Embedded
Labworks

LINUX DEVICE DRIVERS

Agenda

Embedded Labworks

VISÃO GERAL

Título	**Linux device drivers**
Tópicos Principais	Introdução ao kernel Linux, configurando e compilando o kernel, desenvolvendo um módulo do kernel, desenvolvimento de um driver de dispositivo de caractere para acessar um GPIO do kit de desenvolvimento, gerenciamento de memória, frameworks do kernel, usando a gpiolib, integração com o driver model do kernel, trabalhando com o device tree, gerenciamento de processos, trabalhando com interrupções, desenvolvimento de um driver de botão, mecanismos de sincronização, trabalhando com um driver de expansora de GPIO no barramento SPI, depuração em kernel space, contribuindo com o desenvolvimento do kernel, livros e referências para estudos.
Duração	3 dias – 24 horas. 30% de teoria e 70% de prática.
Instrutor	Sergio Prado
Público-alvo	Estudantes, engenheiros, desenvolvedores e líderes de equipes de desenvolvimento de software para sistemas embarcados.
Pré-requisitos	Os participantes devem estar familiarizados com a interface de linha de comandos de sistemas Linux e ter conhecimentos intermediários em linguagem C. Para um melhor aproveitamento, é aconselhável ter conhecimentos em desenvolvimento de sistemas com Linux embarcado.
Equipamentos	Todos os equipamentos necessários serão fornecidos pela Embedded Labworks.
Material de estudo	O material de estudo é composto pelos slides das apresentações, o livro de atividades e exercícios, os guias de referência e documentos de estudo adicionais. Todo o material será fornecido de forma eletrônica no início do treinamento.
Kit de desenvolvimento	Durante as seções abertas é utilizado o módulo Colibri i.MX6 (ARM Cortex-A9) e a placa-base Viola da Toradex. Dependendo da turma, ou das necessidades dos participantes, pode-se utilizar outro kit de desenvolvimento. Possíveis opções são: Wandboard, Beagleboard-xM, Beaglebone Black, FriendlyARM mini2440, Raspberry Pi, etc.

Embedded Labworks

DIA 1

Apresentação: Introdução ao kernel Linux
Histórico, principais características, arquitetura, fontes e versionamento, licença.

Laboratório: Estudando o código-fonte do Linux
Nesta atividade iremos estudar o código-fonte do kernel Linux.

Apresentação: Kit de desenvolvimento
Características do hardware, documentação.

Laboratório: Conectando e testando o hardware
Nesta atividade iremos conectar o target ao host pela porta serial e configurar uma aplicação de terminal para acessar a porta serial do target.

Apresentação: Configurando e compilando o kernel Linux
Configurando e habilitando funcionalidades no kernel, compilando o kernel, compilando módulos, compilando o device tree, instalando o kernel, linha de comandos do kernel.

Laboratório: Compilando o kernel e testando o sistema
Nesta atividade aprenderemos a configurar e compilar o kernel, e testaremos o boot do sistema.

Apresentação: Módulos do kernel
Introdução aos módulos do kernel, como carregar e descarregar um módulo, como passar parâmetros para um módulo, desenvolvendo um módulo, integrando o módulo nos fontes do kernel.

Laboratório: O primeiro módulo do kernel
Nesta atividade iremos aprender a desenvolver e trabalhar com módulos do kernel.

Apresentação: Dispositivos de hardware

Classes de dispositivos, arquivos de dispositivo, como funciona o acesso ao hardware no Linux, implementando um dispositivo de caractere, registrando o device number, implementando as operações em arquivo, trocando dados com user space, registrando um dispositivo de caractere.

Laboratório: O primeiro driver

Nesta atividade começaremos o desenvolvimento de um driver de dispositivo de caractere para acionar um led conectado a um GPIO do kit de desenvolvimento.

Apresentação: Gerenciamento de memória

O subsistema de memória no Linux, MMU, a organização da memória, endereçamento físico x endereçamento virtual, zonas de memória, alocação de memória, page allocator, SLAB allocator, vmalloc e kmalloc.

Apresentação: Hardware I/O

Acessando I/O no Linux, port I/O, memory mapped I/O, requisitando uma região de I/O, mapeando um endereço físico para um endereço virtual, acessando I/O.

Laboratório: Acessando o hardware

Nesta atividade iremos implementar o acesso físico ao hardware no driver de led que desenvolvemos no exercício anterior.

DIA 2

Apresentação: Frameworks

Frameworks do kernel, framebuffer, a camada input, framework de leds, outros frameworks comuns no Linux (TTY, ALSA, V4L, MTD, hwmon, IIO, watchdog, RTC, PWM, pinctrl, GPIO, etc).

Laboratório: Integrando com o framework de led
Nesta atividade iremos integrar o driver desenvolvido na atividade anterior com o framework de leds do kernel.

Apresentação: Gpiolib
Acessando GPIOs no kernel, a API da camada gpiolib.

Laboratório: Usando a camada de GPIO do kernel
Nesta atividade iremos alterar o driver desenvolvido na atividade anterior para utilizar a camada de GPIO (gpiolib) do kernel.

Apresentação: Bus Infrastructure
Arquitetura e vantagens do device model, infraestrutura de barramento, barramento I2C, analisando o driver de um dispositivo I2C, platform devices.

Laboratório: Integração com o driver model do kernel
Nesta atividade iremos integrar o driver desenvolvido no exercício anterior ao driver model do kernel.

Apresentação: Device Tree
Histórico e introdução ao device tree, analisando um device tree, device tree bindings, adaptando um driver para trabalhar com o device tree, gerando e passando o DTB para o kernel.

Laboratório: Trabalhando com device tree
Neste exercício iremos adaptar o driver desenvolvido na atividade anterior para trabalhar com o mecanismo de device tree.

Embedded Labworks

DIA 3

Apresentação: Gerenciamento de processos

Processos e threads no Linux, os estados de uma tarefa, escalonador, tarefas de tempo real, contexto de execução, kernel threads, trabalhando com eventos, sleeping, como usar wait queues.

Laboratório: Kernel threads e wait queues

Nesta atividade estudaremos kernel threads e wait queues através do desenvolvimento de um driver de dispositivo de caractere que irá exportar para o usuário os eventos de um botão conectado no kit de desenvolvimento.

Apresentação: Gerenciamento de interrupções

Tratamento de interrupção no Linux, registrando uma ISR, restrições no tratamento de interrupções, como implementar uma ISR, dividindo o tratamento de uma ISR em top half e bottom half, softirqs, tasklets, work queues.

Laboratório: Trabalhando com interrupções

Neste exercício iremos alterar o driver de botão desenvolvido na atividade anterior para trabalhar com interrupções e workqueues.

Apresentação: Mecanismos de sincronização

Multithread e concorrência, operações atômicas, locking, semáforos, mutex, spinlocks, desabilitando preempção, desabilitando interrupções, deadlock, r/w spinlocks, completion variable.

Laboratório: Usando spinlocks

Nesta atividade estudaremos o uso de mutex como mecanismo de locking do kernel.

Apresentação: Kernel Debugging

Debugging com mensagens, níveis de log, debugfs, magic sysrq key, kernel oops, gdb, kdb, kgdb, jtag, tracing, kernel probes, systemtap, kernel tracepoints, LTTng, profiling, oprofile,

Laboratório: Usando ferramentas de debugging
Nesta atividade estudaremos alguns mecanismos de depuração do kernel Linux.

Laboratório: Analisando a implementação de drivers
Nesta atividade iremos analisar a implementação de alguns drivers disponíveis no código-fonte do kernel.

Apresentação: E agora?
O processo de desenvolvimento do Linux, documentação e filosofia da comunidade de software livre, contribuindo com o desenvolvimento do kernel, reportando problemas, links e livros para continuar os estudos, dúvidas e comentários finais.