

This course provides in-depth training for software developers on UNIX system programming facilities. With an emphasis on writing portable programs using industry standards such as POSIX, X/Open, and the SUS, programming interfaces to several system services are explained in detail. Students will write and modify many C programs in this class, using system calls and library routines. This course can also be delivered on Linux.

Course Objectives:

- Develop the programming skills required to write applications that run on the UNIX operating system.
- Write portable applications using UNIX standards.
- Develop the basic skills required to write network programs using the Berkeley Sockets interface to the TCP/IP protocols.

Audience: Application developers who will be writing advanced programs on UNIX.

Prerequisites: *Fundamentals of UNIX and C Programming. Strong C programming skills are required for this course.*

Number of Days: 4 days

- | | |
|--|--|
| <p>1. Course Introduction
 Course Objectives
 Overview
 Suggested References</p> | <p>link(), unlink(), remove(), and rename() Functions
 Functions to Create, Remove, and Read Directories</p> |
| <p>2. UNIX Standards
 Brief History of UNIX
 AT&T and Berkeley UNIX Systems
 Major Vendors
 What is a Standard?
 What is POSIX?
 Other Industry Specs and Standards
 Library vs. System-Level Functions</p> | <p>4. System I/O
 Standard I/O vs system I/O
 System I/O Calls
 File and Record Locking</p> |
| <p>3. Files and Directories
 Basic File Types
 File Descriptors
 The open() and creat() Functions
 Keeping Track of Open Files
 File Table Entries
 The v-node Structure
 The fcntl() Function
 The fcntl() Function – with F_DUPFD Command
 File Attributes
 The access() Function</p> | <p>5. Processes
 What is a Process?
 Process Creation and Termination
 Process Memory Layout
 Dynamic Memory Allocation
 Accessing Environment Variables
 Real and Effective User IDs</p> |
| | <p>6. Process Management
 The Difference Between Programs and Processes
 The fork() System Function
 Parent and Child
 The exec System Functions
 Current Image and New Image</p> |

- The wait() Functions
- The waitpid() Function
- Interpreter files and exec
- 7. Basic Interprocess Communication:**
 - Pipes**
 - Interprocess Communication
 - Pipes
 - FIFOs
- 8. Signals**
 - What is a Signal?
 - Types of Signals
 - Signal Actions
 - Blocking Signals from Delivery
 - The sigaction() function
 - Signal Sets and Operations
 - Sending a Signal to Another Process
 - Blocking Signals with sigprocmask()
 - Scheduling and Waiting for Signals
 - Restarting System Calls (SVR4)
 - Signals and Reentrancy
- 9. Introduction to Pthreads**
 - Processes and Threads
 - Creating Threads
 - Multitasking
 - Overview of Thread Architectures
 - Processes Versus Threads
 - The Pthreads API
 - Thread Termination
 - Joining Threads
 - Detaching Threads
 - Passing Arguments to Threads
- 10. Pthreads Synchronization**
 - The Sharing Problem
 - Mutexes
 - Creating and Initializing Mutexes
 - Using Mutexes
 - Additional Synchronization Requirement
 - Using Condition Variables
- 11. Overview of Client/Server Programming with Berkeley Sockets**
 - Designing Applications for a Distributed Environment
 - Clients and Servers
 - Ports and Services
- Connectionless vs. Connection-Oriented Servers
- Stateless vs. Stateful Servers
- Concurrency Issues
- 12. The Berkeley Sockets API**
 - Berkeley Sockets
 - Data Structures of the Sockets API
 - Socket System Calls
 - Socket Utility Functions
- 13. TCP Client Design**
 - Algorithms instead of Details
 - Client Architecture
 - Generic Client/Server Model – TCP
 - The TCP Client Algorithm
- 14. TCP Server Design**
 - General Concepts
 - Iterative Servers
 - Concurrent Servers
 - Performance Consideration
 - An Iterative Server Design
 - A Concurrent Server Design
- 15. System V Interprocess Communication**
 - System V IPC
 - Elements Common to msg, shm, and sem Facilities
 - The Three System V IPC Facilities
 - IPC via Message Queues
 - IPC via Shared Memory
 - Coordinating the Use of Shared Memory Segments
 - Semaphore Sets - semget()
 - Semaphore Sets – semctl()
 - Semaphore Sets – the semop() call
 - Shared Memory Coordination
 - Using Semaphores
 - Commands for IPC Facility
 - Handling - ipcs and ipcrm
- 16. Appendix A – Date and Time Functions**
 - Overview

Time Representations
Decoding Calendar Time
Shorthand Functions – asctime() and
ctime()
Formatting Date and Time Strings
Process Times
The Difference Between clock() and
times()
Berkeley High Resolution Timer

17. **Appendix B – Standard I/O**
Standard I/O Calls to manipulate streams
Standard I/O Calls which perform
character I/O
Standard I/O Calls which perform string
I/O
Standard I/O Calls Which Perform
Formatted I/O
Standard I/O Calls Which Perform
Binary I/O