CSCI 340, Operating Systems
Spring 2016

Description

The course will introduce operating systems principles with an emphasis on multiprogramming systems. Among the concept areas covered are real and virtual storage management, processor management, process synchronization and communication, IO management, and file management. Lectures three hours per week.

Prerequisites

Completion of CSCI 230 and CSCI 250 with a C- or better, MATH 207

Instructor

Anthony Leclerc, Ph.D.
Harbor Walk East, Room 324
Office Phone: 953-5963

Office Hours

9:30 a.m. – 11:30 a.m. MWF
other times by appointment

Classroom

HWEA 300/HWEA 334

Required Texts

Operating Systems: Internals and Design Principles, 8th Edition, William Stallings, Prentice-Hall, 2015.

Optional Linux/C Texts

Linux Pocket Guide, 2nd Edition, Daniel Barrett, O'Reilly Media, 2012.

Running Linux, Fifth Edition, Matthas Dalheimer and Matt Welsh, O'Reilly Media, 2005.
Programming in C, Third Edition, Stephen Kochan, Sams Publishing, 2004.
C for Java Programmers, Tomasz Muldner, Addison-Wesley, 2000.
The C Programming Language, Second Edition, Kernighan & Ritchie, Prentice-Hall, 1988.

Learning Outcomes

1. Understand the function, basic principles, and desirable characteristics of an operating system. [Chapter 2]
2. Understand the historical milestones, advances, and terms in OS evolution. [Chapter 2]
3. Learn and apply the fundamentals of the Unix OS (architecture, file system, editor, basic utilities and shell commands) [Handouts, supplemental reading, and web tutorials]
4. Learn and apply the 'C' programming language for systems development on Unix environments. Includes *gcc*, makefiles, text editors, and *gdb* [Handouts, supplemental reading, and web tutorials]
5. Understand and explain OS architectures including monolithic, layered, virtual machine, client-server, kernel, multiprocessor [Chapter 2]
6. Understand the relationship between an OS and hardware. This includes: processor, memory, disks, I/O devices, and buses. [Chapters 1 and 2]

7. Understand the concept of a *process* (vs *thread*), as well as key elements of process description and control, and states. [Chapters 3 and 4]

8. Understand the principles of *concurrency* and *synchronization* mechanisms (e.g., *semaphores*, *monitors*, *mutexes*, message passing, *condition variables*) [Chapter 5]

9. Understand the principles of *deadlocks* and prerequisite conditions (e.g., *mutual exclusion*, hold-and-wait, no preemption, circular wait). This also includes avoidance, prevention, detection, and recovery mechanisms. [Chapter 6]

10. Understand the principles of memory management, including memory partitioning, fitting algorithms, *swapping*, *paging* and replacement algorithms, segmentation, *virtual memory*, working sets, Belady's anaomaly, and thrashing. [Chapters 7 and 8]

11. Understand types of uniprocessor *scheduling* (preemptive and non-preemptive algorithms). [Chapter 9]

12. Understand the basic principles of *file management* and I/O device management. [Chapters 11 and 12]

Grading Procedure and Scale

Midterm Exam	20%
8 Programming Assignments	40%
Final Project (Team of 2)	10%
Comprehensive Final	30%

Percentage	Grade
93-100	A
90-92	A-
87-89	B+
83-86	B
80-82	B-
77-79	C+
73-76	C
70-72	C-
65-69	D
0-64	F

Development Environment

'C' is a very portable language and some students will choose to develop one or more programming assignments in a programming environment such as Microsoft Visual Studio, Project Builder, XCode, or .NET. I discourage you from doing so. Your code *must* compile and run on a Linux system. You are responsible for verifying that your code ultimately compiles and runs correctly in a Linux environment. Moreover, you will be required to utilize the GNU 'C' compiler and Makefiles.

One difficult aspect of programming is debugging. In this class I will show you how to use the GNU debugger (gdb). This debugger is available on UNIX systems and will allow you to step through your program source line at a time. It will also allow you to determine where your program is crashing.

Course Policies

- **Portable electronic device policy:** Before entering class, turn off cellphones, pagers, and other electronic devices.
- **Attendance policy:** You are expected to attend every class. Attendance is critical to your success in this course. While most information will be available either on-line or in the textbook, some information may only be presented during class discussion. Should you miss *6 or more class sessions*, I reserve the right to withdraw you from the class for excessive absences.
- **Submission policy:** An assignment must be submitted electronically using *OAKS* by the due date specified for the assignment. Late assignments will not be accepted by OAKS. Thus, make sure to submit any work you have done on an assignment by the due date.
- **Makeup policy:** *No* makeup tests will be given. If a student presents a written excuse from the *Absence Memo Office* for a missed test, then the following test (possibly the final exam) score will count additionally for this missed exam. A score of zero will be recorded for any other missed test.
- **Collaboration and Plagiarism:** On all assignments, you are expected to *do your own work!*

  If you discuss an assignment with someone other than your instructor, make sure that you are not taking notes, typing, writing on the board, or recording the discussion in any way other than "in your head". As long as you walk away from the discussion with "nothing in hand" and then continue to design and develop your own solution, I have no problem with this level of collaboration.

  On the other hand, if your "discussion" leads to code or an outline or sketch for the solution (pseudo-code, flowchart, etc.) being draw-up (by hand, typed, or electronically manipulated or transferred) then you have crossed the line into academic dishonesty. If in doubt, ask me.
- **E-mail communication:** The best way to contact me "off hours" is via e-mail (leclerca@cofc.edu). Please expect a reasonable 1-day turnaround time for any e-mail inquiries (2-days if sent just prior to or on the weekend).

  It is *very difficult* to debug a program with you via e-mail: turnaround times are slow and interaction possibilities are limited. For this reason see me "face-to-face" for help debugging of your programs.
- **Special Needs** Any student who, because of a disability, may require special arrangements in order to meet course requirements should contact me as soon as possible to make necessary arrangements. The instructor will require a *Professor Notification Letter* (PNL).

Tentative Weekly Course Schedule

Week1: Course introduction, Chapter 1, and Chapter 2 [Course outcomes 1, 2]

Week2: Chapter 2 (Short week MLK Holiday) [Course outcomes 5, 6]

Week3: Introduction to 'C' and development environment [Course outcomes 3, 4]

Week4: More 'C' and program development [Course outcomes 3, 4]

Week5: Chapters 3 [Course outcome 7]

Week6: Chapter 4 [Course outcome 7]

Week7: Chapter 5 [Course outcomes 8, 9]

Week8: Midterm Exam

Week9: Spring Break

Week10: Chapter 6 [Course outcomes 8, 9]

Week11: Chapter 7 [Course outcome 10]

Week12: Chapter 8 [Course outcome 10]

Week13: Chapter 9 [Course outcome 11]

Week14: Chapter 11 [Course outcome 12]

Week15: Chapter 12 [Course outcome 12]