IT628: Systems Programming, Winter 2025-26
Course handout

Instructor: Anish Mathuria
Lectures: CEP 207, Monday 10-10:50, Wednesday 10-10:50, Friday 8-8:50
Practicals: TBA
Prerequisites: programming in C

Course abstract

All operating systems provide services for application programs they run. The service points through which programs request services are called system calls. Both system calls and library functions exist to provide services to application programs. A library function may invoke one or more system calls and in many cases perform the same action as a system call. However, it is important to realize that the library functions may be replaced if desired, whereas the system calls usually cannot be replaced. This course provides an introduction to the system calls for performing essentials tasks like managing processes, files, memory and network communication.

Topics to be covered

- Process management: programs and processes, creating processes, waiting for processes, executing new programs, zombies, Unix shells.
- Signals: Sending signals, signal handlers, signal blocking, signal sets, job-control signals, non-local jumps.
- I/O management: file abstraction, file descriptors, file access, file permissions, reading directories standard I/O: buffering, implementation using low-level I/O, I/O redirection.
- Inter-process communication: pipes, non-blocking reads and writes, shared memory.
- Threads: differences between processes and threads, creating threads, terminating threads, reaping terminated threads, detaching threads.
- Concurrency: race conditions, locking, semaphores, synchronization with semaphores, producer-consumer problem.
- Sockets: types of connection, addressing, creation of transport end point, sending and receiving messages.

Learning outcomes

At the end of the course, students will

- Be familiar with the design and implementation of Unix shells
- Be familiar with writing programs that create, manage and terminate processes and threads on UNIX.
- Be familiar with the relationship between standard I/O and low-level I/O.
- Be familiar with the problems that may arise because of concurrent execution of multiple processes or threads.
- Be familiar with using sockets to implement interprocess communication.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
	X	X		X				X			

Textbook

UNIX System Programming (Second Edition), Keith Haviland, Dina Gray and Ben Salama, 1999. The course text covers most of the lecture material.

References

- Advanced Programming in the UNIX Environment, W. Richard Stevens, 2nd edition, 2002, Addison-Wesley.
- Computer Systems: A Programmer's Perspective, Randal E. Bryant and David O'Hallaron, 3rd edition, 2016, Pearson Education.
- The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie, 2nd edition, 1988, Prentice-Hall.

Evaluation (tentative)

Two in-semester tests and a final exam will take place at the officially scheduled exam time for the course.

- 35% mid-term exams
- 35% final exam
- 20% in-lab exercises
- 10% homework

Lab policy

- Lab attendance is mandatory. You will have to demonstrate your code for the in-lab exercises to a TA as specified in the lab handout. If you fail to do so, you will lose points. There will be no makeup labs for any reason.

- It is a course requirement that you do the programming assignments on Linux OS. If the code you submit does not compile on Linux or we are unable to run the executable, you will not receive credit.