Instructor	**Dr. Yusuf Sahillioğlu**
Lecture Hours	**Friday: 13:40 – 16:30**
TA Contacts	**Name: Eugene O. Casmin** **Email: eugene.owilla@metu.edu.tr**
TA Office Hours	**Thursday: 14:40 – 16:30, or by appointment via email.**

Catalog Description	Introduction to Operating Systems. Memory Management. Process Management. Concurrent Processes. Deadlocks. Processor Management. I/O and Device Management. File Management and File Systems. Introduction to Distributed Operating Systems. Synchronization in Distributed Systems. Distributed File Systems. Overview of contemporary OS technology.
Course Goal	In line with catalog description, the objectives of the course are to: • Introduce the functions of an operating system, and discuss the evolution of the operating systems. • Introduce process and thread concepts, process scheduling and management, inter-process communication, synchronization, deadlocks. Introduce memory management principles (segmentation, paging, swapping). • Introduce file systems and Inputs/Output systems. • Introduce Distributed Operating Systems and their challenges. • Overview of contemporary OS technology such as virtualization technologies.
Learning Outcomes	At the end of this course, students will be able to: • Understand design and implementation of operating systems. • Understand data structures and memory organization mechanisms of complex software systems. • Understand resource sharing mechanisms of a complex software system. • Understand concurrent data exchange mechanisms of a complex software system.
Prerequisite	CNG 331 (Official)
Text Book	Main Text: Operating System Concepts, by Silberschatz, Galvin and Gagne Wiley. 9th Ed.
References	Auxiliary Text: Modern Operating Systems, Third Edition, Andrew S. Tanenbaum, Pearson Education, 2008.
Teaching Format	Three 50-minute lectures per week.

Course Topics	1. Introducing modern operating systems. 2. Introduction to processes. 3. Threads. 4. Thread implementation and multi-threaded programming. 5. Inter-process communication. 6. Synchronization and Deadlocks. 7. Memory Management (paging, segmentation, replacement algorithms). 8. File Systems and Raids. 9. Input/output. 10. Introducing Distributed OSs. 11. Virtualization overview
Weekly Schedule	Week 1: Introduction and OS Overview Week 2: Processes and Threads Week 3: Thread implementation and multi-threading programming Week 4: Process and thread scheduling Week 5: Inter-process communication Week 6: Synchronization and Deadlocks Week 7: Memory Management - Part1 Week#8: Memory Management – Part2 Week#9: File Systems and Raids I Week#10: File Systems and Raids II Week#11: Input/output Week#12: Introduction to Distributed operating systems Week#13: Virtualization overview & *Final project presentations.* Week#14: *Final project presentations (cont.). **
Computer Usage	Any Development Environment
Category content	Mathematics and Basic Sciences 20% Engineering Sciences 80% Humanities and Social Sciences 0 % Departmental 0% Engineering Design 0%
Grading Policy	2 ***individual*** programming assignments: 40%, Midterm Exam 25%, Final Exam: 30%, Attendance/participation: 5%.

Relationship of Course to Student Outcomes	• SO (l) – PI-l3. • Analyze the architectures of real systems, such as operating systems, database management systems, network protocols, compilers and graphic engines. • SO (c) – PI-c2. • Evaluate and adapt standard algorithms, e.g., sorting, searching, string processing and graph processing, for realistic tasks. • SO (k) – PI-k4. • Use a programming language in non-imperative paradigm, e.g., functional and logic. • SO (l) – PI-l2. • Evaluate the quality attributes, such as dependability, efficiency, usability and security, of systems.
Key Course Dates	***NB: exam dates will be announced on CET.**
Last date updated	15th March, 2021.

Course Rules

1. **ATTENDANCE**: Students attending less than 75% of the sessions may not be allowed to sit the final exam.
2. **NO PLAGIARISM**, this means NO Cheating, NO Copying, NO Rewording, NO Paraphrasing, Proper citations MUST be made if needed, …etc. More details see (http://www.plagiarism.org/plag_article_what_is_plagiarism.html). When a breach of the code of ethics occurs (cheating, plagiarism, deception, etc.), student will be added to the BLACK list, and instructor can:
   a) Give a "zero" grade for the relevant exam, project, assignment
   b) Give a "zero" grade for a larger part or all of the coursework,
   c) Give a failing letter grade for the course,
   d) Forward the case to the discipline committee.
3. **RESPECT** Lectures, Tutorials, and Exams times.
4. **RESPECT YOUR PEERS**. When a student asks a question or gives a presentation, other students should listen and discuss.
5. ***No Late Submission*** for assignments and projects is accepted unless delay is caused by unforeseen events like sever illness. Proper documents have to be presented.

Academic Integrity

Copying, communicating, or using disallowed materials during an exam is cheating, of course. Students caught cheating on a midterm or final exam will be reported to the campus disciplinary committee. Students may not leave the classroom during exams; any student leaving the classroom is leaving the exam.

Academic integrity is a more complicated issue for programming assignments, but one we take very seriously. Students naturally want to work together, and it is clear they learn a great deal by doing so. Getting help is often the best way to interpret error messages and find bugs, even for experienced programmers. In response to this, the following rules will be in force for programming assignments:

- Students are allowed to work together in designing algorithms, in interpreting error messages, and in discussing strategies for finding bugs, but NOT in writing code.
- Students may not share code, may not copy code, and may not discuss code in detail (line-by-line or loop-by loop) at any time, i.e., while it is being written or afterwards.
- Similarly, students may not receive detailed help on their code from individuals outside the course. This restriction includes tutors, students from prior terms, Internet resources, etc. Students may not show their code to other students as a means of helping them. Sometimes good students who feel sorry for struggling students are tempted to provide them with "just a peek" at their code. Such "peeks" often turn into extensive copying, despite prior claims of good intentions.
- Students may not leave their code (either electronic versions or printed copies) in publicly accessible areas. Students may not share computers in any way when there is an assignment pending.

We use various code comparison tools including automated tools to help spot assignments that have been submitted in violation of these rules. The tool takes all assignments from all sections and all prior terms and compares them, highlighting regions of the code that are similar.

We (the instructors and the teaching assistants) check flagged pairs of assignments very carefully ourselves, and make our own judgment about which students violated the rules of academic integrity on programming assignments. When we believe an incident of academic dishonesty has occurred, we contact the students involved. All students caught cheating on a programming assignment (both the copier and the provider) will receive an automatic 0 for that assignment. No excuses, no discussions, no exceptions!

For non-programming assignments, students have to use their own ideas, words, algorithms, tables, and figures. If citation is needed, proper and accurate citation for the used information sources must be given. Any sort of plagiarism will not be tolerated. This means no copying, no rewording, no paraphrasing, or giving false/inaccurate information sources. For more details about plagiarism, please see http://www.plagiarism.org/plag_article_what_is_plagiarism.html. For further on METU NCC's rules concerning academic code of ethics, please see http://www.ncc.metu.edu.tr/academic/acadcode-of-ethics.php.