سرفصل آموزشی

پک پایه متخصص اینترنت اشیا

فهرست سرفصل‌های دوره‌های آموزشی

سرفصل‌های دوره آموزشی

Embedded Linux

Level-1

Introduction to embedded Linux

- Advantages of Linux versus traditional embedded operating systems
- Typical hardware platforms used to run embedded Linux systems
- Overall architecture of embedded Linux systems
- Overview of the major software components
- Development environment for Embedded Linux development

Cross-compiling toolchain and C library

- What is inside a cross-compiling toolchain?
- Choosing the target C library
- What is inside the C LIBRARY?
- Ready to use cross-compiling toolchains
- Building a cross-compiling toolchain with automated tools
- Getting and configuring Crosstool-NG
- Executing it to build a custom cross compilation toolchain
- Exploring the contents of the toolchain

Boot process, firmware, and bootloaders

- Booting process of embedded platforms, focus on the x86 and ARM architectures
- Boot process and bootloaders on x86 platforms (legacy and UEFI)
- Boot process on ARM platforms: ROM code, bootloaders, ARM Trusted Firmware
- Focus on U-Boot: configuration, installation, and usage.
- U-Boot commands, U-Boot environment, U-Boot scripts, U-Boot generic distro boot mechanism

Bootloader and U-boot

- Set up serial communication with the board.
- Configure, compile and install U-Boot for the target hardware
- Configure, compile and install Trusted Firmware-A
- Become familiar with U-Boot environment and commands
- Set up TFTP communication with the board. Use TFTP U-Boot commands

Linux kernel

- Role and general architecture of the Linux kernel
- Separation between kernel and user-space, and interfaces between user-space and the Linux kernel
- Understanding Linux kernel versions choosing between vendor-provided kernel and upstream kernel, Long Term Support versions
- Getting the Linux kernel source code
- Fetching Linux kernel sources
- Clone the mainline Linux tree
- Accessing stable releases

Configuring, compiling and booting the Linux kernel

- Configuring the Linux kernel: ready-made configuration files, configuration interfaces
- Concept of Device Tree
- Cross-compiling the Linux kernel
- Study of the generated files and their role
- Installing and booting the Linux kernel
- The Linux kernel command line
- Configuring the Linux kernel and cross-compiling it for the embedded hardware
- Downloading your kernel on the board through U-boot's TFTP client.
- Booting your kernel.
- Automating the kernel boot process with UBoot

Root filesystem in Linux

- Filesystems in Linux
- Role and organization of the root filesystem
- Location of the root filesystem: on storage, in memory, from the network
- Device files, virtual filesystems
- Contents of a typical root filesystem
- Detailed overview. Detailed features
- Configuration, compiling and deploying

Tiny root file system built from scratch with BusyBox

- Setting up a kernel to boot your system on a workstation directory exported by NFS
- Passing kernel command line parameters to boot on NFS
- Creating the full root filesystem from scratch. Populating it with Busy-

Box based utilities.
- System startup using BusyBox init
- Using the BusyBox HTTP server.
- Controlling the target from a web browser on the PC host.

Block filesystems

- Accessing and partitioning block devices
- Filesystems for block devices
- Usefulness of journaled filesystems
- Read-only block filesystems
- RAM filesystems
- How to create each of these filesystems
- Suggestions for embedded systems
- Creating partitions on your SD card
- Booting a system with a mix of filesystems: SquashFS for the root filesystem, ext4 for system data, and tmpfs for temporary system files

Flash filesystems

- The Memory Technology Devices (MTD) filesystem
- Filesystems for MTD storage: JFFS2, Yaffs2, UBIFS
- Kernel configuration options
- MTD storage partitions
- Focus on today's best solution, UBI and UBIFS: preparing, flashing and using UBI images

سرفصل‌های دوره آموزشی

Embedded Linux

Level-2

Root filesystem in Linux

- Filesystems in Linux
- Role and organization of the root filesystem
- Location of the root filesystem: on storage, in memory & from the network
- Device files, virtual filesystems
- Contents of a typical root filesystem
- Overview & features
- Configuration, compiling and deploying

Tiny root filesystem built from scratch with BusyBox

- Setting up a kernel to boot your system on a workstation directory exported by NFS
- Passing kernel command line parameters to boot on NFS
- Creating the full root filesystem from scratch. Populating it with BusyBox based utilities.
- System startup using BusyBox init
- Using the BusyBox HTTP server
- Controlling the target from a web browser on the PC host
- Setting up shared libraries on the target and compiling a sample executable

Accessing hardware devices

- How to access hardware on popular busses: USB, SPI, I2C, PCI
- Usage of kernel drivers and direct userspace access
- The Device Tree syntax, and how to use it to describe additional devic-es
- Finding Linux kernel drivers for specific hardware devices
- Using kernel modules
- Hardware access using /dev and /sys
- User-space interfaces for the most common hardware devices: storage, network, GPIO, LEDs, audio, graphics, video
- Exploring the contents of /dev and /sys and the devices available on the embedded hardware platform
- Using GPIOs and LEDs
- Modifying the Device Tree to control pin multiplexing and to declare an I2Cconnected joystick
- Adding support for a USB audio card using
- Linux kernel modules
- Adding support for the I2C-connected joystick through an out-of-tree module

Cross-compiling user-space libraries and applications

- Configuring, cross-compiling and installing applications and libraries
- Concept of build system, and overview of a few common build systems used by open-source projects: Makefile, autotools, CMake, meson
- Overview of the common issues encountered when cross-compiling Manual cross-compilation of several opensource libraries and applica-tions for an embedded platform
- Learning about common pitfalls and issues, and their solutions
- This includes compiling alsa-utils package, and using its speaker-test program to test that audio works on the target

Embedded system building tools

- Approaches for building embedded Linux systems: build systems and binary distributions
- Principle of build systems, overview of Yocto Project/OpenEmbedded and Buildroot.
- Principle of binary distributions and useful tools, focus on Debian/ Ubuntu
- Specialized software frameworks/ distributions: Tizen, AGL, Android
- Using Buildroot to rebuild the same basic system plus a sound playing server (MPD) and a client to control it (mpc)
- Driving music playback, directly from the target, and then remotely through an MPD client on the host machine
- Analyzing dependencies between packages

Open source licenses and compliance

- Presentation of the most important open-source licenses: GPL, LGPL, MIT, BSD, Apache and etc.
- Concept of copyleft licenses
- Differences between (L) GPL version 2 and 3
- Compliance with open-source licenses: best practices

Overview of major embedded Linux software stacks

- Systemd as an init system
- Hardware management with udev
- Inter-process communication with D-Bus
- The connectivity software stack: Ethernet, WiFi, modems, Bluetooth
- The graphics software stack: DRM/KMS, X.org, Wayland, Qt, Gtk, OpenGL
- The multimedia software stack: Video4Linux, GStreamer, Pulseaudio, Pipewire

Integration of additional software stacks

- Integration of systemd as an init system
- Use udev built in systemd for automatic module loading

Application development and debugging

- Programming languages and libraries available.
- Build system for your application, an overview of CMake and meson
- The gdb debugger: remote debugging with gdbserver, post-mortem debugging with core files
- Performance analysis, tracing and profiling tools, memory checkers: strace, ltrace, perf, valgrind
- Creating an application that uses an I2Cconnected joystick to control an audio player
- Setting up an IDE to develop and remotely debug an application
- Using strace, ltrace, gdbserver and perf to debug/investigate buggy applications on the embedded board

Useful resources

- Books about embedded Linux and system programming
- Useful online resources
- International conferences

سرفصل‌های دوره آموزشی

C programming

& App Development

Part one - Essential C programming

Introduction

C Fundamentals

Formatted I/O

Expression & Statement

Loops

Basic Types

Arrays

Functions

- Programming Process
- Setting Up
- Specification
- Code Design

Program Design

Low-Level Programming

The Standard C library

- Input/output
- Numbers and Characters
- Error Handling
- C99

Part Three – Linux Programming

Memory Management and Allocation

File I/O

Time

Processes

Pipes and Fifo’s

Signals

POSIX Threads

Writing Secure Privileged Program

Inter Process Communication

Mutex

Networking and Sockets

Sockets – Fundamentals of TCP/IP networks

Part Four – Secure C Programming

Introduction

Secure working with strings

Secure pointers

Dynamic memory management

Integer Security

Formatted Output

Concurrency

File I/O

Recommended Practice

- The security development life-cycle
- Design
- Implementation
- Verification