**Rafael Garcia**

**04/19/2019**

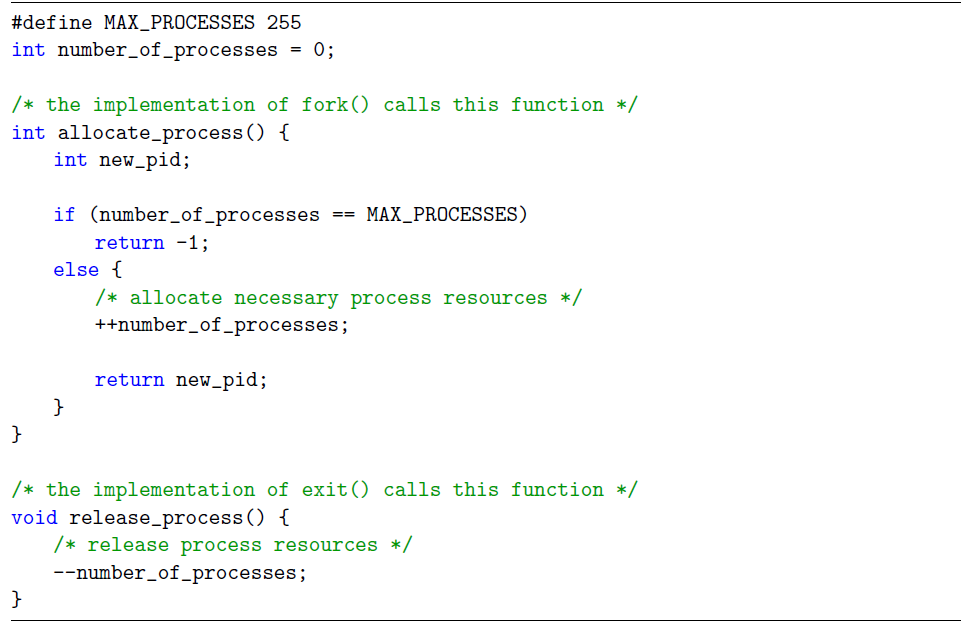
**Csc 139**

**Sect 7**

**CSC HW #2**

**Exercise 1. (OSC 6.22)** (15%) Consider the code example for allocating and releasing pro-

cesses shown below:



1. Identify the race condition(s).

++/-- number\_of\_processes

There is a racist condition on the variable number of processes.

2. Assume you have a mutex lock named mutex with the operations acquire() and release(). Indicate where the locking needs to be placed to prevent the race condition(s).

The mutex acquire() has to be before the critical sections of both functions. The code is shown below.

#define MAX\_PROCESSES 255 int number\_of\_processes = 0;

/\* the implementation of fork() calls this function \*/

int allocate\_process() {

int new\_pid;

**acquire(mutex);**

if (number\_of\_processes == MAX\_PROCESSES)   
 **release(mutex);**

return -1;

else {

/\* allocate necessary process resources \*/

++number\_of\_processes;  
 **release(mutex);**

return new\_pid;

}

}

/\* the implementation of exit() calls this function \*/

void release\_process() {

/\* release process resources \*/

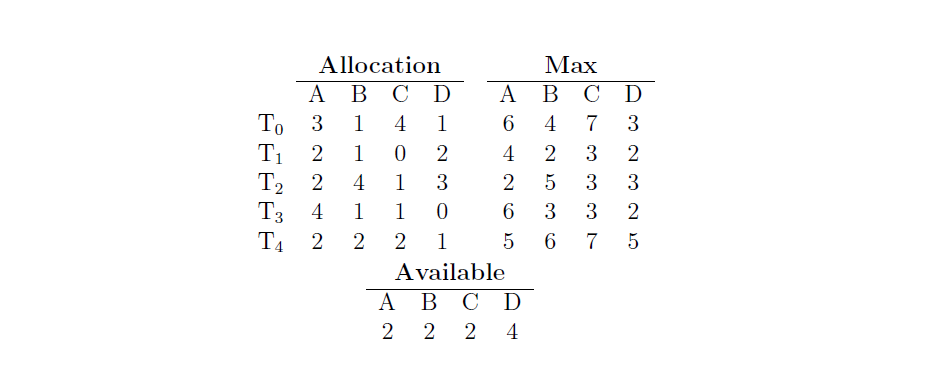
**acquire(mutex);**

--number\_of\_processes;  
 **release(mutex);**

}

**Exercise 2. (OSC 8.28)** (20%) Consider the following snapshot of a system:

Answer the following questions using the banker's algorithm:



1. Illustrate that the system is in a safe state by demonstrating an order in which the

threads may complete.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| T(0) | 3 | 3 | 3 | 3 |
| T(1) | 2 | 1 | 3 | 0 |
| T(2) | 0 | 1 | 2 | 0 |
| T(3) | 2 | 2 | 2 | 2 |
| T(4) | 3 | 4 | 5 | 4 |

2224+0120=2344

2344+2222=4566

4566+3454=8020

Sequence: T(2), T(3), T(4), T(0), T(1)

2. If a request from thread T4 arrives for (2,2,2,4), can the request be granted immediately?

Since 2224 is available the request can be granted immediately

3. If a request from thread T2 arrives for (0,1,1,0), can the request be granted immediately?

Yes, it can be granted immediately. Since 2224 is command requestr.

4. If a request from thread T3 arrives for (2,2,1,2), can the request be granted immediately?

Yes, can be granted since it is less than the available.

**Exercise 3. (OSC 8.22)** (5%) Consider a system consisting of a foursome of the same

type that are shared by threesome processes, each of which needs at most two resources. Is this system deadlock-free? Why or why not?

The system is deadlock-free, because all the process will access one resource and will request another one. Since there is a fourth resource, which will fulfill the requirement of the processes won’t be any cycles, so the system is in safe state or deadlock-free.

**Exercise 4**. (5%) Can a system be in a state that is neither deadlocked nor safe? If yes,

give an example system.

There are states that are neither deadlocked nor safe but could lead to a deadlock state.\\

Ex)

Max

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| T(0) | 2 | 0 | 0 | 0 |
| T(1) | 1 | 0 | 0 | 0 |
| T(2) | 0 | 1 | 2 | 1 |

Need

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| T(0) | 1 | 0 | 2 | 0 |
| T(1) | 0 | 1 | 3 | 1 |
| T(2) | 1 | 0 | 1 | 0 |

The system is not deadlock but if all the thread request consume remaining resources the syswill be in deadlock.

**Exercise 5. (OSC 5.23)** (5%) Consider a system implementing multilevel queue scheduling.

What strategy can a computer user employ to maximize the amount of CPU time allocated

to the user’s process?

The program could maximize the CPU time it by not fully utilizing its time quantums . The computer can increase the priority of the processes.

**Please complete the following survey questions:**

1. How much time did you spend on this homework 1800000 ms= 2 HOURS

2. Rate the overall dickulty of this homework on a scale of 1 to 5 with 5 being the most

Difficult. 2.68

3. Provide your comments on this homework (e.g., amount of work, difficulty, relevance

to the lectures, form of questions, etc.)