Linux/UNIX System Programming Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Overview

This intensive five-day course provides developers with a comprehensive understanding of Linux and UNIX system programming. Participants learn how operating systems work internally and how to build reliable, efficient, and high-performance applications using low-level system interfaces.

The course combines theory with practical exercises covering file systems, processes, signals, multithreading, inter-process communication (IPC), networking, sockets, event-driven programming, and shared memory. Although the training focuses on Linux, it also explains POSIX-compliant programming concepts that apply across many UNIX platforms.

Prerequisites

Participants should have:

A good understanding of the C programming language

Programming experience in C, C++, Go, Rust, D, Python, or a similar language

Basic Linux/UNIX command-line knowledge

Previous system programming experience is helpful but not required.

Who Should Attend

This course is intended for:

Linux and UNIX application developers

System software programmers

Network application developers

Embedded Linux developers

Security engineers

Site Reliability Engineers (SREs)

DevOps engineers

Software engineers building high-performance backend services

Outline

Linux/UNIX System Programming Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

1. Course Introduction

Course objectives

Development environment

Linux programming fundamentals

Overview of system programming

2. Fundamental Concepts

Learn the core concepts behind Linux system programming, including:

System calls

Standard library functions

Error handling techniques

System data types

Reading and understanding example code

3. File Input and Output

Understand how Linux applications interact with files through system calls.

Topics include:

File descriptors

Opening and closing files

Reading and writing data

Basic file I/O operations

4. File I/O Buffering

Explore how buffering affects application performance.

Topics include:

Kernel buffering

Standard I/O buffering

Buffer management

Performance considerations

5. Advanced File I/O

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

Learn advanced file handling techniques.

Topics include:

File offsets

Random access using seek operations

Atomic file operations

File descriptor duplication

File status flags

Advanced file I/O interfaces

6. Working with Files

Understand how Linux stores and manages files.

Topics include:

Inodes

File metadata

File permissions

File modes

umask

Modifying file attributes

7. Directories and Links (Optional)

Learn how Linux organizes files.

Topics include:

Directory structure

Hard links

Symbolic links

Working directories

Directory-relative operations

Directory scanning

8. Processes

Understand process management in Linux.

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

Topics include:

Process IDs

Process memory layout

Command-line arguments

Environment variables

User credentials

Process groups

Sessions

The /proc filesystem

9. Signals

Learn how processes communicate through signals.

Topics include:

Signal fundamentals

Signal handling

Signal masks

Pending signals

Writing reliable signal handlers

10. Advanced Signal Handling

Explore advanced signal management.

Topics include:

Async-signal-safe functions

Interrupted system calls

SA_SIGINFO handlers

Signal trampolines

11. Process Creation and Termination

Learn how Linux creates and manages processes.

Topics include:

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

fork()

Process termination

Parent and child processes

Zombie processes

Orphan processes

SIGCHLD

PID file descriptors

12. Executing Programs

Understand how new programs are launched.

Topics include:

execve()

exec() family

Process replacement

File descriptors during execution

Process attributes

13. System Call Tracing (Optional)

Debug Linux applications using strace.

Topics include:

Monitoring system calls

Tracing child processes

Filtering trace output

14. Multithreading Fundamentals

Learn POSIX thread programming.

Topics include:

Thread lifecycle

Thread creation

Thread termination

Thread IDs

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

Joining and detaching threads

Thread attributes

Signals in multithreaded programs

15. Thread Synchronization

Prevent race conditions and synchronize shared resources.

Topics include:

Critical sections

Mutexes

Condition variables

Synchronization techniques

Dynamic synchronization objects

16. Inter-Process Communication (Optional)

Introduction to IPC mechanisms.

Topics include:

IPC overview

Choosing the appropriate IPC method

17. Pipes and FIFOs

Exchange data between processes.

Topics include:

Anonymous pipes

Named pipes (FIFOs)

Building process pipelines

18. Socket Programming Fundamentals

Build network applications using sockets.

Topics include:

Socket types

Address families

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

Socket creation

Binding

Listening

Accepting connections

Stream sockets

Datagram sockets

19. Internet Domain Sockets

Develop TCP/IP applications.

Topics include:

IPv4 and IPv6 sockets

Address structures

Loopback interfaces

Wildcard addresses

Client/server examples

20. Address Resolution

Convert hostnames and service names.

Topics include:

Host resolution

Port numbers

getaddrinfo()

Modern socket programming techniques

21. Advanced Socket Programming

Improve network application performance.

Topics include:

Socket options

shutdown()

TCP TIME-WAIT

SO_REUSEADDR

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

Performance tuning

22. UNIX Domain Sockets

Communicate efficiently between local processes.

Topics include:

Stream sockets

Datagram sockets

Local IPC techniques

23. Alternative I/O Models

Develop scalable event-driven applications.

Topics include:

Non-blocking I/O

Signal-driven I/O

poll()

Event loops

24. epoll Programming

Build highly scalable Linux servers.

Topics include:

epoll architecture

epoll_create()

epoll_ctl()

epoll_wait()

Event notifications

Edge-triggered mode

Performance optimization

25. POSIX Semaphores

Coordinate concurrent processes and threads.

Topics include:

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

Named semaphores

Unnamed semaphores

Semaphore operations

26. POSIX Shared Memory

Share memory between processes for maximum performance.

Topics include:

Shared memory objects

mmap()

Memory mapping

Shared memory synchronization

High-performance IPC