Description

The course will introduce operating systems principles including device management, process management, memory management, virtual memory, file management, and protection mechanisms. Programming exercises (using Linux) will allow students to gain hands-on experience with the details of how these operating systems principles are implemented.

Prerequisites

Completion of CSCI 230, CSCI 250, and MATH 207

Instructor

Anthony Leclerc, Ph.D.
J. C. Long Building Room 208
Office Phone: 953-5963

Office Hours

8:30 a.m. – 10:00 p.m. MWF
11:00 a.m. – 12:00 p.m. MWF
other times by appointment

Classroom

JCLONG 219

Required Texts


Other Linux/C Texts


Learning Objectives

By the end of this course, the student should be able to:

- demonstrate a basic understanding of the 'C' programming language including pointers and proper memory management
- demonstrate a basic understanding of the UNIX development environment including editing, compiling, debugging, command-line basics, and Makefiles
- demonstrate overview-knowledge of a computer system (ch. 1)
- demonstrate overview-knowledge of an operating system (OS) including OS objectives, functions, evolution, major achievements, virtual machines, and multicore/multiprocessor design (ch. 2)
- demonstrate an understanding of a process, including process states, representation, and control (ch. 3)
• demonstrate an understanding of a thread, including types of threads and multicore and multithreading (ch. 4)

• demonstrate an understanding of concurrency including mutual exclusion and synchronization (ch. 5)

• demonstrate an understanding of deadlock and starvation (ch. 6)

• demonstrate an understanding of memory management including partitioning, paging, and segmentation (ch. 7)

• demonstrate an understanding of virtual memory (ch. 8)

• demonstrate an understanding of uniprocessor scheduling (ch. 9)

• demonstrate an understanding of I/O management and disk scheduling (ch. 11)

• demonstrate an understanding of file management (ch. 12)

• time permitting, understand OS characteristics on embedded systems(ch. 13)

Grading Procedure and Scale

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>93-100</td>
<td>A</td>
</tr>
<tr>
<td>90-92</td>
<td>A-</td>
</tr>
<tr>
<td>87-89</td>
<td>B+</td>
</tr>
<tr>
<td>83-86</td>
<td>B</td>
</tr>
<tr>
<td>80-82</td>
<td>B-</td>
</tr>
<tr>
<td>77-79</td>
<td>C+</td>
</tr>
<tr>
<td>73-76</td>
<td>C</td>
</tr>
<tr>
<td>70-72</td>
<td>C-</td>
</tr>
<tr>
<td>65-69</td>
<td>D</td>
</tr>
<tr>
<td>0-64</td>
<td>F</td>
</tr>
</tbody>
</table>

Midterm 20%
Assignments 50%
Final 30%

Development Environment

'C' is a very portable language. However, I have seen instances where a particular development environment allowed or inserted non-standard statements within a program source file. Some students will choose to develop programming assignments in a programming environment such as Microsoft Visual Studio, Borland, or .NET. I discourage you from doing so. Your code must compile and run on our Linux server, stono. You are responsible for verifying that your code ultimately compiles and runs correctly on stono.

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

- **Noisy 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. You are responsible for all information and announcements (including test times, assignment due dates, etc.) given in class.

- **Submission policy**: An assignment must be submitted electronically using the submit executable program available on the department server, stono.

- **Lateness policy**: An assignment must be submitted prior to the beginning of class on the due date scheduled by the instructor. The submit executable will automatically reject any submissions made after the due date and time.

- **No compilation policy**: No credit will be given for a programming assignment which does not compile. Verify that your submission compiles on stono before you submit.

- **Makeup policy**: No makeup tests or quizzes will be given. If a student presents a written excuse from the Undergraduate Dean’s Office for a missed test (or quiz), then the following test (possibly the final exam) score will count additionally for this missed exam (or quiz, respectively). A score of zero will be recorded for any other missed test or quiz.

- **Plagiarism policy**: On all assignments, you are expected to do your own work!

  If you are discussing the 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 policy**: 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 important to note that it is usually very difficult for me to help debug a program with you via e-mail: turnaround times are slow and interaction possibilities are limited. For this reason I encourage you to see me “face-to-face” regarding 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).