libusb
USB Application Programming Interface

Derald D. Woods

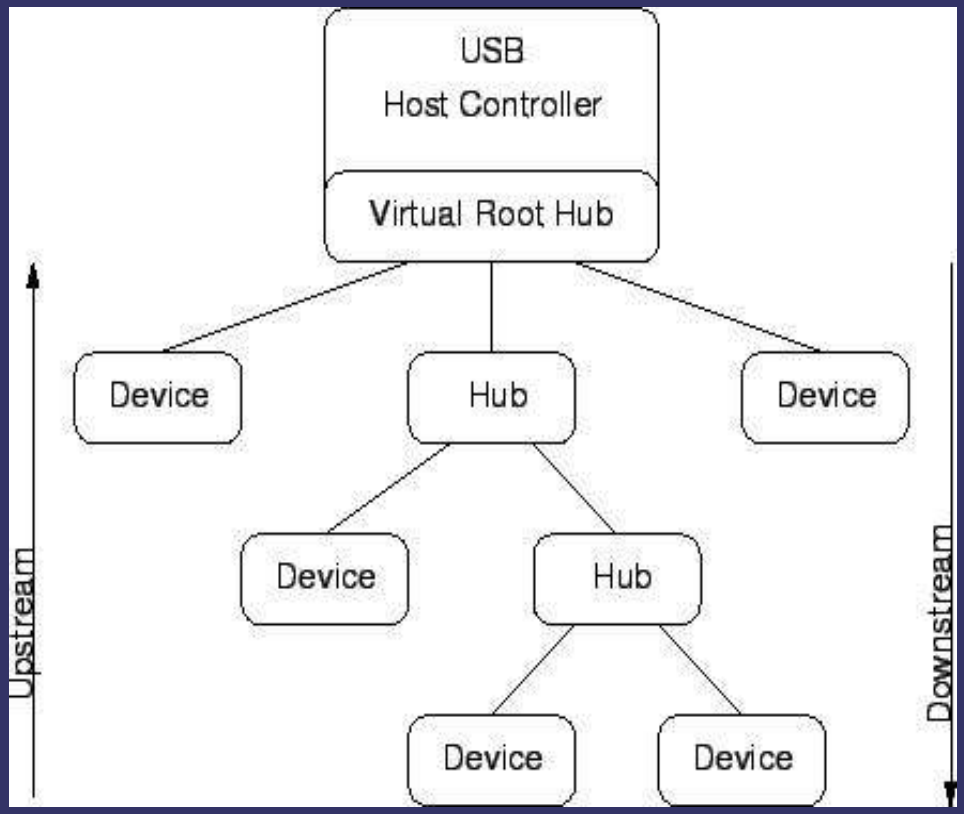
Overview

- General USB Information
- Do we really need another API?
- The libusb API – Function Call Reference
- How do I use libusb?
- Good Implementation (gPhoto2)
- Future for USB (OnTheGo)

USB Terms

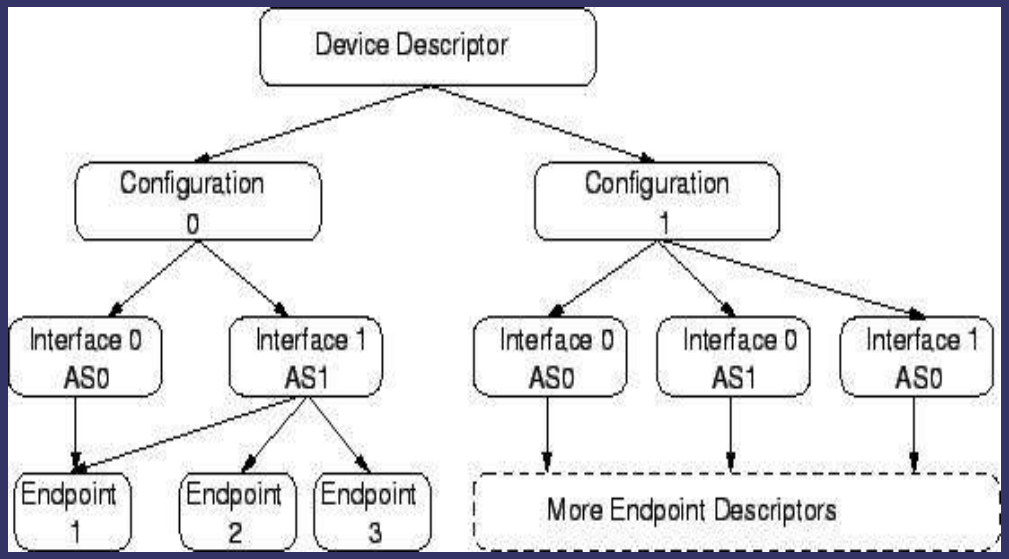
- USB – Universal Serial Bus
- OHCI – Open Host Controller Interface
- UHCI – Universal Host Controller Interface
- Alternate UHCI – UHCI
- EHCI – Enhanced Host Controller Interface
- Host – Device that is controlling Data Flow
- Endpoint – Device designation address
- Descriptor – Device characteristics
- Configuration – Device Data Flow info
- Interfaces – Devices with Devices
- Enumeration – Logical/Dynamic USB Devices

USB Topology



- Host controls all data flow
- Hubs allow multiple device endpoints
- Devices provide information to the host

USB Descriptors



- Descriptors give information about the USB Structure associated with a connected device.

usbdevfs

- Command-Line:
  - mount -t usbdevfs usbdevfs /proc/bus/usb

- /etc/fstab:
  - usbdevfs /proc/bus/usb usbdevfs defaults 0 0

USB Tools

- Linux Hotplugging [Last Release 05/2003]
  - Has a large scope and covers SCSI, CardBus, USB, IEEE1394, Networking, etc..
  - Standard Kernel Module as of 2.4 (Linux Specific)
- jUSB [0.4.4 Last Release 02/2001]
  - Object-Oriented USB Access based on jdk1.1 embedded
  - Provides C/C++ compatibility
  - Abstracts USB Topology from kernel internals
- libusb [0.1.7 Last Release 11/2002]
  - Supports Mac OS X, BSD's, and Linux
  - Potentially Windows can be supported (WDM)
  - Abstracts USB Topology from kernel internals

libusb - The Open Source Project

- Maintainer: Johannes Erdfelt
  - Kernel Developer (Alternative UHCI)
- Project Website
  - http://libusb.sourceforge.net/
- About:
  - The libusb project. It's aim is to create a library for use by user level applications to access USB devices regardless of OS.

libusb - API Overview

- I. Core
- II. Device operations
- III. Control Transfers
- IV. Bulk Transfers

libusb - Core Operations

- usb_init -- Initialize libusb
- usb_find_busses -- Find all USB busses on system
- usb_find_devices -- Find all devices on all USB devices
- usb_get_busses -- Return the list of USB busses found

libusb - Device Operations

- usb_open -- Opens a USB device
- usb_close -- Closes a USB device
- usb_set_configuration -- Sets the active configuration of a device
- usb_set_altinterface -- Sets the active alternate setting of the current interface
- usb_resetep -- Resets state for an endpoint
- usb_clear_halt -- Clears any halt status on an endpoint
- usb_reset -- Resets a device
- usb_claim_interface -- Claim an interface of a device
- usb_release_interface -- Releases a previously claimed interface

libusb - Control Transfers

- usb_control_msg -- Send a control message to a device
- usb_get_string -- Retrieves a string descriptor from a device
- usb_get_string_simple -- Retrieves a string descriptor from a device using the first language

libusb - Bulk Transfers

- usb_bulk_write -- Write data to a bulk endpoint
- usb_bulk_read -- Read data from a bulk endpoint

libusb – API Usage

```
#include<usb.h>

struct usb_bus *busses;
struct usb_bus *bus;

usb_init();
usb_find_busses();
usb_find_devices();
busses = usb_get_busses();

for (bus = busses; bus; bus = bus->next) {
        struct usb_device *dev;

        for (dev = bus->devices; dev; dev = dev->next) {
                if (dev->descriptor.bDeviceClass == 0x10) {
                /* Open the device and do your processing */
                }
        }
}
```

libusb – Data Structures

```
struct usb_dev_handle {
  int fd;
  struct usb_bus *bus;
  struct usb_device *device;
  int config;
  int interface;
  int altsetting;
  void *impl_info;
};
```

libusb - Data Structures

```
struct usb_device {
  struct usb_device *next, *prev;
  char filename[PATH_MAX + 1];
  struct usb_bus *bus;
  struct usb_device_descriptor descriptor;
  struct usb_config_descriptor *config;
  void *dev;  /* Darwin support */
};

struct usb_bus {
  struct usb_bus *next, *prev;
  char dirname[PATH_MAX + 1];
  struct usb_device *devices;
};
```

libusb - gPhoto2

- UNIX/Linux Interface to Digital Cameras
- USB cameras controlled via libusb

gPhoto2 – Example Function

```
static int
gp_port_usb_write (GPPort *port, const char *bytes, int size)
{
        int ret;

          if (!port || !port->pl->dh)
                  return GP_ERROR_BAD_PARAMETERS;

          ret = usb_bulk_write (port->pl->dh, port->settings.usb.outep,
                          (char *) bytes, size, port->timeout);
        if (ret < 0)
                  return (GP_ERROR_IO_WRITE);

        return (ret);
}
```

Information

- http://www.usb.org, Universal Serial Bus Implementers Forum
- http://www.linux-usb.org, Linux USB Developer information.
- http://usb.cs.tum.edu, Linux USB Developer Pages
- http://libusb.sourceforge.net, libusb USB API
- http://linux-hotplug.sourceforge.net, Linux Hotplugging
- http://jusb.sourceforge.net, jUSB

Questions?