Introduction to Operating Systems I

Instructor:
D. Kevin McGrath
Office: KEC 2109
e-mail: dmcgrath@eecs.oregonstate.edu

Office Hours:
Time: Check home page
Location: KEC 2109
Appointments are also encouraged, and can be made via email.

Schedule:
Lecture MWF 12:00 - 12:50, WNGR 116

Text:

Course Description

Introduction to operating systems using UNIX as the case study. System calls and utilities, fundamentals of processes and interprocess communication.

Enforced Prerequisites:

CS 261, (CE 271 or CS 271), knowledge of C

You will use Unix and C extensively throughout this course, yet basic Unix and C will not be taught in this course. If you have not used Unix or C before, you will be expected to learn the basics on your own time. The beginning of your textbook is all about UNIX/Linux commands. There are many Unix/Linux and C tutorials available on the web.

The specific variant of UNIX that you will use in this class is Linux. If you are not sure where to start looking for tutorials on the web, I recommend using a search engine. You are also expected to have familiarity with common data structures, such as stacks, heaps, queues, linked lists, and hash tables, which are covered by the prerequisite class CS261 and its predecessors.

Academic Honesty:

See University and college policies.

Students are expected to do their own work / groups are expected to be the sole source of their code. We use software designed to find similarities between programs. Each group’s program is compared to every other group’s program to find similarities. Please do your own work.

Programming assignments present unique challenges for graders. It is often difficult for a grader to distinguish between legitimate help and plagiarism. Therefore, it is sometimes possible to get a
good score without really understanding what you have handed in.

Honesty is absolutely essential in order for learning to take place. It will form the foundation of your professional integrity in your career.

If you are having trouble with an assignment, you are encouraged to discuss it with other students, TAs, the instructor, or anyone else who will listen, but don’t just have someone else tell you how to solve the problem! If other students ask you for help, don’t just let them copy your work! It is possible to discuss problems without plagiarizing. One of the best methods of debugging is to explain your solution to someone else.

If you get help from, give help to, or work together with someone, you must (in the program header block) list that person as a collaborator and describe the help. Programs that are very similar will be subjected to review unless both programs indicate that they were produced collaboratively. We use plagiarism-detection software check your code against the code from other students. It is quite sophisticated and can easily see through variable name changes and formatting differences.

If you get help from printed or online sources, you must cite your references.

If you are found in violation of any of the above policies, whether you are the giver or receiver of help, you will receive a zero on the assignment or fail the course (Instructor’s discretion). The academic dishonesty charge will be documented and sent to your school’s dean and the Office of Student Conduct. The first offense results in a warning; the second offense results in an academic dishonesty charge on your transcript, a disciplinary hearing, and possible expulsion.

The bottom line is: Each student is expected to understand all aspects of the programs s/he submits for credit.

**Learning Objectives**

- Explain why multiprogramming is important for modern operating systems.
- Explain the general structure of a multiprogrammed operating system.
- Explain the purpose and operation of system calls.
- Write a program utilizing system calls.
- Write a program using a scripting language.
- Write a program that uses regular expressions to parse input data.
- Write a program that spawns processes and provides mutual exclusion for variables or other resources shared by the processes.
- Write a program that uses sockets to implement a client/server system.
- Explain how a common file system works, including structure, I/O operations, and security.
- Describe the memory organization of a typical process in a common operating system.

**Grading**

- Programming projects: 50%
• Midterms: 20%
• Quizzes: 10%
• Final exam: 20%

Grades are recorded as their GPA equivalent. As such, a 95 is recorded as 4, an 81 is recorded as a B, etc. Pluses and minuses are not used in this course. Therefore, while extra credit is available, it can only help offset missed points on the current assignment.

Exams

There are 3 exams offered during the term. Only the two highest will count. The final midterm is offered the Friday of dead week. As it is technically an optional exam, this does not violate any policies.
If you miss an exam, it will simply be counted as your dropped exam.

Projects

All project must be submitted electronically by 11:59pm on the due date at the engr submission site.
They will be accepted late with a 5% penalty per weekday and / or 5% per weekend, UP TO ONE WEEK LATE. If you submit a late assignment, you must notify the TAs separately via email or else he/she will not know to look for your assignment. You may not submit the last project late.

You have the right to make use of 2 grace days, used in increments of 1 day. These allow you to have an unpenalized late assignment up to 2 days (weekends still count as a single day for this). These must be invoked prior to the submission deadline, and must be invoked via email. Failure to follow this procedure renders the late day invalid, though consumed.

All grading will be done on Linux on os-class. We highly recommend that you do your development on a Linux operating system or at least test your assignments on os-class.engr.oregonstate.edu, a Linux machine provided specifically for use by CS311. You can log onto it via SSH. Standard policies apply: you must have a campus IP to directly log in, or log in via flip or name.
Compilation will be on the os-class server. Assignments which do not compile will receive a grade of 0. It does not matter if your code compiles with icc, suncc, msvc, on your computer, on your best friend’s uncle’s neighbor’s dog’s computer, or anywhere else. If it doesn’t compile on os-class, it doesn’t compile.

All work must be done individually unless specifically allowed to work in groups.

Final Grade Assignment

There will be no curve given in this course. The grading scale is as follows:
There will be opportunities on nearly every assignment for extra credit. Do not ask for extra credit beyond what is offered in class, as it will not be given.

Your 3 exams scores must have a weighted average ≥ 1.0 in order to pass the course. Only the 2 highest midterm grades will be counted.

All programming projects must be submitted in order to pass the course. Students missing assignments at the end of the term will automatically receive an F grade.

Students who do not submit the final exam will automatically receive an F grade.