MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG334 Introduction to Operating Systems – Assignment 1

Date handed out: 29 March 2024, 02:00 PM
Date submission due: 15 April 2024, 11:00 PM

Important: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

1. **In the report to be submitted**: please include your name and student IDs on the first page.
2. **You code files,** MUST include name and student IDs.
3. We recommend typesetting your submission in word or any similar tool as you must submit the PDF version of it. Images of hand written solutions, unreadable report due to unclarity or English language typos will not be accepted.
4. As part of the typesetting requirement, all (state) graphs (if any needed) must be computer-generated (no hand-drawn or stylus-drawn graphs will be accepted). We recommend using Powerpoint/Google Slides or any other tool you prefer to draw any graphs.
5. Your code must be well commented, if your code is unreadable, unclear you may lose marks.
6. This is an individual assignment and not to be done in a group. All the material included in this assignment could be included in future quizzes, midterms and exams.
7. Cheating will be punished according to the rules mentioned in the syllabus.

Task 1 [10 Points]: Processes Synchronization

Q1) (10 Points) Many universities use commercial software packages. These kind of software provides limited number of licenses (indicated the maximum number of concurrently running applications). When the application is started, the license count is decremented. When the application is terminated, the license count is incremented. If all licenses are in use, requests to start the application are denied. Such requests will only be granted when an existing license holder terminates the application and a license is returned. Assume you have the following program segment to manage the available licenses (or any other resources):

```
#define MAX RESOURCES 5
int available resources = MAX RESOURCES;
```

When a process wishes to obtain a number of resources, it invokes the decrease_count() function:

```
/* decrease available_resources by count resources */
/* return 0 if sufficient resources available, */
/* otherwise return -1 */
int decrease_count(int count) {
   if (available_resources < count)
      return -1;
   else {
      available_resources -= count;

      return 0;
   }
}
```

When a process wants to return a number of resources, it calls the decrease count() function:

```
/* increase available_resources by count */
int increase_count(int count) {
   available_resources += count;

   return 0;
}
```

The preceding program segment produces a race condition. Do the following:

a. Identify the data involved in the race condition **(3 Points)**.

b. Identify the location (or locations) in the code where the race condition occurs **(3 Points)**.

c. **Explain** how you can use semaphore to fix the race condition and **provide** the pseudocode for it **(4 points, 2 points for each).**

Task 2 [90 Point]: Processes and Threads Creation

A) (10 points, 5 for each) mystery Code:

1) Check the code in mystery.c, explain why child and parent process print different values?

2) Convert the code you find in the file mystery.c to a code that uses threads and explain what was the difference between your new code and the one in the file mystery.c.

B) (5 points) CNG334_Task4 Code:

1. There are various ways to calculate the value of PI, one of the them is explained in the link below:

   Reference: https://towardsdatascience.com/estimate-pi-using-random-numbers-8b13a7e8c791

2. The code to do the calculation is already implemented for you, find it attached to the assignment. The file name is CNG334_Task4. For this task you are required to:

   * Create a constant the define the total number of threads you are planning to use (1 Point).

   * Add a global variable named circle_count and initialize it to zero (1 Point).

Protect your global variable using mutex from pthread. You can use the following reference to learn how to use Pthread mutex directly (1 Point).

Reference: https://www.ibm.com/docs/en/i/7.3?topic=ssw_ibm_i_73/apis/users_61.html

3. Explain clearly how your code is working before and after the mutex. To get the full mark you have to explain why the mutex is necessary and what each function you used is doing (2 Point).

NOTE: Initialize your mutex to null.

C) (85 Points) Write a POSIX C program that includes:

In this exercise you will be requested to prepare a small simulation using processes and threads.

C.1) Processes:

1. function **initializeProcesses()**, the propose of this function is to prepare an array based queue with 6 processes (this will be your ready queue), and to assign a random execution time per each, and priority value. Assume that processes id goes as follow: 1, 2, 3,..6 and the execution time is in the range of [10 and 30] millisecond and priority [1 and 10] (assume all the processes arrived at time 0) **(15 points – 5 data structure that represent the process, 5 points for creating the array correctly, 5 points for initialization).**

2. function **sortProcesses()** this function takes and sorts the ready queue (your array) based on execution time (shortest to longest) or priority (the lowest value is the highest priority) then print the sorted processes. Use selection sort algorithm for priority sorting and insertion sort for execution time based sorting **(10 points – 5 for each)**

3. **executeProcesses()** takes the ready queue and executes one of the scheduling technique (1- non-preemptive priority, 2- shortest process next) (20 Points – 10 for each algorithm). The function must print which process is in execution until the execution is done.

   NOTE: The previous functions should be done by one process, and the execution is done one after the other. For the functions: sortProcesses() and executeProcesses() you can add a parameter to specify your simulation sittings. You are free to decide how the simulation parameters will be taken.

4. **[10 points]** Now, create a child process and let it execute **sortProcesses()**.

5. **[10 points]** The parent process should wait for the child process to complete, then it should call **executeProcesses()**. Provide a print screen for the output. **Use wait().**

   NOTE: The sorted array should be visible for the parent (modification by the child should be visible for the parent process). Hint: you can revise mystery code provided to you.

C.2) Threads:

6. **[10 points]** Create two threads that executes the previous two functions (sortProcesses() and executeProcesses()). What is the different between using threads and using processes?

For this assignment, you have to submit:

A) A PDF file that include:

1. Answers to the proposed questions.

B) The source code, well commented.