



K Computing

Creating Applications For Linux (2 day version of Linux Dev. Fundamentals)

Description

This course is designed to bring C developers up to speed with a variety of tools and capabilities of Linux. This includes development and debugging tools as well as system and library functions.

The intent is to provide background that will be of general interest to all Linux based developers.

Students should be prepared for a some lengthy programming exercises.

Overview

This course provides practice with key tools and capabilities available to developers of Linux based applications and system software. The course shows attendees how to use development and debugging tools and how to make use of many Linux system calls and library routines.

Attendees will spend approximately 50 percent of the class time actually gaining hands-on experience with these topics.

Course Objectives

After this course attendees will be able to ...

- Effectively use a variety of tools for Linux application development
- Demonstrate how to use a number of Linux system calls and library functions
- Compile programs with a variety of options
- Use GDB to debug applications
- Use electric fence, gprof, gcov, and other tools for debugging and performance analysis
- Write a simple shell
- Use a variety of file system related system calls such as mmap().
- Create socket based applications

Attendees will learn:

- How to use GNU tools for compiling and debugging.
- How to use various tools for debugging and performance measurement

- How to use system calls for such things as inter-process communication and interacting with the file system,.

Who Should Attend:

The course is for programmers who are new to Linux. Attendees should have experience with C and be able to perform Linux commands.

Duration

Two days

Course Materials

The workshop materials include a comprehensive student workbook. The workbook contains all of the slides used in the course as well as hands-on lab exercises.

Students should bring a personal USB thumb drive to use to bring home class files.

Course Workshop and Set-up:

The workshop makes use of laptops with a desktop Linux distribution for development.

Students will share a computer with two students per computer. Students work as a team on the laboratory exercises.

Course Outline

1. Linux Development

- Objectives and format
- Course overview

2. Overview Of Linux Programming

- Linux kernel overview
- System calls and library routines

Lab Exercises

- Use perror()
- Examine and use errno.
- Experiment with different errors from open()
- Trace errors within library functions

3. Programs In Pieces

- System Include files
- Using and creating libraries
- Linking

- Using Make

Lab Exercises

- Create a new dynamic shared library
- Switch the libraries an executable uses via LD_LIBRARY_PATH
- Create a Makefile for your library

4. Programming with GNU tools

- gcc
- optimization
- linker
- debugging with gdb

Lab Exercises

- Compile programs with different optimization options
- Examine generated code
- Experiment with dead code removal
- Use pre-processor macro features
- Debug a program after a core dump with GDB
- Step through a running program and change variable values on the fly

5. Tools

- eclipse
- electric fence
- valgrind
- gprof
- gcov
- patch

Lab Exercises

- Try eclipse
- Use gprof to profile a program for performance
- Use Electric Fence to look for dynamic memory allocation bugs
- Use valgrind to look for dynamically allocated memory bugs
- Use cachegrind to examine cache utilization
- Use dmalloc to test for memory leaks
- Create and use patches
- Use source code tools like ctags

6. Process Management

- Creating processes
- Process signaling and status
- Process and user ID's

Lab Exercises

- Use the variety of exec() calls
- Write a simple shell that forks and execs commands

7. Linux File System

- Access Permissions
- I/O System Calls

- Manipulating files
- Higher performance I/O

Lab Exercises

- Open/Read/Write with files
- Implement “tail -f” like functionality
- Use lseek() and mmap()
- Do non-blocking I/O with devices
- Use select() and poll()

8. Inter-process Communication

- Pipes
- Semaphores
- Message Queues
- Sockets
- Shared Memory

Lab Exercises

- Use pipe()
- Use named pipes
- Create a socket based server
- Create a xinetd based service