all the script files in one directory. Create a folder "Firstname_Lastname" first. Please make sure the **folder name** correct. You can use the command mv to rename the folder:

```
$ mv folder-name Firstname_Lastname
```

Make sure all the three files are into this folder. You should have those following files inside the folder:

1. commands recording file: Firstname_Lastname.script
2. executable binary file: Firstname_Lastname
3. source code file: Firstname_Lastname.cpp

Achieve all the folder into a single tarred and compressed file with a tar command.

```
tar -zcvf Firstname_Lastname.tar.gz Firstname_Lastname
```

You need to submit one tarred file with a format: Firstname_Lastname.tar.gz

Grading Criteria:

1. Follow the format specification: 10%
    a. Do not break the special comments.
    b. Input your name properly (mark #0).
    c. Input your banner id properly (mark #8).
2. Complete the source code: 70%
    a. Each blank is worth 10, total 70 (mark #1 ~ #7).
    b. See detailed specification for each blank in the source code file.
3. Compiling and running result: 10%
    a. Compile the code successfully.
    b. Record the running results.
4. Deliverables: 10%
    a. Contains all the files.
    b. Naming all the files properly.

Rebuttal Period:

You will be given **two business days** to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.