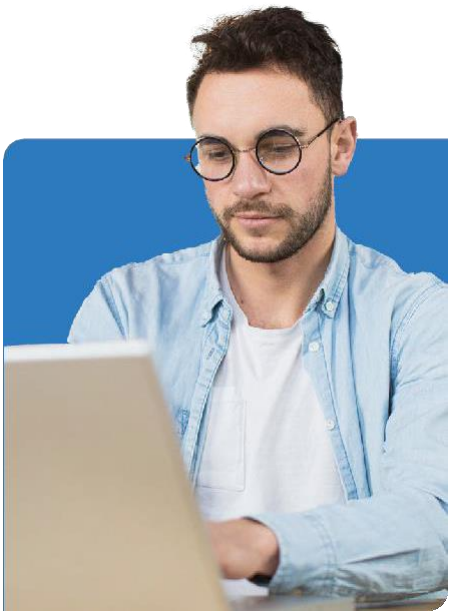
ThriveDX™

Linux Kernel

TDXAISD-102



ThriveDX Linux Kernel

Time Commitment
5 days (total of 40 hours / 8 hours per day)

Skill Level
Professional Level

Course Category
AI | Software Development

This Linux kernel is now the most widely used operating system on the planet and many development teams need to know how to add device support to it, understand it's performance characteristics, tune it, compile it, reduce it's size, provide board support for it, develop platforms based on it, understand it's real time support and much more.

Prerequisites

No kernel prior knowledge is required but knowledge of Linux user space API and programming tools is required.

Objectives

This course is all about the Linux kernel but is focused on writing device drivers for the kernel as this is the first typical thing one learns when becoming a kernel developer.

Program Structure

Linux Kernel Introduction Module 01

- System and kernel overview
- Kernel code specifics
- Kernel subsystems
- History and versioning scheme
- Understanding the development process
- Specific legal issues
- Kernel user interface

Kernel Sources Module 02

- Getting the sources
- Using the patch command
- Structure of source files
- Kernel source code browsers

Compiling Module 03

- Kernel configuration
- Useful settings for embedded systems
- Compiling
- Generated files
- Make commands for configuring, compiling or installing a kernel

Booting Module 04

- Linux system booting overview
- The boot-loader’s job
- Review of Linux boot-loaders
- U-boot details
- Linux kernel booting
- Advantages of initramfs over initrd
- Booting parameters
- NFS boot example
- System startup

Cross-compiling Module 05

- Kernel cross-compiling setup
- Ready-made configuration files for specific architectures & boards
- Cross-compiling

Basic Driver Development Module 06

- Linux device drivers
- A simple module
- Programming constraints
- Loading, unloading modules
- Module parameters and dependencies
- Adding sources to the kernel tree

Linux Memory Management Module 07

- Linux memory management
- Physical and virtual (kernel and user) address spaces
- Linux memory management implementation
- Allocating with kmalloc, by pages and with vmalloc

I/O Memory Module 08

- I/O register and memory range registration
- I/O register and memory access
- Read / write memory barriers

Character Drivers Module 09

- Device numbers
- Getting free device numbers
- File operations
- Character driver registration

Kernel Debugging Module 10

- Using printk, /proc or /sys
- Debugfs
- Using an ioctl interface, gdb and kgdb

Processes and scheduling Module 11

- Process life
- Timer frequency
- Priorities and timeslices
- Sleeping and waking up API

Interrupts Module 12

- Waiting for the availability of resources
- Interrupt handler registration
- Scheduling deferred work

Concurrency management Module 13

- Managing concurrent access to resources: mutexes, spinlocks
- Atomic operations

Advice and resources Module 14

- Getting help and contributions
- Bug report and patch submission to Linux developers
- References: websites, books & international conferences

Kernel boot-up details Module 14

- Detailed description of the kernel boot-up process, from execution by the boot-loader to the execution of the first userspace program
- Initcalls: how to register your own initialization routines?

Introduction to BSP development Module 15

- Board Support Packages (BSP)
- Porting U-boot and the Linux kernel
- Creating board dependent code
- Studying code for an ARM board

Introduction to power management Module 16

- Supporting frequency scaling
- CPU and board specific power management
- Power management in device drivers
- Control from user space

- Saving power in the idle loop
- Studying power management implementations in the Linux kernel

Introduction to Linux Real-Time Programming Module 17

- Understanding the sources of latency in standard Linux
- Soft real-time solutions for Linux: improvements in Linux 2.6
- Use the latest RT preempt patches for mainstream Linux
- Real-time kernel debugging
- Measuring and analyzing latency
- Hard real-time solutions for Linux
- RTLinux issues, the RTAI and Xenomai projects
- Comparing with RT preempt patches
- Real-time offerings from commercial Linux vendors: MontaVista, TimeSys, Wind River, LynuxWorks etc.

C library and cross-compiling tool-chain Module 18

- Choosing the target C library
- Ready to use cross-compiling tool-chains
- Building a cross-compiling tool-chain with automated tools
- Installing cross-compiled libraries in the root filesystem

Embedded system development tools Module 19

- Commercial toolsets and distributions
- Community toolsets (focus on Buildroot & Scratchbox)
- How to find existing Free Software for a particular need

BusyBox Module 20

- Detailed Overview and features
- Configuration, compiling and deploying
- Saving space by implementing your own applets

Lightweight tools for embedded systems Module 21

- HTTP and ssh servers
- Graphical toolkits

- Web browsers
- Text editors
- Precompiled packages and distributions

Choosing file-systems Module 22

- File-systems for block devices
- Usefulness of journaled filesystems
- Read-only block file-systems
- RAM file-systems
- File-systems for Memory Technology Devices (MTD)
- Suggestions for embedded systems

Udev and hot-plugging Module 23

- Handling hardware events from userspace
- Creating and removing device files
- Identifying drivers, notifying programs and users