Instructor: Anish Mathuria, Faculty Block 1, Room 1105
Lectures: Tuesday, Friday 11-11:50; Thursday, 9-9:50, LT-2
Labs: As per lab timetable

Course Abstract
The course provides knowledge about writing software that makes use of the programming abstractions supported by modern operating systems. Topics covered include: exception control flow, processes, signals, virtual memory, linking, dynamic memory allocation, system-level I/O, inter-process communication, threads and network programming.

The course is based on C, Linux, and the Intel IA32 architecture. It is modeled after a very successful course developed by Randy Bryant and David O’Hallaron at CMU.

Textbooks

Grading
- 3 Tests: 60%. In-sem exams 1 and 2 are worth 12% and 18%, respectively. The end-sem exam is worth 30%. The end-sem will be cumulative.
- Labs: 20%. Lab check-offs count 12% and the remaining 8% is for attendance.
- Viva-voce: 8%. Oral evaluation to be conducted in lab before end-sem.
- Projects: 12%. Two programming projects to be turned in.

Course Outcomes
Upon completion of this course, students should be able to:
- Write programs that create and handle multiple processes or threads.
- Write programs that use signal handlers to catch signals.
- Write programs that implement facilities provided by command shells, e.g. I/O redirection and pipes.
- Write makefiles for modular compilation of programs with several source files.
- Write programs that use sockets for network communication.
- Explain how a compiler and linker put together a program.
- Explain the potential problems arising from the concurrent execution of multiple processes or threads.