1. Course description
A functional level view of multiprocessing operating systems including processor, memory, peripheral, and file systems management in timesharing, real time, and distributed systems targeted for various hardware. Several programming assignments will be given for implementation on a UNIX/Linux platform (other OS platforms are negotiable, but in-class examples will be targeted for UNIX/Linux and structured in 'C').

Prerequisites: 91.308 or equivalent, and programming skill with 'C' or 'C++'

The above is the course description as it appears in the Graduate School Catalogue. In this course, particular emphasis will be placed on the problems of concurrency control in both centralized and distributed systems. It is assumed that the student has a working knowledge of operating systems at the level of an undergraduate course; should this not be the case, a text such as one of the following is suggested for background knowledge.

Avi Silberschatz, Peter Glavin and Greg Gagne.
Operating System Concepts, 9th Ed.
(ISBN 978-1118063330)

William Stallings.
Operating Systems, 9th Ed.
Pearson Education, 2018
2. Required Textbook
Mukesh Sinhgal and Niranjan Shivarati.  
*Advanced Concepts in Operating Systems.*  
(ISBN 007057572X)

Readings in the text will be supplemented from time to time by assigned readings of classic papers in the field; copies of these papers will be available through sources discussed in class.

3. Tests and Grading
In addition to the assigned readings, there will be two exams and several programming assignments. The proportions of your cumulative grade will be as follows:

- Full class Exam: First Half Material 30%
- Full class Exam: Second Half Material 30%
- Assigned Programming Projects 40%

The first exam will cover the first half of the course and the second will cover only the second half of the course. All exams will be closed book and notes. See the attached schedule for dates.

4. Assignments
Regular assignments will include readings in the course textbook and assigned papers. A detailed description of the programming assignments will be supplied separately. Programming assignments are graded from 0 – 10 points. For each one of our class meeting days that an assignment or project is late, 20 % of the total points (2 points) will be deducted from the points received. This will continue until midnight of the third class meeting day after the assignment is due (approximately 2 weeks, except for the final assignment which will no longer be accepted after midnight (11:59:59 PM) on Sunday, December 17), after which the assignment or project will no longer be accepted. The cutoff date for each assignment is included in the attached course outline. An assignment will be graded as zero points if not submitted by a cutoff date.

5. Academic Dishonesty
In this course, all work is to be each student's own. Students should therefore be familiar with the University's rules on academic dishonesty, which can be found in the *Graduate School Catalog*
and in the Schedule of Classes. In particular, plagiarism will not be tolerated! Any student caught plagiarizing another’s work will automatically receive a grade of F for the course. If you are unsure as to what constitutes plagiarism, it is your responsibility to check with the instructor. Other forms of dishonesty will result in similar actions. You may collaborate with your classmates on the design and results of the programs you will write in this course, but each student must implement these programs alone. Submission of shared student code is not permissible, and will result in a grade of F for the course. Help files are typically provided for each programming assignment, and students are encouraged to cut and paste useful code from these help files into their assignment submissions, but all other code must be the specific work of each student.

6. No Posting of Solution Code Policy:
You are not allowed to post solution code to problem sets assigned in this class in public places (e.g. Github). This includes your own solutions as well as solutions that may be provided by the instructors.

The University policy on academic integrity states that assisting students in their own acts of academic dishonesty is itself a violation of academic integrity. See Academic Misconduct Subject to Disciplinary Action, 1(f).

Please note that this is typical policy at premier computer science departments. E.g.:
- Princeton COS 126. "Your work must never be shown or communicated to anyone who is taking COS 126 now or who might take COS 126 in the future. ... You must never place your work in any public location (including websites, leaving printouts in a classroom, etc.). ... The rules ... continue to apply even after this semester is over."
- Harvard CS50. "Not reasonable: Providing or making available solutions to problem sets to individuals who might take this course in the future."
- MIT 6.01. "Students should never share their solutions (or staff solutions) with other students, including through public code repositories such as Github." (emphasis in the original)

Thus: Do not publish your solutions to problem sets.
Doing so will be considered an act of academic dishonesty and you will receive a grade of F for the course.

Note: You may save your work to code repositories provided that they are private and cannot be retrieved by others. I encourage you to sign up for Github's Student Developer Pack, which allows you to create private repositories (among other benefits). It’s free to students.