Linux Course
at
Intel's Israel Development Center
December 24, 25, 31 2006 and January 2, 2007

Ascender Technologies Ltd.

April 7, 2008

Abstract

This document is a proposal for a Linux kernel level course. This course is to be largely original material based on a syllabus provided by Intel with further suggestions from Ascender Technologies Ltd. The course will be delivered over a period of four days. Each day will consist of two half day (2 and 1/2 hour) sessions.

Ascender Technologies Ltd.
POB 71,
Rehovot 76100
Tel: 08-9491586
Email: joel@ascender.com
Mobile: 050-5975146
Fax: 08-9452645

Contents

1 Course Content

1.1 The Course Syllabus

- Linux Design
  - Memory Spaces
  - Name Spaces
  - Processes
  - Sockets (IPC)
- Examples
  - Initrd
  - /proc filesystem
  - Dynamic linking
  - Light weight processes
- Memory management
  - How paging is implemented in Linux (including page fault algorithm).
  - Main data structures
  - Zones
  - Bootmem allocator
  - High memory
  - Memory mapped files
  - Swapping
  - Handing out of memory conditions
- Interrupt handling
- Bottom halves
- Locking mechanisms
  - Overview of available locking primitives
  - Big kernel lock
  - Locking between kernel processes, bottom halves and ISRs
- Process management

  - Main data structures
  - Process creation and termination
  - Threads VS. processes
  - Kernel threads
  - Signal management
  - Process layout in memory (ELF, code/stack/data/bss)

- Scheduling
  - Policy
  - Thread priority
  - Preemption
  - Real time threads

- Multi processor support
  - Interrupt distribution
  - Processor affinity
  - CPU initialization
  - Process migration

- VFS and block IO level
  - Overview of VFS
  - Main data structures
  - Page and buffer cache
  - IO schedulers

1.2 Special Topics

In addition special topics, about once a day, will be introduced both to add interesting relevant material and to give a break to the very technical material presented. My current suggestions are:

- **Build an actual embedded system**: This illustrates many kernel design issues.
- **Device Drivers**: A series of device drivers that exercise the functionality of the parallel port. This is not to provide a cookbook, "You too can write Linux device drivers", but rather illustrate general kernel techniques, dynamic code loading, locking, interrupts etc.
- **Analyze an embedded Linux product**: Crack open a commercial embedded router and see how Linux is used to leverage Open Source Software in a commercial environment and what kernel issues arise.

- **The politics of Linux**: The idea is to give developers some idea of the unique issues that effect development of software in a GPL environment. GPL's legal requirements sometimes constrain technical solutions in a commercial environment. An awareness of these issues is important even to the most commercially sheltered kernel developer working in industry.