Course Title: **COSC 4361: Operating Systems**

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the principles, purposes, and organization of operating systems. The goal is to prepare students an understanding of the theory as well as practices of the design and implementation of operating systems software.

II. Competencies Addressed

Students in this course develop conceptual and programming skills necessary for continued success in computer science. The skills enhances their abilities to appreciate the theory and practices of operating systems common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both conceptual reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4361: Operating Systems is concerned with the study of the principles, purposes, and organization of operating systems, including processes, tasks, scheduling, interprocess communication, synchronization, mutual exclusion, memory management, device management, file systems, security and protection, multi-CPU systems, computer networking, and distributed computing.

IV. Requirements Fulfilled

This course satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the first semester of the senior year.

V. Required Prerequisites

- GEIT 1311: Computer Organization
- COSC 3411: System Programming
- COSC 3421: Data Structures

VI. Learning Outcomes

In this course, students learn:

- To understand the underlying system structures of operating systems.
- To understand the concepts of threads and processes in computer systems.
- To learn the design requirements on storage management in computer systems.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the compiler construction, to enhance the student's programming techniques to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Three programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 30% on in-class examinations, 30% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (3 hours per week) **Class:** 3
Lab: 0

IX. Topics to Be Covered

A. Basic Concepts
   1. Computer system structures
   2. Operating system structures

B. Threads and processes
   1. Processes, threads, and address spaces
   2. Thread and process coordination
   3. CPU scheduling
   4. Deadlocks

C. Storage management
   1. Memory management
   2. Virtual memory
   3. File systems
   4. Input-output systems
   5. Mass storage structure

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Silberschatz, A., P., B. Galvin, G. Gagne. *Operating Systems Concepts*. _____: Wiley, 2002.
ISBN: 0-471-41743-2

B. Alternative Textbooks

Tanenbaum, A. *Modern Operating Systems*. _____: Prentice Hall, 2001.
ISBN 0-13-031358-0.

C. Supplemental Print Materials

None

D. Supplemental Online Materials

As available from the publishers.