Linux Device and Driver Model

Wednesday, August 24, 2016

5:54 AM

Below are the references:

Kernel Driver (Free Electrons)

<http://www.bravegnu.org/device-model/device-model.html> (Save as pdf at Driver/DeviceModel)

Free Electrons (Device drivers)

Linux-kernel-slides.pdf

LDD3 book

<https://lwn.net/Articles/31185/>

<https://lwn.net/Articles/645810/>

<http://www.bravegnu.org/device-model/device-model.html>

<http://www.linux-mag.com/id/2617/>

<http://linuxkernel51.blogspot.in/2011/02/linux-driver-model.html>

<http://www.linuxjournal.com/article/6717>

Device model is what I can say is similar to device manager in windows. It give you the view of how the system is put together and how the devices are connected. In Linux 2.4, there was no unified database of what devices were present in the system and how they were connected. Since lInux runs on wide variety of architecture, it need to maximize the reusability of the code between platforms

For example, we want the same USB device driver to be

usable on a x86 PC, or an ARM platform, even though the

USB controllers used on these platforms are different.

▶ This requires a clean organization of the code, with the device

drivers separated from the controller drivers, the hardware

description separated from the drivers themselves, etc.

▶ This is what the Linux kernel Device Model allows, in

addition to other advantages covered in this section

In Linux, a driver is always

interfacing with:

▶ a framework that allows the

driver to expose the

hardware features in a

generic way.

▶ a bus infrastructure, part

of the device model, to

detect/communicate with

the hardware.

Machine generated alternative text:
App ication 
User space 
System Ca 
Interface 
Framework 
Kerne 
Driver 
Bus infrastructure 
Hardware 

▶ The device model is organized around three main data

structures:

▶ The struct bus\_type structure, which represent one type of

bus (USB, PCI, I2C, etc.)

▶ The struct device\_driver structure, which represents one

driver capable of handling certain devices on a certain bus.

▶ The struct device structure, which represents one device

connected to a bus

▶ The kernel uses inheritance to create more specialized versions

of struct device\_driver and struct device for each bus

subsystem.

**Bus drivers**

▶ The first component of the device model is the bus driver

▶ One bus driver for each type of bus: USB, PCI, SPI, MMC,

I2C, etc.

▶ It is responsible for

▶ Registering the bus type (struct bus\_type)

▶ Allowing the registration of adapter drivers (USB controllers,

I2C adapters, etc.), able to detect the connected devices, and

providing a communication mechanism with the devices

▶ Allowing the registration of device drivers (USB devices, I2C

devices, PCI devices, etc.), managing the devices

▶ Matching the device drivers against the devices detected by

the adapter drivers.

▶ Provides an API to both adapter drivers and device drivers

▶ Defining driver and device specific structures, mainly

struct usb\_driver and struct usb\_interface

Machine generated alternative text:
USB Adapter 
driver A 
USB Core 
Registers the bus_type structure 
USB Device 
driver 2 
DEV4 
USB Device 
driver 3 
DEV5 
USB Adapter 
driver B 
USBI 
System 
USB2 
USB Device 
driver 1 
DEVI 
DEV3 
DEV2 