
MWF 11:00–11:50 a.m., ENGR A309C (CS Conference Room)

See the course web page for more information and resources: http://www.cs.siu.edu/~cs407

Professor
Dr. Norman Carver, Faner 3121, phone 453-6048, email carver@cs.siu.edu.
Office hours: MWF 1:00–3:00pm (or by appointment).

Required Textbook

Other Textbooks

Course Description
This course builds on the knowledge gained in CS 306, to prepare students to do advanced development on Linux/UNIX platforms. The topics studied are critical for achieving high performance in large-scale, high-load networked software systems. These topics include development techniques such as profiling, concurrent programming and synchronization, network programming for high-load servers, advanced I/O alternatives, and IPC such as shared memory. The course will involve the study of code from Open Source projects like Apache and Nginx. The focus will be on the C language, but other languages will also be considered. Students must complete a significant network software project.

Objectives of the Course
- Advancing students C development skills.
- Improving students knowledge of concurrent programming.
- Improving students knowledge of network and distributed programming.
- Familiarizing students with advanced Linux/UNIX system calls.
- Familiarizing students with performance and security trade-offs in software.
- Preparing students for advanced software engineering jobs (e.g., Site Reliability Engineering at Google).

Prerequisites
- Undergraduates: CS 306 and CS 335 with grades of C or better. (CS335 prereq. is being waived this semester)
- Graduates: C language and Linux/UNIX system programming experience.
Overview of Course Topics

1. Advanced C Development
   • Compilers: GCC vs. Clang
   • C vs. C++ vs. Objective C
   • compiler options (optimization, etc.)
   • code disassembly and analysis
   • debugging from core files
   • performance profiling
   • library creation and use

2. Concurrent Programming
   • issues in concurrent programming
   • process vs. threads comparison
   • Pthreads calls and usage
   • thread synchronization: mutexes, condition variables
   • process synchronization: semaphores, signals
   • process/thread pools
   • thread-safe and async-signal-safe functions
   • event-based (event-driven) programming

3. Signals
   • signal characteristics in detail
   • signal usage patterns
   • writing proper signal handlers
   • async-signal-safe functions
   • real-time signals
   • signals vs. file descriptors (e.g., signalfd())

4. Advanced Network Programming
   • TCP vs. UDP servers and clients
   • alternative server models
   • the SCTP protocol
   • UNIX sockets
   • raw sockets
   • distributed programming and RPC

5. Advanced I/O
   • non-blocking I/O
   • scatter/gather I/O
   • interleaved I/O (poll() and select())
• epoll API (Linux-specific) and UNIX alternatives
• signal-based I/O
• async I/O (AIO)
• sendfile() and splice(), and equivalents
• issues in handling large numbers of devices/clients
• understanding kernel internals

6. Advanced IPC
• message queues
• shared memory
• memory mapped files
• understanding kernel internals

7. Devices
• terminals and terminal I/O
• pseudo terminals and pty
• drivers

8. Writing Secure Programs
• security considerations in C
• program privileges
• Linux capabilities and UNIX alternatives