Course Description: Credit 3 hours. Prerequisite: Computer Science 375 and 390. Design and implementation of operating systems. Topics include process management, processor management, memory management, device management, file management, process synchronization and interprocess communication, and user interface. Other issues such as distributed computing and system performance may be discussed.

Minimum Topics:
- Overview: Role and purpose of operating systems; history of operating system development; functionality of a typical operating system; design issues (efficiency, robustness, flexibility, portability, security, compatibility)
- Use and basic administration of an operating system (excluding Microsoft Windows)
- Basic principles: Structuring methods; abstractions, processes, and resources; design of application programming interfaces (APIs); device organization; interrupts; user/system state transitions
- Concurrency: The idea of concurrent execution; states and state diagrams; implementation structures (ready lists, process control blocks, and so forth); dispatching and context switching; interrupt handling in a concurrent environment; writing concurrent programs
- Mutual exclusion: Definition of the “mutual exclusion” problem; deadlock detection and prevention; solution strategies; models and mechanisms (semaphores, monitors, condition variables, rendezvous); producer-consumer problems; synchronization; multiprocessor issues
- Scheduling: Preemptive and nonpreemptive scheduling; scheduling policies; processes and threads; real-time issues
- Memory management: Review of physical memory and memory management hardware; overlays, swapping, and partitions; paging and segmentation; page placement and replacement policies; working sets and thrashing; caching

Learning Objectives: Students will be able to:
- Identify significant continuing trends in the history of the computing field, especially regarding operating systems.
- Define the necessary components and functions of an operating system.
- Use both Windows and Unix-class systems.
- Describe the similarities and differences between Windows and Unix-class systems.
- Analyze operating system requirements and recommend an appropriate operating system to meet the requirements.
- Install a Unix-class operating system.
- Write programs and manage applications in a Unix-class operating system.
- Explain the objectives and functions of modern operating systems.
- Describe how operating systems have evolved over time from primitive batch systems to sophisticated multiuser systems.
- Analyze the tradeoffs inherent in operating system design.
- Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve.
- Discuss networked, client-server, distributed operating systems and how they differ from single user operating systems.
- Identify potential threats to operating systems and the security features design to guard against them.
- Describe how issues such as open source software and the increased use of the Internet are influencing operating system design.
- Explain the concept of a logical layer.
- Explain the benefits of building abstract layers in hierarchical fashion.
- Defend the need for APIs and middleware.
- Describe how computing resources are used by application software and managed by system software.
- Contrast kernel and user mode in an operating system.
- Discuss the advantages and disadvantages of using interrupt processing.
- Compare and contrast the various ways of structuring an operating system such as object-oriented, modular, micro-kernel, and layered.
- Explain the use of a device list and driver I/O queue.
- Describe the need for concurrency within the framework of an operating system.
- Demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks.
- Summarize the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each.
- Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.
- Summarize the various approaches to solving the problem of mutual exclusion in an operating system.
- Be familiar with classic concurrent programming problems, and demonstrate the ability to write a program using multiple concurrent threads and/or processes.
- Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system.
- Discuss the utility of data structures, such as stacks and queues, in managing concurrency.
- Explain conditions that lead to deadlock.
- Compare and contrast the common algorithms used for both preemptive and nonpreemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes.
- Discuss the types of processor scheduling such as short-term, medium-term, long-term, and I/O.
- Describe the difference between processes and threads.
- Compare and contrast static and dynamic approaches to real-time scheduling.
- Discuss the need for preemption and deadline scheduling.
- Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and other problems unrelated to computing.
- Explain memory hierarchy and cost-performance tradeoffs.
- Explain the concept of virtual memory and how it is realized in hardware and software.
- Summarize the principles of virtual memory as applied to caching, paging, and segmentation.
- Defend the different ways of allocating memory to tasks, citing the relative merits of each.
- Describe the reason for and use of cache memory.
- Compare and contrast paging and segmentation techniques.
- Discuss the concept of thrashing, both in terms of the reasons it occurs and the techniques used to recognize and manage the problem.
- Analyze the various memory portioning techniques including overlays, swapping, and placement and replacement policies.
- Explain the distributed paradigm.
- Explain one simple distributed algorithm.
Grading Policy:

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs / Homework</td>
<td>30%</td>
</tr>
<tr>
<td>Progress Reports / Documentation</td>
<td>30%</td>
</tr>
<tr>
<td>Project / Presentations / Demos</td>
<td>40%</td>
</tr>
</tbody>
</table>

A: 90 – 100 %; B: 80 – 89 %; C: 70 – 79 %; D: 60 – 69 %; F: 0 – 59 %

ADA Accommodation: If you are a qualified student with a disability seeking accommodations under the Americans with Disabilities Act, you are required to self-identify with the Office of Disability Services, Room 203, Student Union. More information can be obtained at this web address, [http://www.selu.edu/StudentAffairs/DisabilityServices/](http://www.selu.edu/StudentAffairs/DisabilityServices/)