Operating Systems
ECSE 427/COMP 310 Fall 2017

Instructor:

Dr. Rola Harmouche
Department of Electrical and Computer Engineering
Office: McConnell Engineering Building, Room 637
Email: rola.harmouche@mcgill.ca
Office Hours: Mondays and Wednesdays, 3:30 PM – 4:30 PM. Dr. Harmouche is also available outside of office hours. Please contact to arrange a time.

Lectures:

- Monday, Wednesday: 01:05 PM-02:25 PM, Trottier Room 0100 (ENGTR 0100), September 5-December 7.

- Thursday December 7 01:05 PM-02:25 PM, Trottier Room 0100 (ENGTR 0100)

The first lecture will be Wednesday September 6 and the last lecture will be Thursday December 7.

Midterm:
Wednesday October 18, 1:05 PM – 2:25 PM, rooms TBD.

Tutorials:

- Monday 02:35 PM-03:25 PM ENGTR 1090,

- Thursday Dec 7: M02:35 PM-03:25 PM ENGTR 1090.

Tutorials start on Monday September 11, 2017

Teaching Assistants:

Information on teaching assistants is available on my courses.

Credits: 3. Course Hours: (3,1,5) (Lectures, Labs and tutorials, outside work).

Prerequisites: ECSE 322 or COMP 273.

Required Textbook: A. Tenenbaum and H. Bos, Modern Operating Systems, 4th Edition, Pearson, 2015.

Web page: Information for the course will be posted on the course myCourses web page. The myCourses page for the course should be the first place you look to get information about the course. The use of email and discussion board for communication with the

instructors or the teaching assistants is encouraged. You must send email from myCourses or from your McGill Email address.

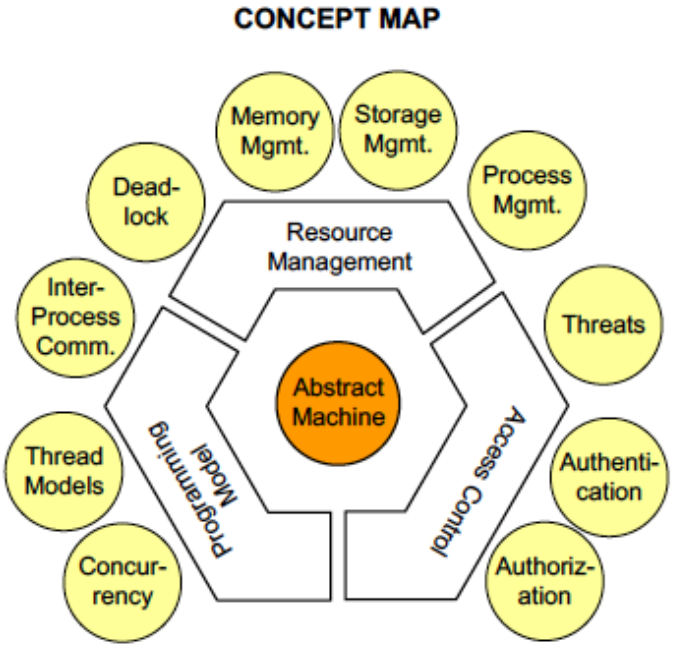
Course Description and Learning Outcomes:

Description: This is an introductory course in computer operating systems. In this course we will study the theoretical and practical concepts behind modern operating systems. In particular, we will study the basic structure of an operating system, its components, design strategies, algorithms and schemes used to design and implement different components of an operating system. Major components to be studied include: processes, inter-process communication, scheduling, memory management, virtual memory, storage management, network management, and security.

Primary learning outcome: To get a clear understanding of the major principles/ algorithms that underlie an operating system and how they interplay within it.

Secondary learning outcomes: After taking this course, you should be able to: (i) identify the core functions of operating systems and how they are architected to support these functions, (ii) explain the algorithms and principles on which the core functions are built on, (iii) explain the major performance issues with regard to each core function, and (iv) discuss the operating system features required for particular target applications.

Course Content:



Course Schedule:
The table below shows a **tentative course schedule.** I will make a "best effort" to stick to the proposed schedule. Except for the Midterm, dates might change.

Week	Topics	Reading Material	Comments
1	OS Introduction: OS tour; simple OS; beyond simple OS	1.5, 1.7	1.8 highly recommended
2	Process Concept: processes, introduction to process scheduling, IPC, Threads – multithread programming, issues	1.6, 2.1, 2.2	1st Mini Assignment out
3	Synchronization: Critical section problem, Peterson's solution, synchronization hardware, mutex, semaphores, monitors, well-known problems in synchronization	2.3, 2.5	
4	Deadlocks: models, characterization, prevention, avoidance, detection, recovery	Chap. 6	2nd Mini Assignment out
5	File Systems: file concept, file system structure, implementation, allocation methods, free-space management, efficiency & performance, access methods, directories, disk structure, disk scheduling, etc	4.1-4.5	
6	Advanced File Systems: Log-structured file systems, fault tolerance, examples of modern file systems, SSD issues	Same as week 5	Midterm
7	CPU Scheduling: basic concepts, scheduling criteria, algorithms, thread scheduling, real-time scheduling, lottery scheduling, stride scheduling, performance of scheduling algorithms	2.4, 10.3.4	
8	Main Memory: swapping, memory allocation, segmentation, paging	3.1-3.7	Programming Assignment out
9	Virtual Memory: demand paging, copyon-write, page replacement, allocation of frames, thrashing	Same as week 8	
10	Virtualization and Clouds: Virtualization requirements, hypervisors, memory virtualization, I/O virtualization, clouds	7.1-7.11	
11	Security: goals of protection, domain of protection, access matrix, implementation, access control, capability-based systems	9.1-9.4	
12	Security: security problem, program threats, system & network threats, user authentication	9.7-9.10	

Instructional Method:
The course will consist of three hours of instructor led classes per week together with a maximum of one hour of tutorial per week taken by the TAs. The class time will be devoted to the presentation and development of new concepts and the application of these concepts to examples and problems, while the tutorials will discuss solutions to the programming assignments/projects and written assignments. The primary focus of the tutorials is to provide sufficient "how-to" knowledge through the discussion of the assignments to help in the development of the programming project series.
Students are strongly encouraged to use the My Courses discussion groups to talk about the programming and written assignments. These discussion groups will be monitored by the TAs and by the instructor for providing the necessary answers.

Evaluation Procedures:

Activity	Weight (in percent)
Mini Programming Assignment 1:	7%
Mini Programming Assignment 2:	8%
Programming Assignment:	15%
Midterm (during class time)	20%
Final (comprehensive)	50%
Total	100%

Double Grading Policy: This course has a significant portion of the grade allocated for the programming component. You are expected to submit only your work in these assignments. You can receive advice or tips from others (instructor, teaching assistants, or peers), but the final submission should be yours. You are expected to know all the design decisions in the program and explain all aspects of the program handed in as part of the assignment. To test this condition, we will randomly select some students and ask them to explain their programming assignments. The eventual marks for an assignment will be the minimum of the two marks. For example, if 85 is the marks obtained in the first (normal) evaluation of the programming assignment and 50 is the marks obtained in the second evaluation, then effectively you have 50.

Late Assignment Policy: There will be two deadlines for each assignment: proper deadline and cut-off date. After the proper deadline, there will be a penalty of 10% for each day the assignment is late until the cut-off date. After the cut-off date, the assignment cannot be handed in. No individual requests for extensions will be granted unless they are for medical reasons.

The deadlines will be set for 11:55 pm or 11:59pm. Please observe the time and date very carefully. It is your responsibility to make sure that the assignment is properly submitted via the WebCT.

Regrading Policy: If you find your assignments or exams are not marked according to the marking scheme, you are encouraged to consult me or the TAs. When you resubmit

your assignment or exam for regarding, we reserve the right to regrade the full exam or assignment without restricting the attention to the disputed portion.

Academic Integrity: McGill University values academic integrity. Therefore, all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures (see www.mcgill.ca/students/srr/honest/ for more information). (approved by Senate on 29 January 2003)

In accord with McGill University's Charter of Students' Rights, students in this course have the right to submit in English or in French any written work that is to be graded." (approved by Senate on 21 January 2009 - see also the section in this document on Assignments and evaluation.)