COMP 346 – Operating Systems
Course Outline – Fall 2020

Lecture Times:

Instructor:

Dr. Hakim Mellah: hakim.mellah@concordia.ca

- Recorded lectures will be available on Moodle before each class.
- Live Session: Tuesday: 18:00 – 20:00 (Live on Zoom). This is not a lecture but Q&A session.
- Quiz (15 minutes): Tuesday: **19:00 – 19:15**.

Tutorial Times: You can attend any of these tutorial times:

- Tuesday 16:15-17:05 (Live on Zoom)
- Tuesday 20:30-21:20 (Live on Zoom)
- Thursday 16:15-17:05 (Live on Zoom)

Labs Times: You can attend any of these lab times:

- Thursday 17:45-19:35 (Live on Zoom)
- Thursday 19:45-21:30 (Live on Zoom)
- Friday 17:45-19:35 (Live on Zoom)

Tutors and Labs instructors:

Yanal Alahmad: yanal.tg@gmail.com
Mayank Vadariya mayankkumar.vadariya@mail.concordia.ca

Markers:

Niloofar Keypour keypour.niloofar@gmail.com
Yasaman Sarlati yasaman.sarlati@yahoo.com
Maryam Rasti maryrasti91@gmail.com
Bhupinder Kaur bhangu.bhupinder@gmail.com

Pre-Requisites: COMP 352; COMP 228 or SOEN 228.

Objectives

This course has two components: a *theory component* to teach you the concepts and principles that underlie modern operating systems, and a *practice component* to relate theoretical principles with operating system implementation. In the theory component, you will learn about processes and processor management, concurrency and synchronization, memory management schemes, file

system and secondary storage management, security and protection, etc. The practice component will complement the theory component through programming assignments illustrating the use and implementation of these concepts. At the end of the course you should:

- Understand fundamental operating system abstractions such as *processes*, *threads*, *files*, *semaphores*, *IPC abstractions*, *shared memory regions*, etc.,
- Understand how the operating system abstractions can be used in the development of application programs, or to build higher level abstractions,
- Understand how the operating system abstractions can be implemented,
- Understand the principles of concurrency and synchronization, and apply them to write correct concurrent programs/software,
- Understand basic resource management techniques (scheduling or time management, space management) and principles and how they can be implemented. These also include issues of performance and fairness objectives, avoiding deadlocks, as well as security and protection.

Textbook and Syllabus

The textbook for the course is: ***Operating System Concepts, 10th Edition***, by: **Silberschatz/Galvin/Gagne**, published by John Wiley & Sons, 2018. **ISBN 978-1-118-06333-0**. (6th edition ISBN **0-471-41743-2**, and later ones, are also okay).
The exact syllabus for the course will depend on how quickly we can cover the introductory chapters. In general, our aim for this semester is to cover chapters 1 through 14, and parts of the following chapters. Any exclusion will be announced in class and made available through the web page.

Other References:

Operating Systems, 3rd edition By: ***Gary Nutt***, published by *Addison-Wesley.* ISBN 0201773449 (ISBN-10: 0201773449 | ISBN-13: 9780201773446). *Second edition of this book (Operating Systems: A Modern Perspective) is also okay.*
Applied Operating System Concepts*, First Edition,* By: *Silberschatz/Galvin/Gagne* - published by *John Wiley & Sons, 2000.*

Background

The official pre-requisite for this course is Data Structures & Algorithms (COMP 352, and COMP 228 or SOEN 228). Much of the material covered in these two courses is directly relevant for this course, and therefore it is critical that you understand and remember the knowledge that you acquired in these courses.
Some of the critical concepts that you should know thoroughly to do well in this course include:

- The hardware structure of modern computers, the role of the different parts (CPU, main memory, secondary memory, system bus, peripheral devices), the inter-connections between the different parts, and how the different parts interact to support computation, communication, and storage.
- The binary number system, basic arithmetic and logical operations on binary numbers, and converting binary numbers to decimal and vice-versa.
- The hardware-software interface including assembly language programming, accessing registers and main memory from software, I/O instructions, interrupts and interrupt handling, etc.

- Common data structures such as queues, stacks, lists, prioritized lists, etc. Implementation of such data structures using arrays, dynamic arrays, and linked-lists. Evaluation of algorithmic complexity of the operations.

The important skills needed to do well in this course include: (1) good programming skills, including the ability to understand, design, implement, and debug programs with non-obvious flow of control, (2) the ability to understand trade-offs using a mix of qualitative and quantitative reasoning of design choices, (3) the ability to relate the theoretical material covered in class to the practical aspects of implementation, and (4) the ability to abstract the knowledge learnt and apply it to a wide-range of problems (not necessarily related to operating systems, or even computer science for that matter).

Workload and Grading

This is a reasonably heavy course with several new concepts, and a fair bit of programming workload. Therefore, you should be prepared to spend adequate time and effort on this course. The practice/programming component of the course will utilize Java programming language. Although you are not required to have a strong previous knowledge of Java, it is assumed that you have a reasonably solid knowledge with object-oriented programming. Sufficient introduction of Java will be provided during the early tutorials, so it is very important that you attend the tutorials.

1. **Mid-Term Exam:** 25%

   There will be one midterm exam (on COLE). The date for the exam will be announced at the beginning of the term. The midterm will cover all material presented in the lectures, the textbook, and in the assignments and labs, up to and including the lecture preceding the exam.

2. **Quizzes**: 20%

   There will be 10 quizzes, each is worth 2.5%. The quizzes mark will be selected from the best 8 quizzes (to give 20%). Quizzes are scheduled every Tuesday from 19h30 to 19h45. **Quizzes are timed online activities and there will be no replacement or substitution for missed quizzes under any circumstances**.

3. **Final Exam:** 40%

   The final exam will be scheduled by the University Exam's office (on COLE). The exam will cover material from the whole semester, including lectures, textbook, and assignments. Passing the final exam is necessary for passing the course.

4. **Assignments:** 15%

   There will be a total of 3 theoretical assignments, which must be done individually. The main purpose behind these assignments is to provide you with good preparation for the final**.**
   There will be 3 programming assignments. These assignments are designed to give you more understanding of the course material and to give you practices into aspects of operating system design and implementation. The assignments are using the Java language under Linux operating system. More information on the specific compiler version as well as where it is located in the system will be provided during lab times. Reasonable attempts of these assignments are necessary for passing the course.

IMPORTANT: A demo will be required for each of these assignments. The marker will communicate with you through the mailing list and you must book a demo time for your team. You **must** perform the demo (if working in a group, both members of a team must be present during the demo). Failing to demo the assignment or failing to attend the demo at the reserved time, will result in a 0 mark regardless of your submission. There will be no replacement for a missed demo time.

⇨ Please note that all assignments will be placed on the course website (Moodle). All assignments (theoretical and programming) must be submitted electronically.

Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must also contain your name(s) and student ID(s). Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. Only electronic submissions will be accepted. Students will have to submit their assignments (one copy per group for the programming assignments; theory assignments are individual assignments) using the Moodle system. Assignments must be submitted in the right folder of the assignments. **Assignments uploaded to an incorrect folder will not be marked and result in a zero mark. No resubmissions will be allowed.**

The grading of the course will be done based on the relative percentages assigned to the assignments, quizzes and the exams. For reasons of fairness, we may choose to scale up/down the marks in a particular exam or assignment to ensure that all aspects of the course receive a fair weight. Any such "fine-tuning" will be made known to you before the final grades are assessed. Finally, there are no pre-set cutoff points for the final grades; the cutoff points will be decided based on an assessment of difficulty level, class performance, fairness, and instructor's wisdom from teaching and grading the course in the past. That is, there is no definite rule for translation of number grades to letter grades.
In the event of extraordinary circumstances beyond the University's control, the content and/or evaluation scheme in this course is subject to change.

VERY IMPORTANT

- The instructor reserves the right to conduct an individual oral examination after each exam/quiz to verify the student's response to specific questions.
- Student must have access to internet and hardware equipment (computer, webcam, microphone).
- Student must be able to do online timed exams.

Graduate Attributes

This course emphasizes and develops the following CEAB graduate attributes:

• **Knowledge-base:** *Knowledge of fundamentals of operating system functionalities, design and implementation. Multiprogramming: processes and threads, context switching, queuing models and scheduling. Inter-process communication and synchronization. Principles of concurrency. Synchronization primitives. Deadlock detection and recovery, prevention and avoidance schemes. Memory management. Device management. File systems. Protection models and schemes.*

➢ **Indicators:** Indicator 1.3: Knowledge-base in a specific domain.

• **Problem analysis:** *Analyze multithreaded programs to check for deadlocks and race conditions. Design race-condition and deadlock-free solutions for concurrency.*

➢ **Indicators:** Indicator 2.1: Problem identification and formulation.

Indicator 2.2: Modeling.
• **Design:** *Design and implement programs that interact with operating systems kernels. Develop programs using concurrency principles.* ➢ **Indicators:** Indicator 4.3: Architectural and detailed design.

Indicator 4.4: Implementation and validation.
• **Use of Engineering tools:** *Use appropriate system kernel resources to develop system software. Use semaphores and Java's built-in synchronization primitives and add-on utilities (e.g., locks and condition variables) to implement concurrent programs.* ➢ **Indicators:** Indicator 5.1: Ability to use appropriate tools, techniques and resources.

The evaluation of these attributes will be based on: Assignments, Midterm, Quizzes and Final exam questions. This evaluation is used to indicate your proficiency in all of the attributes as per accreditation requirements.

Course Structure

Lectures

For fall 2020, the course is given online. There will be no synchronous lectures. Lectures are pre-recorded and put available on the course website. During the live lecture time, the professor will answer questions asked by students. You may ask your questions during the live lecture or send them before to allow the professor the time to get answers. The course material is extensive and includes several difficult concepts. Accordingly, it is strongly advised that you stay current in your reading of the textbook and attend the live lectures regularly. That will enhance your learning experience and prevent you from being lost in the lectures. In fact, we suggest that you go through the recorded lectures once before the live lecture. Discussing the material with your fellow classmates, solving problems, and asking questions in the live lectures are also likely to help you.

Tutorials

A live tutorial will take place every week. Tutorials are pre-recorded and put available on the course website. During the tutorial, your tutor will explain the questions in the tutorial notes. We strongly encourage you to attend these tutorials.

Labs

These live lab hours are designated to provide you with personalized help on questions related to the lab assignments. You can also seek the lab instructor's help on general Java/Unix questions. Please make the best use of these lab hours. Due to extreme time limitations, no discussions in relation to the lab assignments will take place during the lecture or with the course instructor, so you must take advantage of these lab hours for any questions related to your lab assignments.

Tentative Coverage and schedule

The coverage and schedule are **tentative** and might change anytime.

1) Coverage

Chapter	Sections
Chap1: Introduction	1.1 to 1.8 & 1.10
Chap2: OS structures	2.1, 2.3 to 2.9
Chap3: Processes	3.1 to 3.6
Chap4: Threads & concurrency	4.1 to 4.3
Chap5: CPU scheduling	5.1 to 5.3
Chap6: Synchronization tools	6.1 to 6.7
Chap7: Synchronization examples	7.1
Chap8: Deadlocks	8.1 to 8.8
Chap9: Main Memory	9.1 to 9.5
Chap10: Virtual Memory	10.1 to 10.6
Chap11: Mass-Storage Structure	11.1 and 11.2
Chap12: I/O Systems	12.1, 12.2, 12.3.3, 12.3.4
Chap13: File-System Interface	13.1 to 13.3
Chap14: File-System Implementation	14.1 to 14.7

2) Schedule

Week	Content	Quiz/exam	Assignments
W1: 8 sep	Chap1		A1 out
W2: 15 sep	Chap2	Quiz #1	
W3: 22 sep	Chap3	Quiz #2	
W4:29 sep	Chap4	Quiz #3	A1 in
W5: 6 oct	Chap5	Quiz #4	A2 out
W6: 13 oct	Chap6 & chap7	Quiz #5	
W7: 20 oct	Chap8	Midterm	
W8: 27 oct	Chap9	Quiz #6	A2 in
W9: 3 nov	Chap10	Quiz #7	A3 out
W10:10 nov	Chap11 & Chap12	Quiz #8	
W11:17 nov	Chap13	Quiz #9	
W12: 24 nov	Chap14	Quiz #10	A3 in
W13: 1 dec	Last class - review		

Plagiarism

The most common offense under the Academic Code of Conduct is plagiarism which the Code defines as *"the presentation of the work of another person as ones own or without proper acknowledgement."*
This could be:

- material copied word for word from books, journals, internet sites, professors course notes, etc.
- material that is paraphrased but closely resembles the original source.
- the work of a fellow student, for example, an answer on a quiz, data for a lab report, a paper or assignment completed by another student.
- a solution or Java code purchased through one of the many available sources.

Plagiarism does not refer to words alone; it can also refer to copying images, graphs, tables, and ideas. Presentation is not limited to written work. It also includes oral presentations, computer assignments and artistic works. Finally, if you translate the work of another person into French or English and do not cite the source, this is also plagiarism.

In Simple Words:
Do not copy, paraphrase or translate anything from anywhere without saying where you obtained it!

Graduate Attributes

As part of either the Computer Science or Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of engineers, computer scientists and information technology professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. The following is the list of graduate attributes covered in this course, along with a description of how these attributes are incorporated in the course:

- **A knowledge base for engineering:** *Demonstrated competence in university level mathematics, natural sciences, engineering fundamentals, and specialized engineering knowledge appropriate to the program.* Knowledge of abstract data types: stacks and queues, trees, priority queues, dictionaries. Data structures: arrays, linked lists, heaps, hash tables, search trees. Design and analysis of algorithms: asymptotic notation, recursive algorithms, searching and sorting, tree traversal, graph algorithms.

- **Problem analysis:** *Ability to use appropriate knowledge and skills to identify, analyze, and solve complex engineering problems in order to reach substantiated conclusions.* Analyze problems and determine their constraints in order to make a choice as to what data structures and algorithms to use for their implementation.

- **Design:** *Ability to design solutions for complex, open-ended engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations.* Use and compose appropriate data structures and algorithms to solve a variety of problems.

- **Use of engineering tools:** *Ability to create, select, apply, adapt, and extend appropriate techniques, resources, and modern engineering tools to a range of engineering activities, from simple to complex, with an understanding of the associated limitations.* Make educated choices as to what data structures and algorithms to use to solve problems following their respective strengths and constraints.

Learning Objectives

- **Knowledge base:** Demonstrate competence in fundamentals of data structures and algorithms.

- **Problem analysis:** Analyze and state model limitations and elements of uncertainty. Formulate and calculate qualitative and quantitative qualities of the problems inputs and outputs. Estimate computational complexity. Evaluate and pick the most appropriate approach based on relevant criteria.

- **Design:**
  - Critique/evaluate many possible diverse solutions and use techniques to evaluate different solutions with sound arguments related to the problems requirements and constraints. Demonstrate thinking outside the box to create innovative solutions.
  - Develop a system architecture adapted to the systems application context and its requirements and constraints. Development and specification of internal and external software interfaces at different modularity levels. Describe a solution that presents enough details for implementation.
  - Write code according to design. Validate implemented systems against system requirements, specifications and constraints, as well as interface specifications.

- **Use of Engineering tools:** Demonstrate appropriate operational use of tools (e.g. algorithms, abstract data types, data structures, asymptotic complexity analysis) for specific tasks in a laboratory environment.

Important Notes

1. **One credit** represents, for an average student, a minimum of 45 hours of workload spread across the various academic activities (Source: Article 16.1.2 of the Undergraduate Calendar.) For an average student, this suggests a minimum of 135 hours of workload for a 3-credit course, including the time spent in lectures, tutorials, laboratories, examinations, and personal and team work.

2. Assignments will consist of a theoretical and a programming part. Each student must independently and separately prepare and submit the theory part of the assignment. This is also the case for the programming part, unless it is allowed to work in a group (maximum

2 students), which will be indicated in the assignment. In such case, both team members must prepare the programming assignment but only one submission is to be made by either member.

For the programming part a demo for about 5 minutes will take place with the marker. You (or **both** members, if groups are permitted) **must** attend the demo and be able to explain your program to the marker. Different marks may be assigned to teammates based on this demo. The schedule of the demos will be determined and announced by the markers, and students must contact the marker to reserve their time slot. **Demos are mandatory. Failure to do your demo will entail a mark of zero for that assignment regardless of your submission. Missing your reserved demo time, will similarly result in a zero mark for the assignment regardless of your submission. There will be no substitution/replacement for a missed demo time.**

3. **Criteria used in evaluation of assignments:**
   - Correctness and Testing: the program should conform to the specification given in the assignment. This includes the proper handling of special cases and error conditions and the providing of correct results. The submitted test cases take into consideration special cases and error conditions.
   - Design: the program should be constructed from coherent, independent, and decoupled functions. A function should usually access only its own parameters and local variables.
   - Style: the program should be general-purpose and well-organized.
   - Documentation and Layout: The documentation should consist of a well-annotated program and clearly formatted output. Helpful identifiers and a clear layout are part of documentation. The documentation should include the description of your design and the algorithm implemented.
   - Efficiency: The program must implement the most appropriate method.
   - Program-User Interface: The program should be easy to use.
4. **Programming assignments:** For all programming components of your assignments, you need to use Java version 8.0 or later. You may use the CSE computing facilities by remotely accessing them.
   AITS has launched a new service, Remote Access Computer Labs. This new service enables Remote Desktop Connections to the computers in the GCS public computer labs. GCS students can use this service to run software for learning, research and completing assignments remotely on lab computers.
   Please refer to the following web page on how to use this service.

   https://www.concordia.ca/ginacody/aits/support/faq/connect-from-home.html
   Should you have any questions about this service, please contact Service Desk at help@concordia.ca

If you have your own computer and prefer to use it, you may do so, but be aware that your programs must compile and run with Java 8.0, or later version.

Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must also contain your name(s) and student ID(s). Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. Only electronic submissions will be accepted. Students will have to submit their assignments (one copy per group in case groups are allowed) using the Moodle. **Assignments uploaded to an incorrect folder will not be marked and result in a zero mark. No resubmissions will be allowed.**

For the Java programming assignments, you have to submit the complete source code **and** the compiled files, which must be executable without changes. If this is violated, you will get a zero mark for these parts of the assignments.