Carleton University
Department of Systems and Computer Engineering
SYSC 4001 Operating Systems Fall 2020

Assignment 1

POSTED: Thursday, January 14th, 2020.

DUE DATE: Friday, February 5th (10 PM). No late assignments are accepted.

THIS ASSIGNMENT MUST BE DONE BY TEAMS OF TWO STUDENTS. Only ONE student must submit the assignment, clearly stating who the members of the team are. The programming part of the Assignment should be done using the Pair Programming technique.

Please submit the assignment using the electronic submission on CULearn found in your SCE account on the undergraduate network. The submission process will be closed at the deadline. No assignments will be accepted via email.

Part I – Concepts [45 marks]. Answer the following questions (from Chapter 1, 2 Silberschatz + lectures, marks in brackets)

a) [3] What are the three main purposes of an Operating System?

b) [3] Explain briefly how the dual-mode of protection works and why this hardware mechanism is necessary.

c) [4] Which of the following instructions should be privileged? Why? (1-line justification)
   i. Set value of timer
   ii. Read the clock
   iii. Set a memory position in user space to zero (*i.e.*, clear memory in the User's space)
   iv. Issue a trap instruction (software interrupt instruction)
   v. Turn off interrupts
   vi. Switch from user to kernel mode
   vii. Multiply two numbers
   viii. Access I/O device directly

d) [2] In a multiprogramming and time-sharing system several users share the system simultaneously resulting in security problems:
   i. Discuss these problems
   ii. Can we ensure the same degree of security in a time-shared machine as in a dedicated machine? Explain.

e) [4] What is the purpose of interrupts? What are the differences between a trap and an interrupt?

f) [6] Batch Operating Systems use special cards to automate processing and to identify the jobs to be completed. A new job, consisting of a batch of cards, needed to include a special card at the top of the pile, containing a command with the form:

```
**JOB
```

and at the end of the pile, the job is ended using another special card that contains 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 running program begins.

> i. [3] Explain why these special cards are needed, what their function is, and how they interact with the Operating System.
>
> ii. [3] Explain what would happen if the program crashes in the middle of the program execution. What should the Operating System do in that case?

g) [4] What is the INPUT SPOOLER in an Operating System? Explain briefly how it works. What is the relationship between the INPUT SPOOLER and the Job Control Language interpreter? (the OS “Command Interpreter”)

h) [4] Define the essential properties of the following types of OS:

- i. Interactive
- ii. Real Time
- iii. Distributed
- iv. Parallel

i) [2] Explain briefly how a Time-Sharing Operating System works. How does it manage to handle multiple interactive users with a single CPU?

j) [5] (Kernel structure) Let us suppose we have a process in a multitasking OS. The process is currently running (*i.e.*, the CPU has been allocated to that process and the process is in the *running* state).

What are the possible events that can make that process **abandon** the use of the CPU? Explain how the OS Kernel will react to these different events (in terms of the states of the system and the transitions between these states, and the routines of the kernel involved in the process).

Hint: there are at least *three* kinds of events to be considered, which will produce three completely different state changes and their corresponding transitions. Explain how the OS Kernel will react to this event in detail.

k) [3] DMA is used for high-speed I/O devices to reduce overhead:

- i. How does the CPU interface with the device to coordinate the transfer?
- ii. How does the CPU know when the DMA operations are complete?
- iii. The CPU can execute other programs while DMA is transferring data. Does this process interfere with the execution of the user programs? If so, explain how.

l) [2.5] What are the five major activities of an OS regarding process management?

l) [2.5] What are the five major activities of an OS regarding memory management?

Part II – [55 marks] Design and Implementation of a Kernel Simulator

The objective of this assignment is to build a small simulator of an OS kernel, which could be used for performance analysis of different scheduling algorithms. This simulator will also be used in Assignment 2.

i) Input data

The simulator will receive, as an input, a list of processes to run with their trace information, as follows:

Pid	Arrival Time	Total CPU Time	I/O Frequency	I/O Duration

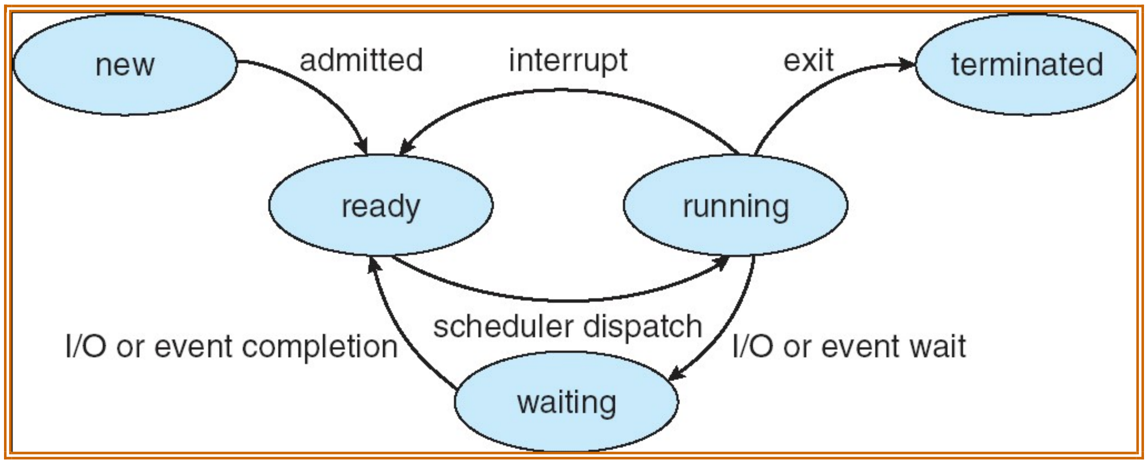
- Pid: a unique identifier for the process
- Arrival Time: the initial simulated time is 0 and the arrival time can be at 0 or any time value thereafter. The time units to be used are milliseconds.
- Total CPU Time: it is the total time the process needs to complete (does not include I/O time: only the total time needed in the CPU)
- I/O Frequency: the processes are assumed to make an input/output with this frequency
- I/O duration: this is the duration for the I/O for each of the processes (assumed to be the same for all the I/O operations)

The information will be stored in a file (to be submitted).

ii) Simulation implementation

Using the information on the input file, you must define a data structure in memory (array of structs, linked list) like a PCB (only include the information needed: PID, CPU, I/O information, and remaining CPU time – needed to represent preemption –).

The simulation should try to reproduce the behavior of this state diagram (from Silberschatz *et al.*):



Note: The New and Terminated states are optional (you can simply load the process information in the Ready queue when they start.

ASSUME THAT THE TRANSITIONS TAKE ZERO TIME (that is, do not worry for the time taken by the ISRs, scheduler, context switch, *etc.*). Every time the simulation changes states, the following information should be included in an output file.

ASSUME THAT YOU ALWAYS HAVE AN I/O DEVICE AVAILABLE (that is, do not worry about waiting queues at the I/O devices: whenever you request an I/O, the I/O starts immediately).

Time of transition	Pid	Old State	New State

- Time: The simulation time starts at 0, and it is measured in milliseconds.
- Pid: the id for the process that has changed state
- Old State/New State: The state of the process before and after the transition

The scheduler should get the first element of the list; do not worry about scheduling.

We will submit test cases in CULearn; the TAs will mark these cases first. You can include your own test cases too.

The program must be written in C. You must submit an executable, the source code, all files needed to compile, test scenarios, and scripts to run them. Include, at least, two scripts named "test1.bat" and "test2.bat" that will be used to run 2 different tests automatically. The TAs will use these two basic tests, and then will modify your input files,

Marking Scheme:

TOTAL: 100 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%