COMP 346: Operating Systems - Winter 2020

Instructors: Dr. Aiman Hanna, Dr. Hakim Mellah, Prof. Kerly Titus

Department of Computer Science and Software Engineering
Concordia University

24-hour Take-home Final Examination

Submission Due date and time (EXACT):
Wednesday, April 22nd 9:00 AM EST

Exam Rules: Please read very carefully:

1. General Information

Date and time of exam release: **Tuesday, April 21, 2020 at 9:00 AM EST**

Deadline for exam submission: **Wednesday, April 22, 2020 at 9:00 AM EST**

Exam is a 24-hours exam. You must submit your solution by 9:00 AM on Wednesday April 22. Submission received at, or after 9:01 AM EST, will be rejected and a FNS grade will automatically be issued for the course. [In fact, the system will reject any submissions beyond 9:00 AM EST]. After the exam, you *may* be called for an oral evaluation. If you are called for an online oral evaluation, you **MUST** present yourself for this evaluation. Failing to do that will result in your grade being suspended and eventually an FNS grade is assigned.

Quality of Submission: You will be marked based on correctness, comprehension, completeness and quality of produced work.

Submissions: You need to submit your solution in a file named <Your-ID#_Your-Name>.pdf (or .doc. These are the only two allowed formats). For instance: 9080332_Mike_Simon.pdf. The file must include you name, ID, and you must **sign** it. **The solution MUST follow the order of the questions (i.e. Question 1 solution, then Question 2, etc.).** You must submit that file to Moodle of your section under the titled: ***Final Exam Submission*.**

⇨ **VERY IMPORTANT: You will also need to sign this exam document itself and include it with your submission as well. Your exam will not be marked unless this exam is signed and returned back.**

If we do not submit these two signatures; we will have to ask you for it manually before your exam can be considered. This request will cost you 10% of the mark (no exceptions will ever be considered)! So; in simple words, return the exam signed, and return the answer document signed.

2. Terms and Conditions

The exam is an individual exam. The Exam is open-book. You can consult any materials, including textbooks, notes, Moodle materials, tutorial notes, and online materials. However; you are **NOT** permitted to communicate with any persons neither offline nor online (or through any other type of communication), including your classmates.

In simple words, **plagiarism will NOT be tolerated under any conditions** and it will officially be reported to the University for disciplinary actions.

3. Questions that arise during the exam

It is important to note that there will be no support from the instructors or the TAs during the exam. The exam contents are clear; however, in case you find anything ambiguous, you must express your assumptions very clearly and continue with the solution. We may however send you some announcements if needed through Moodle and the mailing list, so you should check from time to time. In simple words, Avoid hesitations! The text of the exam is very clear and the requirements are detailed carefully. You should not send requests to your professors requesting clarifications. They may not even have a chance to see these requests on-time.

4. Academic Integrity

Concordia University takes academic integrity very seriously and expects its students to do the same.
You need to declare and sign the following:
I understand that any form of cheating, or plagiarism, as well as any other form of dishonest behaviour, intentional or not, related to the obtention of gain, academic or otherwise, or the interference in evaluative exercises committed by a student is an offence under the Academic Code of Conduct.
I understand that the above actions constitute an offence by anyone who carries them out, attempts to carry them out or participates in them.
By way of example only, academic offences include:

- Plagiarism;
- Contribution by a student (or anyone else) to another student's work with the knowledge that such work may be submitted by the other student as their own;
- Unauthorized collaboration;
- Obtaining the questions or answers to an exam or other unauthorized resource;
- Use of another person's exam during an exam;
- Communication with anyone (in any shape or form) during an exam and any unauthorized assistance during an exam;
- Impersonation;
- Falsification of a document, a fact, data, or reference.

For more information about academic misconduct and academic integrity, refer to the Academic Code of Conduct and Concordia University's Academic Integrity webpage.

I have neither given nor received unauthorized aid on this exam and I agree to adhere to the specific Terms and Conditions that govern this exam.

Exam is ©Concordia University. I will NOT publish this exam (or any extracts or texts of it) in any form or shape online or elsewhere, or pass it to anyone in any form or shape, neither now nor at any future point. I understand that publications of this exam requires prior written permission from ALL instructors of the course and that this permission is NOT given at this time.

Student Name **Signature:**

ID #: ---------- **----------**

Question 1: [10 Points]:

Assume a system that is capable of supporting multiprogramming, where the following measures are defined:

- ***Turnaround Time***: Total time to complete a process;
- ***Throughput:*** Average number of processes completed per time period *T,* which is defined as **12ms** (milliseconds).
- ***CPU Utilization:*** % of time where CPU is being utilized (not idle) before *N* processes are completed.
- ***Wait Time:*** Total amount of time a ready process (a *ready* process is a process that requires CPU execution) is waiting (for the CPU) from the moment it is ready until it terminates.

Assume that all processes in the system have similar behavior, where a process spends ½ of its time performing I/O and ½ of the time performing execution. The system is capable of overlapping execution time for a process with I/O time of another. The system is utilizing also RR for scheduling with a time quantum of **3ms** (ignore context switch time; i.e. assume it takes 0 ms). Show the Gantt Chart for all your answers.

A) Assume that each process has a total time of 48ms, and for each period of 12ms, each process does/perform an I/O for the first ½ of a period (6ms), then follows with CPU operations (execution) during the 2nd ½ of a period (6ms). All processes arrive at time t=0.

i) Which number of running process (1 process, 2 processes, or 4 processes) produces best CPU utilization? Show all your work.

ii) What is the throughput if 2 processes are running?

iii) What is the average wait time if 4 processes are running? Give the wait time for P_1, P_2, P_3 and P_4.

iv) What is the average turnaround time for 4 processes? Give the turnaround time for P_1, P_2, P_3 and P_4.

B) Now assume that four processes, P_1, P_2, P_3 and P_4, will run, and all of them arrive at time t=0. Each of these processes has a total time of 24ms, with the exception of P4, which has 60ms. Assume also that for each of these processes, the first and last quarter of a period (that is the first and last 3ms) are I/O, and that the second and third quarters are executions.

i) What is the system throughput?

v) What is the average wait time for the processes? Give the wait time for P_1, P_2, P_3 and P_4.

ii) What is the CPU utilization (during the time these processes are running until they terminate)?

iii) What is the average turnaround time for the processes? Give the turnaround time for P_1, P_2, P_3 and P_4.

Question 2: [8 Points]:

Given the following code, where ***id*** indicates a process number (or ID), ***launch***() indicates concurrent start of the processes passed on its parameters, and ***N*** is an integer representing the number of processes that will be launched (i.e. processes that will run on that system).

```
boolean blocked[N];

for(int i = 0; i < N; i++)
    blocked[i] = false.

int turn = 0;
int other;

P(int id)

{
        while(true)

        {
             blocked[id] = true;
             while(turn != id)
             {
                if(id == 0 || id == 2)
                        other = 1;
                else
                        other = 0;

                while(blocked[other]) { turn = other;}
                turn = id;

              }

             /*  <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */

            blocked[id] = false;

        }

}
```

i) Will the above code work (achieves mutual execution, good progress and deadlock-free) if ***launch*** is called as: **launch(P(0), P(1))**. Explain your answer in full details and in terms of each of mutual exclusion, good progress and deadlock-free.

ii) Will the above code work (achieves mutual execution, good progress and deadlock-free) if ***launch*** is called as: **launch(P(0), P(1), P(2))**. Explain your answer in full details and in terms of each of mutual exclusion, good progress and deadlock-free.

Question 3 [8 Points]:

As a part of a development team in an Operating Systems company, your team was given the sole responsibility of looking at solutions to the Deadlock problem. After a large research in the subject, here is what your team (after being divided into smaller sub-teams) came up with as possible solutions. Team 1, Team 2 and Team 3 proposed i, ii, and iii, respectively.

i) Roll-Back and Random:

The main idea here is that if deadlock occurs, detect it (which is already possible), then roll/revert back each process to the exact moment before deadlock started. At that point, randomly change the way resources were allocated (compared to the previous time), then resume the processes again. The idea is based on other probabilistic studies, which proved that such random allocation will likely change the order of allocation next time and hence avoid the same deadlock situation from occurring.

ii) Release-All and Restart Differently:

The main idea here is that if deadlock occurs, detect it (which is already possible), then terminate all processes involved in the deadlock, hence releasing all allocated recourses, then restart the processes again but that time in different order. The idea is based on other probabilistic studies, which proved that such random restarting of the processes will likely change the allocation next time and hence avoid the same deadlock situation from occurring.

iii) Terminate and Revert to Limited Uni-programming

The main idea here is that if deadlock occurs, detect it (which is already possible), then terminate all processes involved in the deadlock, then restart them randomly but in a limited uni-programming fashion. The idea does not change the system to a uni-programming system; rather it only schedule these involved processes in a uni-programming fashion. In other words, the system will still run multiple processes, but none of these processes involved in the deadlock will run at the same time; rather one after another. Hence the term "Limited" uni-programming instead of just uni-programming.

A) Which team would you support, or wish to be a part of?
B) Indicate clearly the reasons why you would support such solution.
C) Indicate clearly the reasons why you would not support the proposals that you are rejecting.

You should be as detailed as possible. However, your answer should not exceed ½ a page. In other words, just indicate what is important and needed, instead of having lots of unnecessary explanations.

Question 4 [10 Points]:

You have just been hired to develop a program that searches in a huge file having 10 million elements. This file stores the social insurance numbers of the customers of an investment company. You have discussed multiple strategies in order to obtain the best average search time. The best option is to sort the elements so that binary search can be used for an average search time of $O(log_2 n)$, which would result in a maximum of 24 look-ups. Using sequential search would result in an average search time of 5,000,000 look-ups, 1 look-up for the best case and 10 million look-ups for the worst case. However, keeping the elements sorted would be very time-consuming as new elements are added or removed frequently (that by itself would cost a complexity of *n* as shifting may be needed!). Consequently, the team leader decided to just go with a sequential search. You accepted his decision but proposed him to use multiple-threads to speed-up the processing.

The computing system has a single CPU, and is using a Round-Robin scheduling algorithm with a time quantum of 20 *ms* (milliseconds) and has a context switch time of 1 *ms*. Also, each search operation (that is, each array loop, or each single iteration over the array) requires 5 *ns* (nanoseconds). The search application has other operations beside the loop but their execution time is not significant and is not taken into consideration. For each of the questions below, your answer should not exceed ½ a page (each).

1. Explain whether using multiple threads would really provide an advantage over using just a single thread. You need to explain and justify your answer for the *most optimal* solution (best case), as well as for *worst case* and the *average case*.

2. Whether or not the use of multi-threading is more advantageous than single-threading, discuss another type of application (give a completely distinct real-life problem that is different from the search problem described in this question) that could benefit more by using multi-threading in a single CPU environment.

3. Consider a load of 2 threads handling the search operation such that each thread searches one half of the array. Assume that both threads are on the ready queue at time t=0 and that the search value is not found in the list of elements.
    a. Draw the Gantt chart.
    b. Calculate the average turnaround time and provide the turnaround time for each thread.

4. Suppose we use a load of 4 threads instead of 2, and that all of them are on the ready queue at time t=0. Would there be any change in the average turnaround time as little as it can be? Explain – show all your numbers/calculations.

5. Now assume 4 CPUs and a load of 4 threads, and that all threads are on the ready queue at time t=0. Will that improve the performance of the search algorithm as opposed to using a single CPU? Explain clearly why or why not?

Question 5 [10 Points]:

Now, we continue from Question 4 above. Before making his final decision on whether to use multi-threading or single-threading, the team leader asked you to show him a demonstration about how the multiple threads will be synchronized (so we completely ignore anything related to Round-Robin scheduling or time quantum here). You decided to implement 4 threads and divide the array into 4 sections. Thread T_1 searches the first quarter (index 0 to 2,249,999), thread T_2 searches the second quarter (index 2,500,000 to 4,999,999), thread T_3 searches the third quarter (index 5,000,000 to 7,499,999) and thread T_4 searches the last quarter (index 7,500,000 to 9,999,999. Thus, every thread knows its quarter and runs independently from the other threads.

When a thread $T_1 \ldots T_4$ has finished searching its quarter, successfully or unsuccessfully, it sets its state to finish using the boolean array finishRequest[i], where i = 1,...,4, and waits until the application, through its main thread T_0, sends it a signal to terminate. However, if the search is successful, then this tread must inform all other threads immediately, and so the other threads consequently just terminate their searches, set their state to *finish*, using the boolean array finishRequest[i], and wait for a signal from the main thread, T_0, to terminate. Once all 4 threads have terminated, the main thread, T_0, can finally terminate.

- Using **Semaphores** or **Monitors,** Write the synchronization code (You are **NOT** required to (and should **not**) write actual Java code for the solution; rather you need to write the solution using semaphore/monitor pseudo-code notation, and structures such as if, while, … if necessary). In particular, you need to write the code for thread T_0 and for threads T_1, T_2, T_3 and T_4. Use the following data structures and layouts for T_0 and $T_1 \ldots T_4$ as given below.
  ➔ It is better that you write the code of $T_1 \ldots T_4$ using one single method that takes the thread ID as a parameter. However, if you wish, you are allowed to give the code of these 4 threads separately as well (i.e., in that case you will give the code of 5 different methods for each of the 5 threads).

```
// Shared data structures; you need to add other constructs as needed

    // Initialize the state of all threads to not finished.
    // A thread T1…T4 sets finishRequest[i] to true when it needs to terminate.
    boolean finishRequest[n] = {false}; // All initialized to false here

T(0)

{

    … // This is what you need to fill for T0 code

}

T(int tid ) // tid is 1…4

{

    …     // This is what you need to fill for T1 to T4 code
   …     // Again, if you wish, you can write 4 separate methods; one for each of the 4 threads

}
```

Question 6 [4 Points]:

Linked lists allocation can be used to keep track of allocated blocks of a file system, as well as to keep track of unallocated blocks. Assume that the probability of pointer corruption in the system is assumed to be very small, and hence negligible.

For each of these two allocation schemes, and considering the given constraints/conditions of the system, indicate if it is possible to safely and efficiently use linked-list allocation for these two types of allocation? your answer should not exceed ½ a page.

Question 7 [5 Points]:

You and your classmates have looked carefully at the SSTF disk scheduling then went through an argument, where some indicated that the scheme will have very negative side effects and so it must not be considered, while the others indicated that it will produce superior performance compared to the other available schemes and hence it should be used.

i) Which side are you with? Explain why? your answer should not exceed ¼ a page.

ii) Understanding that technique, explain clearly what are the details/reasons behind both support and rejection by the classmates. your answer should not exceed ½ a page.

iii) Assume that priority is needed, where the requests should be served on a local area on the disk (that is, particular tracks at a local part of the disk are more important than others and hence they need to execute first). Based on these new constraints, would your answer from (i) above change? Explain clearly why or why not. your answer should not exceed ½ a page.

Question 8 [5 Points]:

a. Concerning page replacement; is it possible to implement *Belady Optimal* algorithm, or even an approximation of it? If yes, explain how this can be done; if no, explain why it is not feasible to implement this algorithm. your answer should not exceed ½ a page.

b. It is indeed possible to implement LRU algorithm for page replacement. Outline a procedure/algorithm for implementing this algorithm **efficiently**. If you believe your algorithm is efficient, clearly explain why you think so (give strong supportive arguments). If your algorithm is not efficient, explain why it is not (keep in mind that the question is requesting you to implement an efficient algorithm.) your answer should not exceed ¾ of a page.

c. Assume that the size of an executable file of a process is evenly divided by the page size (for example, size is 700 and page size is 100, so you have exactly 7 pages). Assume also that we are not interested in external fragmentation and that LFU algorithm, where 3 frames are allocated for the process, is used. Under these conditions, is there anyway that the system suffers from internal fragmentation? Explain your answer very clearly. your answer should not exceed ½ a page.

Question 9 [5 Points]:

Consider a demand-paged system that has a fast associative memory (TLB, or cache). To speed up access to pages, it is desired to place the page table in the cache. However, since the page table of a process is too large, the system included only the *first* 20% of the page table in the cache, 30% in the main memory, and the rest of the table had to be kept in the secondary memory (i.e. page faults may be needed for the page table itself!). Consider uniform request for each of the pages (i.e. each page is requested more or less the same amount of times). To simplify the calculations, assume that cache reference costs 30 ms (milliseconds), memory access costs 200 ms, and page fault time costs 900 ms if the fault is for the page table, and 2 seconds if the fault is for a page. The page fault time for a page includes exactly the following: obtaining the page from the secondary memory, possible swapping, and updating the page table.

i) What is the average effective access time for a page, if that page is in the main memory? Explain clearly; you must show all your calculations; and provide the exact final answer.

ii) What is the average effective access time for a page, if that page is NOT in the main memory? Explain clearly; you must show all your calculations; and provide the exact final answer.

Question 10 [35 Points]:

Technology and Market Impact on Operating systems

Some years back, operating systems played a critical rule in changing worldwide technologies and the market, and consequently drove what trended at that time. While this may be true as well today, it is also true that technology trends are also changing and impacting operating systems in many different ways. In this question, you will need to use the concepts learned in class to go beyond what was covered. In other words, you will need to use what you have learned and **understood** in class to conduct a small research, which surely include ideas, subjects, technologies, market pressures, etc., which were not explicitly covered in class. In particular, your research needs to look/examine the last 5 years or so, as well as the next 5 years as you can anticipate from available predictions on the subjects. **Each** of the following parts (these are # i) and # ii) below) needs to be documented between 1 page and 2 pages, excluding figures, tables, drawings, etc. (Use New Times Roman as font, with size 12, and single line spacing. Keep the Layout Margin as Normal). You must respect these limits/constraints. Notice that you cannot just copy and paste from any sources. You must understand, analyze, validate, …. etc, before documenting your answer.

i) For each of the following subjects/matters, describe how technology trends in the **past 5 years** impacted this subject, in relation to operating systems, (or impacted operating systems in relation to it!). You should be as comprehensive as possible (comment on different relevant aspects, such as performance, cost, ……etc.)

- IoT (Internet of Things)
- CPU processing (speed, capacity, cores, etc.)
- Main Memory (i.e. RAM)
- Cloud Computing
- Big-data Processing

ii) For each of the following subjects/matters, describe how "**you anticipate**" that very recent and future technology trends in the short-term of the **next 5 years** will impact this subject, in relation to operating systems (or impact operating systems in relation to it). You should be as comprehensive as possible. Remember that your answer should reflect your **own** anticipation based on your research.

- 5G Networks
- Artificial intelligence
- Reacting to world pandemics, such as the current COVID-19 at the writing time of this exam!
- Fault-tolerance

// End of Exam; Good Luck!