System programming for embedded Linux

The performance and stability of a system ultimately depends on how well you use the resources of the underlying operating system. This class explains how to make effective use of the facilities of the POSIX compliant Linux GNU C library when implementing embedded devices. Attendees will begin with the basics of file handling, device I/O and memory and process management. They will learn how to use signals safely, how to compartmentalize an application into multiple processes and threads and the trade-offs between various types of Inter Process Communication using sockets, message queues and shared memory

Duration

4 days

Audience

This course is designed for engineers and system architects working on embedded devices

Prerequisites

Essential: good knowledge of the C programming language, since this is used in the programming portions of the course

Desirable: *either* a good background in embedded devices, *or* a reasonable proficiency in Linux command-line tools. Delegates with neither will find the learning curve rather steep

Course materials

All students will receive:

- A printed copy of the presentations and lab notes
- A USB flash drive containing worked solutions to the problems, plus electronic copies of the course materials
- A free copy of the trainer's book, “Mastering Embedded Linux Programming”

About the trainer

Chris Simmonds has been using Linux in embedded systems for over 15 years. He has been running training courses and workshops in embedded Linux since 2002 and has delivered hundreds of sessions to many well-known companies in the UK, Europe, USA, South America and SE Asia. He is the author of the book “Mastering Embedded Linux Programming”, and is a frequent presenter at open source and embedded conferences, including Embedded Linux Conference and Embedded World. You can see some of his work on the "Inner Penguin" blog at www.2net.co.uk

Enquiries and bookings

Please email training@2net.co.uk or call +44 (0)7788 130719

System programming for embedded Linux

Course outline

Developing for embedded Linux

- The tool-chain: choosing, installing and testing
- Application program interfaces: the POSIX standard
- Open source licenses: GPL/LGPL, BSD, etc

Debugging

- Debugging a remote target device using gdb and gdbserver

Files and devices

- Files and file-related API: waiting for several things to happen with select() and poll()
- Devices: everything is a file. Interfacing with a simple device driver. Using the ioctl() function to access device-specific operations

Processes

- Process life cycle: fork(), exit() and exec()
- Scheduling: real-time and non-real-time policies; setting priority and niceness

Memory

- Virtual memory and its consequences
- Allocating from the heap and stack
- Mapping memory using mmap

Signals

- Standard and real-time signals
- Writing robust signal handlers
- Signal masks and how to handle signals synchronously

Real-time Linux kernels

- Kernel preemption
- The real-time PREEMPT_RT patch

Inter-process communication

- Pipes
- Shared memory and semaphores
- Message queues
- Sockets: internet and UNIX (local). Stream and datagram connections

POSIX Threads

- Thread life cycle: pthread_create(), pthread_exit(), pthread_join()
- Scheduling threads: real-time and non-real-time
- The thread stack and how to set the stack size

Thread synchronisation

- Synchronisation using mutexes; priority inversion and priority inheritance
- Condition variables: producer and consumer threads
- Thread cancellation and clean-up operatorsh

Time and timers

- Timer accuracy: high-resolution timers, POSIX clocks and timers
- Measuring time
- Periodic task