Carleton University
Department of Systems and Computer Engineering
SYSC 4001 Operating Systems Fall 2016

Assignment 1

POSTED: Thursday, September 22nd 2016.

DUE DATE: Friday, October 7th (10 PM). No late assignments are accepted.

Submit it using the electronic submission on CULearn found in your SCE account on the undergraduate network. The submission process will be canceled at the deadline. No assignments will be accepted via email.

Part I – Concepts. Answer the following questions

a) [6 marks] Batch Operating Systems used special cards to automate processing and to identify the jobs to be done. A new job started by using a special card that contained a command like:

**JOB

and ended using another special card that contained the command

**END

After the **JOB card, there are different command cards to request specific services. For instance, the **DATA card would indicate where the data for the program running startrs.

a. [4 marks] Explain why special these cards are needed, what is their function and how do they interact with the Operating System.

b. [4 marks] Explain what would happen if in the middle of the program execution, the program crashes. What should the Operating System do in that case?

b) [6 marks] Explain briefly how does the dual-mode of protection works, and why is this hardware mechanism necessary.

c) [6 marks] What is the INPUT SPOOLER in an Operating System? Explain briefly how it works. What was the relationship between the INPUT SPOOLER and the Job Control Language interpreter? (the OS “Command Interpreter”)

d) [8 marks] Assume that you need to execute two programs in a Batch OS, in which the execution trace of each program is as follows:

a. Executes 1 s, card reading 2 s, Executes 1 s, printing 2 s.
b. Executes 1 s, card reading 1 s, Executes 1 s, printing 1 s.

Discuss what happens with these 2 programs and their execution times in the cases of:

i) Off-line execution
ii) Buffering
iii) Spooling (no multitasking)

iv) Spooling (multitasking)

Compare their turnaround time (i.e., the total time since ONE person arrives with ONE job to be executed until the results of that job are completely ready) the throughput, and the total execution time for both programs. Discuss your ideas (you can draw a timeline if you want; if you need information about anything missing to draw such timeline, make an assumption, write it on the paper, and answer the question using such missing information).

e) [4 marks] Define the essential properties of the following types of OS:
. Interactive
. Real Time
. Distributed
. Parallel

f) [5 marks] Explain briefly what is the basic concept of a Time-Sharing Operating System

Part II – Concurrency in Operating Systems [50 marks]

Design and implementation of a Classic Concurrency problem: The Sleeping Barber Shop system.

> *Objectives: this assignment is devoted to understand concurrency using Unix/Linux shared memory, semaphores, processes and threads.*

Problem description (from http://en.wikipedia.org/wiki/Sleeping_barber_problem):

The problem is analogous to that of keeping a barber working when there are customers, resting when there are none and doing so in an orderly manner. The analogy is based upon a hypothetical barber shop with one barber. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it.

Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves.

Based on a naïve analysis, the above description should ensure that the shop functions correctly, with the barber cutting the hair of anyone who arrives until there are no more customers, and then sleeping until the next customer arrives.

Part A)

Write a solution to the problem written in C/C++ running under Linux.

Your solution must have 3 processes running concurrently (one for the barber room, one to generate customers, and one for the waiting room). Each of the processes should inform what they are doing on screen. Name your program "partA.cpp" and your executable "partA". At this point do not worry for concurrency issues.

HINT: work incrementally. Start with each process individually and test them. You can also use extra processes for "other activities" (i.e., starting or ending the processes).

Part B)

Run the program. If (all or some of) the processes seem to be in deadlock. Open the files and check the current status. Report the conditions that lead to the deadlock. If there is no deadlock on any of your runs, briefly discuss the execution order for the processes. Write your answers in a file named "partB.txt"

Part C)

Convert Part A) into a Semaphore-based solution with shared memory. Use semaphores for process coordination. Other Linux system calls may be used for process creation, termination, creation and deletion of shared memory etc.

Your solution must have 3 processes running concurrently. Your solution should use a shared vector to store the status of the barber shop (representing customers in the waiting room or in the barber chair, and the barber's status). This version should deal with any concurrency issues through the use of semaphores. DO NOT WORRY ABOUT DEADLOCK/LIVELOCK. Each of the processes should inform what they are doing on screen. Name your program "partC.cpp" and your executable "partC".

Part D)

Run the program. If (all or some of) the processes seem to be in deadlock, report the condition that lead to the deadlock. If there is no deadlock on any of your runs, briefly discuss the execution order for the processes. Write your answers in a file named “partD.txt”

Hand In
You must use the Submit program and submit:
- your programs
- A README file describing how to compile and run your program with the various test cases.
- A discussion of your design in the context of the three requirements associated with the solution to the critical section problem.

Marking Scheme:

TOTAL: 80 marks (corresponds to 10% of the total for the course)

For Part II:

Correctness (including error checking and final cleaning up): 65%
Documentation and output: 25% (You need to print **CLEARLY**, especially before and after each reading or writing to the shared data area)
Program structure: 5%
Style and readability: 5%