COMP 346 (Operating Systems)

Course Outline
Fall 2019

Professor in charge	Dr. Dhrubajyoti Goswami E-mail: goswami@cs.concordia.ca Web : http://www.cs.concordia.ca/~goswami/ Office: EV 3.145 Tel: (514) 848-2424 Ext. 7882
Tutors	TBA
Markers	TBA (Refer to Moodle)

COURSE DESCRIPTION

Pre-requisites: COMP 228/SOEN 228 & COMP 352

Fundamentals of operating system functionalities, design and implementation. Multiprogramming: processes and threads, context switching, queuing models and scheduling. Interprocess 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.

OBJECTIVES

This course has two components: a theory component to teach the concepts and principles that underlie modern operating systems, and a practice component to relate theoretical principles with operating system implementation and use. In the theory component, the student will learn about 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 The practice component will complement the theory component through programming assignments illustrating the use and implementation of these concepts. The programming language is Java.

COURSE MATERIALS

The textbook for the course is: *Operating System Concepts*, 10th Edition, by: Silberschatz, Galvin, Gagne. Published by John Wiley & Sons, 2018.

Other references:
Operating Systems by *Gary Nutt*, published by *Addison-Wesley*.

The Logical Design of Operating Systems, by Bic and Shaw, published by: Prentice Hall.

Operating Systems Internals and Design Principles by: William Stallings, published by: Pearson.

COURSE SCHEDULE

We plan to cover selected topics from chapters 1-10 and 13-15 of the textbook. The following is a tentative weekly breakdown of the topics covered:

Week	**Topics**
Week 1-2	Introduction to Operating Systems (OS); OS structures; Process management: processes and Threads.
Week 3-5	Process-Thread synchronization: the critical section problem, hardware and software solutions to the critical section problem.
Week 6-8	CPU scheduling algorithms. Deadlocks: prevention and avoidance schemes; recovery from deadlocks.
Week 9-11	Memory management: virtual memory based on paging and segmentation; hybrid schemes; demand paging.
Week 12-13	File systems: interface and implementation; Unix file system.

GRADING

Assignments (30%): There will be tentatively 3 Theory and 3 Programming assignments (in Java). **You must pass the assignments to pass the course.**

Midterm (20%): There will be one midterm exam The exam will include multiple choice questions, and may also include other types of questions. There will be no substitute midterm.

Final Exam (50%): There will be one final exam. The exam will include multiple choice questions, and may also include other types of questions. **You must pass the final exam and pass in the total marks obtained to pass this course.**

The grading of the course will be done based on the relative percentages assigned to the assignments 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. 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.

In the event of extraordinary circumstances beyond the University's control, the content and/or evaluation scheme in this course is subject to change.

COURSE STRUCTURE

Lectures

The lectures are a key component of the course, and you are advised to attend the lectures regularly and attentively. The course material is extensive and includes several difficult concepts. Accordingly, we will try to utilize the lecture time in doing things that you would not get by simply reading the book (explaining difficult concepts, giving you alternate perspectives, relating course material to other fields, etc.) rather than merely repeating facts that you can simply read from the text book. It is strongly advised that you stay updated in your reading of the textbook, and attend the lectures regularly. That will enhance your learning experience and prevent you from being lost in the lectures. In fact, we suggest that you casually go through the textbook sections once before the lecture. You are also strongly advised to go and read the material thoroughly after the lecture. Discussing the material with your fellow classmates, solving problems, and asking questions in the lectures are also likely to help you.

Tutorials
The tutorials will take place every week. During the tutorial, your tutor will explain the Lab assignments and answer questions related to the course, and more specifically related to the programming assignments. We strongly encourage you to attend these tutorials.

Labs

The lab hours are indicated in the course schedules. These designated lab hours are to be used to get personalized help on questions related to the programming assignments. You can also seek their help on general Java/Unix questions. Please make the best use of these lab hours.

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: 1) Assignments, 2) Midterm, and 2) Final exam questions. This evaluation is used to indicate your proficiency in all of the attributes as per accreditation requirements.

MOODLE AND OTHER RESOURCES

You will be able to access the course material and general discussion for this course via Moodle, which you can access through your MyConcordia portal: www.myconcordia.ca.

All assignment submissions are electronic via the electronic assignment submission system (https://fis.encs.concordia.ca/eas). Other useful information will be posted from time to time via Moodle.

ACADEMIC MISCONDUCT

Any form of academic misconduct will not be tolerated and will be handled as per University guidelines. You are required to be familiar with the following information: http://www.concordia.ca/students/academic-integrity.html