UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS/DESCRIPTION

Department and Course Number	CSCI 4500
Course Title	Operating Systems
Course Coordinator	Stanley Wileman
Total Credits	3
Repeat for Credit?	No
Date of Last Revision	June 16, 2015

1.0 Course Description Information

1.1 Catalog Description:
Overview of operating systems and operating system principles. Concurrency, process scheduling and dispatch, memory management, security and protection, virtual machines, device management, and file systems.

1.2 Prerequisites of the course:
CSCI 3710 (Introduction to Digital Design and Computer Organization), CSCI 3320 (Data Structures), and MATH 1950 (Calculus I). CSCI 4350 (Computer Architecture) is recommended.

1.3 Overview of content and purpose of the course:
An operating system is an abstraction of computer system hardware; it manages the sharing of various hardware and software resources among the users of the computer system. The parallel history of hardware and operating system development introduces many key concepts including, for example, processor modes, direct memory access (DMA), device controllers, and virtual memory. Basic approaches to kernel organization and implementation are considered. This is often the first course in which students encounter concurrency and concurrent programs. Additional topic areas include system performance evaluation (particularly relating to processor and memory management), security, virtualization, resource allocation and scheduling, and file systems.

1.4 Unusual circumstances of the course
None

2.0 Course Justification Information

2.1 Anticipated audience / demand
This is a required course for computer science majors. It is also required for computer engineering and information assurance majors. It is often available as an elective for students in other computing-related disciplines.

2.2 Indicate how often this course will be offered and the anticipated enrollment:
This course is usually offered at least once each semester (fall, spring, and summer) with a typical enrollment of at least 30 students.

2.3 If it is a significant change to an existing course, please explain why it is needed:

3.0 List of performance objectives stated in learning outcomes in a student's perspective:

1. Explain the objectives and functions of modern operating systems.
2. Analyze the tradeoffs inherent in operating system design.
3. Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve.
4. Discuss networked, client-server, distributed operating systems and how they differ from single user operating systems.
5. Identify potential threats to operating systems and the security features design to guard against them.
6. Explain the concept of a logical layer.
7. Explain the benefits of building abstract layers in hierarchical fashion.
8. Describe the value of APIs and middleware.
9. Describe how computing resources are used by application software and managed by system software.
10. Contrast kernel and user mode in an operating system.
11. Discuss the advantages and disadvantages of using interrupt processing.
12. Explain the use of a device list and driver I/O queue.
13. Describe the need for concurrency within the framework of an operating system.
14. Demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks.
15. Summarize the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each.
16. Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.
17. Summarize techniques for achieving synchronization in an operating system (e.g., describe how to implement a semaphore using OS primitives).
18. Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system.
19. Create state and transition diagrams for simple problem domains.
20. Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes.
21. Describe relationships between scheduling algorithms and application domains.
22. Discuss the types of processor scheduling such as short-term, medium-term, long-term, and I/O.
23. Describe the difference between processes and threads.
24. Compare and contrast static and dynamic approaches to real-time scheduling.
25. Discuss the need for preemption and deadline scheduling.
26. Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and problems beyond computing.
27. Explain memory hierarchy and cost-performance trade-offs.
28. Summarize the principles of virtual memory as applied to caching and paging.
29. Evaluate the trade-offs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed.
30. Defend the different ways of allocating memory to tasks, citing the relative merits of each.
31. Describe the reason for and use of cache memory.
32. Discuss the concept of thrashing, both in terms of the reasons it occurs and the techniques used to recognize and manage the problem.
33. Articulate the need for protection and security in an OS.
34. Summarize the features and limitations of an operating system used to provide protection and security.
35. Explain the mechanisms available in an OS to control access to resources.

36. Explain the concept of virtual memory and how it is realized in hardware and software.
37. Differentiate emulation and isolation.
38. Evaluate virtualization trade-offs.
39. Discuss hypervisors and the need for them in conjunction with different types of hypervisors.
40. Explain the key difference between serial and parallel devices and identify the conditions in which each is appropriate.
41. Identify the relationship between the physical hardware and the virtual devices maintained by the operating system.
42. Explain buffering and describe strategies for implementing it.
43. Differentiate the mechanisms used in interfacing a range of devices (including hand-held devices, networks, multimedia) to a computer and explain the implications of these for the design of an operating system.
44. Describe the advantages and disadvantages of direct memory access and discuss the circumstances in which its use is warranted.
45. Identify the requirements for failure recovery.
46. Implement a simple device driver for a range of possible devices.
47. Describe the choices to be made in designing file systems.
48. Compare and contrast different approaches to file organization, recognizing the strengths and weaknesses of each.
49. Summarize how hardware developments have led to changes in the priorities for the design and the management of file systems.
50. Summarize the use of journaling and how log-structured file systems enhance fault tolerance.
51. Describe what makes a system a real-time system.
52. Explain the presence of and describe the characteristics of latency in real-time systems.
53. Summarize special concerns that real-time systems present, including risk, and how these concerns are addressed.
54. Explain the relevance of the terms fault tolerance, reliability, and availability.
55. Outline the range of methods for implementing fault tolerance in an operating system.
56. Explain how an operating system can continue functioning after a fault occurs.
57. Describe the performance measurements used to determine how a system performs.
58. Explain the main evaluation models used to evaluate a system.

4.0 Content and Organization Information:

	Contact hours
4.1 Introduction to Operating Systems	6.0
4.1.1 Views of an operating system	
4.1.1.1 Resource management	
4.1.1.2 Extended machine	
4.1.2 What is not in an operating system	
4.1.2.1 Command interpreter	
4.1.2.2 Development tools	
4.1.2.3 Applications	
4.1.3 Operating system history	

4.1.3.1 Stand-alone systems
4.1.3.2 Batch systems
4.1.3.3 Multiprogramming and time-sharing systems
4.1.3.4 Personal computers, networks, distributed systems
4.1.4 Basic operating system concepts
4.1.4.1 Processes, tasks, threads
4.1.4.2 Files
4.1.4.3 The command interpreter
4.1.4.4 Other user interfaces
4.1.5 Application program interfaces (system calls)
4.1.5.1 Process management
4.1.5.2 Signaling
4.1.5.3 File management
4.1.5.4 Directory management
4.1.5.5 Protection
4.1.5.6 Time management
4.1.6 Operating system structuring
4.1.6.1 Monolithic systems
4.1.6.2 Layered systems
4.1.6.3 Virtual machines
4.1.6.4 Client-server computing
4.2 Concurrency 10.0
4.2.1 Methods for achieving concurrency
4.2.1.1 Processes
4.2.1.2 Threads
4.2.2 Task communication and synchronization
4.2.2.1 Race conditions
4.2.2.2 Critical sections
4.2.2.3 Busy waiting mutual exclusion
4.2.2.4 Sleep and wakeup
4.2.2.5 Semaphores and mutexes
4.2.2.6 Monitors
4.2.2.7 Message passing
4.2.3 Classic task communication problems
4.2.3.1 Dining philosophers
4.2.3.2 Producer/consumer
4.2.3.3 Readers and writers
4.2.4 Process and thread scheduling
4.2.4.1 Process states

- 4.2.4.2 Long term scheduling
  - 4.2.4.2.1 First come first served
  - 4.2.4.2.2 Priority
  - 4.2.4.2.3 Shortest job first
- 4.2.4.3 Short term scheduling
  - 4.2.4.3.1 Round robin
  - 4.2.4.3.2 Priority
  - 4.2.4.3.3 Multiple queues
- 4.2.4.4 Real time scheduling

4.3 Input/Output 7.0

- 4.3.1 I/O hardware principles
  - 4.3.1.1 Devices
  - 4.3.1.2 Controllers
  - 4.3.1.3 Direct memory access
- 4.3.2 I/O software principles
  - 4.3.2.1 Interrupt handlers
  - 4.3.2.2 Device drivers
  - 4.3.2.3 Kernel functions
  - 4.3.2.4 Application models (traditional, event driven)
- 4.3.3 Deadlocks
  - 4.3.3.1 Specific examples
  - 4.3.3.2 Deadlock theory
  - 4.3.3.3 Dealing with deadlock
    - 4.3.3.3.1 Ignore
    - 4.3.3.3.2 Detect and recover
    - 4.3.3.3.3 Prevention
    - 4.3.3.3.4 Avoidance
- 4.3.4 Specific device classes
  - 4.3.4.1 Disk drives
    - 4.3.4.1.1 Hardware
    - 4.3.4.1.2 Software
  - 4.3.4.2 Clocks and timers
    - 4.3.4.2.1 Hardware
    - 4.3.4.2.2 Software
    - 4.3.4.2.3 Implementing multiple timers
  - 4.3.4.3 Terminals
    - 4.3.4.3.1 Hardware
    - 4.3.4.3.2 Software

4.4 Memory management 10.0

- 4.4.1 Early systems: monoprogramming
- 4.4.2 Multiprogramming with fixed partitions
- 4.4.3 Multiprogramming with variable partitions
- 4.4.4 Managing available memory
  - 4.4.4.1 Bit maps
  - 4.4.4.2 Linked lists
- 4.4.5 Swapping

4.4.6 Virtual memory
- 4.4.6.1 Paging
- 4.4.6.2 Page tables
- 4.4.6.3 Associative memory
- 4.4.6.4 Inverted page tables

4.4.7 Page replacement algorithms
- 4.4.7.1 OPT
- 4.4.7.2 FIFO
- 4.4.7.3 LRU
- 4.4.7.4 LRU approximations
- 4.4.7.5 Clock

4.4.8 Optimization
- 4.4.8.1 The working set model
- 4.4.8.2 Optimizing applications for paged virtual memory
- 4.4.8.3 Local and global allocation policies
- 4.4.8.4 Page size issues

4.4.9 File mapping to virtual memory
- 4.4.9.1 Principles
- 4.4.9.2 Unification of I/O and virtual memory
- 4.4.9.3 Segmentation
- 4.4.9.4 Paging and segmentation

4.5 File systems 8.0

4.5.1 File characteristics
- 4.5.1.1 Naming
- 4.5.1.2 Structure
- 4.5.1.3 Attributes
- 4.5.1.4 Operations

4.5.2 Directory structures
- 4.5.2.1 Flat file systems
- 4.5.2.2 Hierarchical file systems
- 4.5.2.3 Path names
- 4.5.2.4 Directory operations

4.5.3 File implementation

4.5.4 Directory implementation

4.5.5 Disk space management

4.6 System security 4.0

4.6.1 Security concepts

4.6.2 Physical security

4.6.3 Types of security attacks

4.6.4 User authentication

4.6.5 Protection mechanisms
- 4.6.5.1 Protection domains
- 4.6.5.2 Access control lists
- 4.6.5.3 Capabilities

5.0 Teaching Methodology

5.1 Methods to be used

The course is presented primarily through lectures, with ample opportunity for student discussion of the topics presented. The lectures will be accompanied by illustrations of components of contemporary operating systems.

5.2 Student role in the course

Students will be expected to complete selected programming assignments that illustrate various operating system algorithms and various APIs in a contemporary operating system (e.g. UNIX or Windows NT). Examples include process scheduling, memory management, complete system simulation, and solution of process synchronization problems. A “toy operating system” (e.g. MINIX) may also be used as the basis for the programming assignments, and can directly connect them with the lectures.

6.0 Evaluation

6.1 Types of student projects

Student projects may include written homework problems (similar to those found as exercises in common textbooks), and will always include multiple programming assignments. One programming assignment will require the use of the C programming language and extensive use of the API (system calls) for a UNIX/Linux system (to the exclusion of the standard C library for input/output and process management), including demonstration of the student’s understanding of the APIs for input/output, concurrency and processes. This assignment will often take the form on an extension to a simple command line interpreter (shell). In some offerings and for some assignments, students may be permitted to work in small groups to complete the assignments. It may be possible for some assignments (especially those involving simulation of OS algorithms) to be implemented in a language like C++, Java, or Python. Multiple quizzes (each covering a major division of the topics) and a comprehensive final examination will be given. It is typical for the students to be required to produce a formal report on the final programming assignment/project. Depending on the class size, these reports may be presented orally.

6.2 Basis for determining the final grade

The final grade will be determined from a weighted average of the programming assignments (about 50 percent), periodic quizzes (30 percent) and the final examination (20 percent).

6.3 Grading type

Letter grades will be determined using the weighted average of the various items used to evaluate students. A typical grade mapping is illustrated below.

Points	Grade
97 – 100%	A+
93 – 96%	A
90 – 92%	A-
87 – 89%	B+
83 – 86%	B
80 – 82%	B-
77 – 79%	C+
73 – 76%	C
70 – 72%	C-
67 – 69%	D+
63 – 66%	D
60 – 62%	D-
0 – 59%	F

7.0 Resource Material

7.1 Textbooks and/or other required readings used in the course

Tanenbaum and Woodhull, *Operating Systems: Design and Implementation* (3rd edition), Prentice-Hall (2006)

7.2 Other suggested reading material

Additional materials will be identified for students detailing the API for the selected operating system. In some cases this may be an additional manual, but it may also include instructor-prepared material or on-line reference material (e.g. “man pages”).

Additional documents on the operating system used for the programming assignments, C programming in that environment, and concurrency will also be supplied.

This material will typically be provided as HTML or PDF documents on the class web page.

7.3 Other sources of information

ACM SIGOPS (Special Interest Group on Operating Systems)

7.4 Current bibliography of resources for student’s information

Anderson and Dahlin, *Operating Systems: Principles and Practice* (2nd edition), Recursive Books (2014)
Kifer and Smolka, *Introduction to Operating System Design and Implementation: The OSP 2 Approach*, Springer (2007)

Silberschatz et al, *Operating System Concepts* (9th edition), John Wiley (2012)
Stallings, *Operating Systems: Internals and Design Principles* (8th edition), Pearson (2014)
Tanenbaum, *Modern Operating Systems* (4th edition), Pearson (2014)

8.0 Other Information

8.1 Accommodations statement:
8.2 Other:
8.3 Author(s): Stanley Wileman

9.0 Computer Science Accreditation Board (CSAB) Category Content (class time in hours)

CSAB Category	Core	Advanced
Data structures	5	
Computer organization and architecture	2	4
Algorithms and software design	6	28
Concepts of programming languages		

10.0 Oral and Written Communication

Every student is required to submit at least __1_____ written reports (not including exams, tests, quizzes, or commented programs) to typically 10_____ pages and to make _1_____ oral presentations of typically _____15____ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

11.0 Social and Ethical Issues

Operating system security and its significance in the unwanted disclosure or modification of information is treated in this course. About 4 hours are used to present this material.

12.0 Theoretical content

		Contact hours
12.1	Concurrent processes	2.0
12.2	Mutual exclusion	4.0
12.3	Deadlock analysis	2.0
12.4	Process scheduling	2.0
12.5	Disk head scheduling	1.0
12.6	Virtual memory	3.0
12.7	Disk block placement analysis	0.5
12.8	Disk space block size analysis	0.5

13.0 Problem analysis

The class identifies the problems an operating system must address, and demonstrates the various approaches that have been used to address those problems in real operating systems.

14.0 **Solution design**

Solutions are designed to three classes of problems. The first class includes implementation of the solution for a real-world problem using only the API for a real operating system for input/output and process management; the goal is to emphasize the fundamental nature of that API. A common problem of this class is the implementation of a simple command line interpreter, or extension of a given primitive command line interpreter by adding support for concurrent execution of commands or piping the standard output of one command to the standard input of another. The second class of problems requires concurrency and process synchronization (e.g. sleeping barber, dining philosophers). The third category includes actual operating system problems; characteristic problems in this class include process and disk scheduling (often addressed through simple simulations) and memory management.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
01/10/2002	Initial ABET version	Wileman	
06/14/2003	Cleanup	Wileman	
10/23/2008	Revision – Changed the textbook and reviewed the entire thing. Added the Mapping table.	Mahoney	There’s likely a newer version than 2003, but this is the one I located on the (old) CS web site and worked from; so there’s a gap in the dates but now this one is the “up to date” version.
6/16/2015	Revised the syllabus format; updated evaluation to match current practice; updated catalog description and learning objectives to better match ACM/IEEE CS Curricula 2013; updated resource material list.	Wileman	
6/26/2015	Numbered list of performance objectives	Wileman	