BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

CHƯƠNG 6. PHÁT TRIỂN TRÌNH ĐIỀU KHIỂN CHO THIẾT BỊ CHUẨN USB

Ngành: Công nghệ thông tin

Chuyên ngành: Kỹ thuật phần mềm nhúng và di động

Mã số: 52.48.02.01

Nội dung

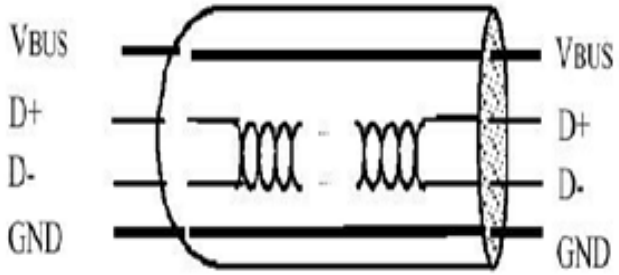
- Giao thức USB
- Phát hiện thiết bị và cấu trúc thông tin usb trên Linux
- Quy trình phát triển trình điều khiển USB trong Linux
- Ví dụ minh họa về xây dựng trình điều khiển cho ổ nhớ USB

Giao thức USB

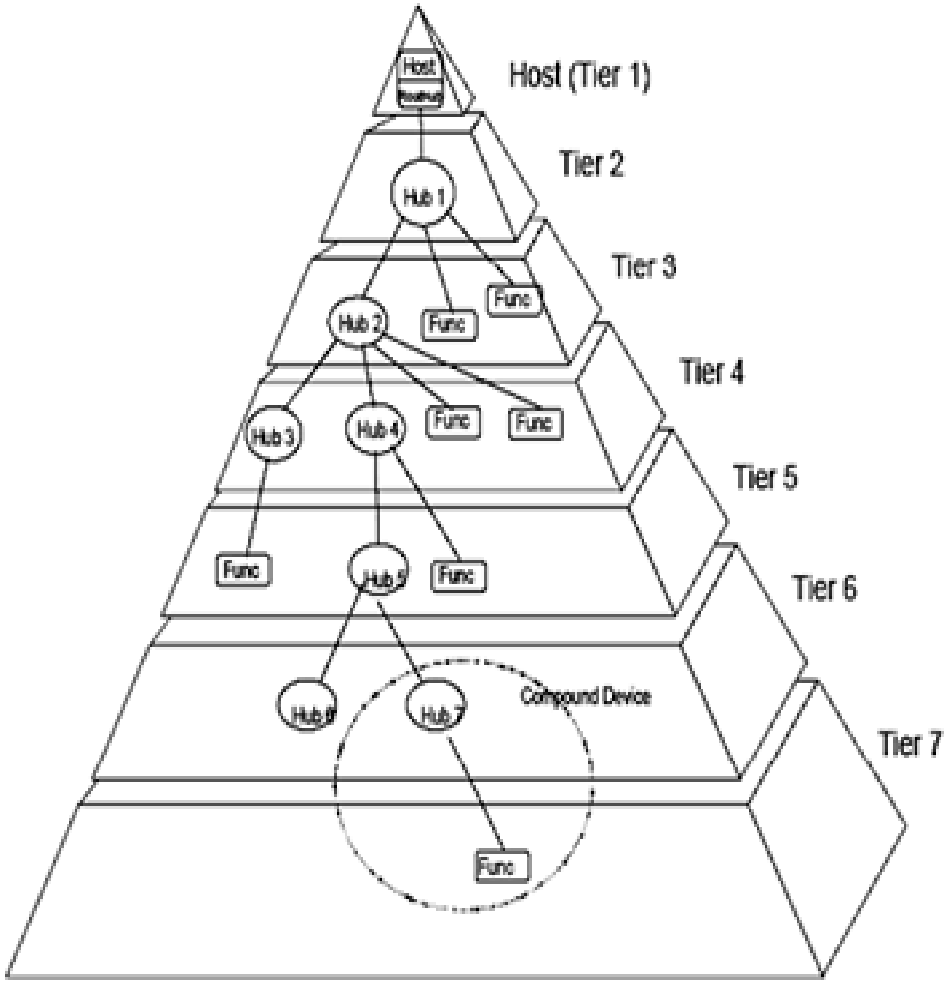
- Chuẩn tín hiệu
- Thiết bị USB
- Quá trình hoạt động
- Các chế độ truyền

Chuẩn tín hiệu

- USB (Universal Serial Bus) là một chuẩn giao tiếp nối tiếp, sử dụng 4 đường tín hiệu
  - 2 đường cấp nguồn DC (VBUS-5V và GND)
  - 2 đường còn lại là D+ và D- cho phép truyền dữ liệu
- Cổng USB trên máy tính cho phép cấp nguồn nuôi ra bên ngoài với dòng lên tới 500mA
  - Các thiết bị sử dụng ít điện năng như chuột, thẻ nhớ USB, v.v. đều có thể lấy trực tiếp nguồn từ cổng USB



Thiết bị USB



3 loại thiết bị USB

- USB host
- Thiết bị USB
- USB hub

Thiết bị USB

- **USB host**
  - USB host là thiết bị đóng vai trò điều khiển toàn bộ mạng USB và có thể kết nối lên tới tối đa 126 thiết bị
    - Trên máy tính, một USB Host được gắn trên bo-mạch chính
    - Để giao tiếp và điều khiển các thiết bị USB, khối điều khiển USB cần được thiết kế tích hợp với USB RootHub (hub mức cao nhất)
  - Vai trò của USB host
    - ✓Trao đổi dữ liệu với các thiết bị USB
    - ✓Điều khiển đường truyền USB:
      - Quản lý các thiết bị cắm vào hoặc tháo ra khỏi Bus USB thông qua quá trình điểm danh
      - Phân xử luồng dữ liệu trên Bus, đảm bảo các thiết bị đều có cơ hội trao đổi dữ liệu tùy thuộc vào cấu hình của mỗi thiết bị

Thiết bị USB

- **Thiết bị USB**
  - Là các thiết bị đóng vai trò như các thiết bị khách giao tiếp với USB Host
    - ✓ Một chú ý quan trọng đó là các thiết bị này hoàn toàn đóng vai trò bị động, không bao giờ được tự ý gửi gói tin lên USB Host hay gửi gói tin giữa các thiết bị USB với nhau, tất cả đều phải thông qua quá trình điều khiển của USB Host
    - ✓ Chức năng của thiết bị thiết bị USB:
      - Trao đổi dữ liệu với USB Host
      - Phát hiện gói tin hay yêu cầu từ USB Host theo giao thức USB.
- **USB Hub**
  - Đóng vai trò như các Hub trong mạng Ethernet và cấp nguồn cho các thiết bị USB
  - Sử dụng USB hub giúp tăng số lượng thiết bị USB kết nối vào hệ thống mà không phải thay đổi gì về chương trình, giao thức

Quá trình hoạt động

- **Quá trình điểm danh**
  - Là quá trình USB Host phát hiện các thiết bị cắm vào và rút ra khỏi đường truyền USB
  - Khi một thiết bị tham gia vào Bus USB, USB Host sẽ tiến hành đọc các thông tin mô tả của thiết bị USB để thiết lập địa chỉ (NodeID) và chế độ hoạt động tương ứng cho thiết bị USB
  - Các địa chỉ sẽ được đánh từ 1 đến 126 nên chuẩn USB cho phép kết nối 126 thiết bị vào đường truyền dữ liệu
  - Khi thiết bị rút ra khỏi đường truyền USB, địa chỉ này sẽ được thu hồi

Quá trình hoạt động

- **Quá trình truyền dữ liệu**
  - Hai khái niệm khó và quan trọng nhất trong chuẩn USB đó là khái niệm giao diện (Interface) và điểm cuối (Endpoint)
  - Chỉ thiết bị USB mới có Endpoint, USB Host không có Endpoint. Một thiết bị USB sẽ có thể có nhiều giao diện, một giao diện có thể sử dụng nhiều Endpoint
  - Có thể xem giao diện chính là các dịch vụ khác nhau mà thiết bị cung cấp còn các Endpoint chính là các cổng cần thiết cho mỗi dịch vụ

Quá trình hoạt động

- **Quá trình truyền dữ liệu** (tiếp)
  - Endpoint được phân loại theo hướng truyền dữ liệu nhìn từ phía USB Host. Cụ thể:
    - ✓Các endpoint IN truyền dữ liệu từ thiết bị USB tới USB Host
    - ✓Các endoint OUT truyền dữ liệu từ USB Host tới thiết bị USB
  - Trên Linux, ta có thể dùng lệnh *lsusb* để xem các thông tin này với bất cứ thiết bị USB nào cắm vào Bus.
  - Đề truyền được dữ liệu theo chuẩn USB
    - ✓ Các thiết bị USB phải được kết nối với USB Host thông qua các Pipe (đường ống)
    - ✓ Mỗi Pipe sẽ nối một Endpoint của thiết bị USB với USB Host

Các chế độ truyền

- **Truyền điều khiển:** là chế độ truyền được tất cả các thiết bị USB hỗ trợ để truyền các thông tin điều khiển.
- **Truyền ngắt:** sử dụng cho các thiết bị cần truyền một lượng dữ liệu nhỏ, tuần hoàn theo thời gian ví dụ như chuột, bàn phím.
  - Ví dụ cứ 10s một lần, USB Host sẽ gửi request xuống và thiết bị USB sẽ trả dữ liệu về cho USB Host.
- **Truyền theo khối:** sử dụng cho các thiết bị cần truyền một lượng dữ liệu lớn, yêu cầu độ chính xác tuyệt đối
- **Truyền nhanh:** sử dụng cho các thiết bị cần truyền một lượng dữ liệu lớn với tốc độ rất nhanh, đảm bảo ràng buộc về thời gian thực
  - Chấp nhận hy sinh độ chính xác ở một mức nhất định như các thiết bị nghe nhạc, xem phim kết nối theo chuẩn USB

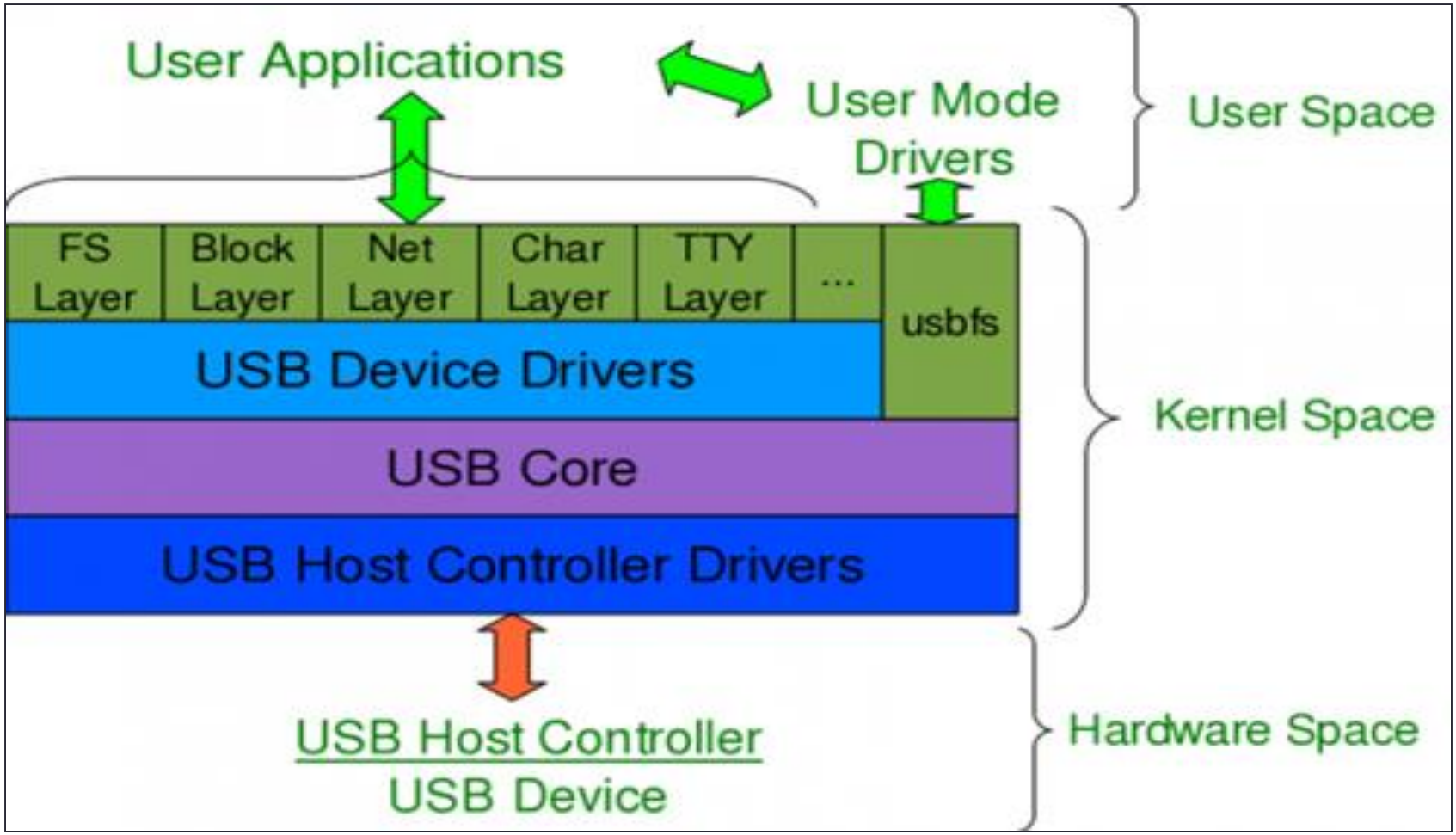
Phát hiện thiết bị và cấu trúc thông tin usb trên Linux

- Quá trình phát hiện thiết bị
- Cấu trúc thông tin về thiết bị USB

Quá trình phát hiện thiết bị

- Khi có một thiết bị USB hợp lệ được cắm vào hệ thống, dù có trình điều khiển hay không thì nó cũng vẫn được phát hiện bởi phần cứng ở tầng nhân của hệ thống Linux mà đã được hỗ trợ giao thức USB
- Việc phát hiện ra thiết bị USB cắm vào được thực hiện bởi khối điều khiển USB host
- Khối điều khiển USB host sẽ thu thập và diễn giải các thông tin ở tầng thấp đến các thông tin đặc tả giao thức USB ở tầng trên
- Các thông tin về thiết bị theo khuôn dạng quy định của giao thức USB lại tiếp tục được đưa vào tầng USB core tổng quát trong tầng nhân
  - Điều này giúp cho các thiết bị USB được hệ thống phát hiện ở tầng nhân dù chưa có trình điều khiển

Quá trình phát hiện thiết bị



Quá trình phát hiện thiết bị

- Xem thông tin các thiết bị USB sử dụng lệnh *lsusb*

```
Bus 001 Device 010: ID 090c:1000 Feiya Technology Corp. Flash Drive
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x090c Feiya Technology Corp.
  idProduct          0x1000 Flash Drive
  bcdDevice            11.00
  iManufacturer           1
  iProduct                2
  iSerial                 3
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
```

Quá trình phát hiện thiết bị

- thông tin chi tiết về thiết bị usb đã được phát hiện có thể xem thông qua tệp tin thông tin thiết bị trong /proc
  - có thể sử dụng lệnh cat /proc/bus/usb/devices để mở
  - Chú ý: nhiều hệ thống Linux mới không còn chứa thông tin ở đường dẫn này nữa mà đặt trong /sys/kernel/debug/usb/devices

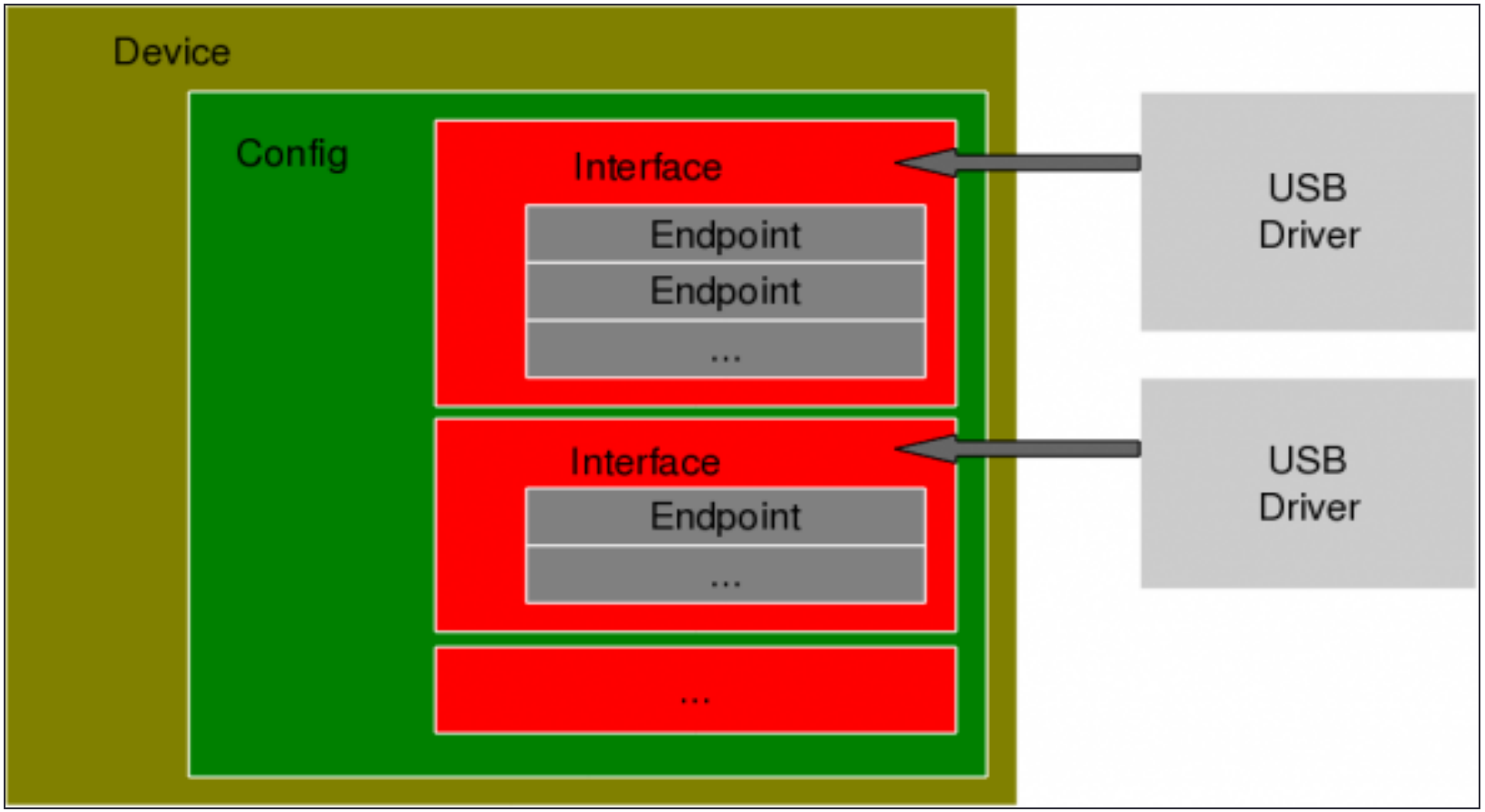
Cấu trúc thông tin về thiết bị USB

- Tất cả các thiết bị USB hợp lệ đều chứa thông tin về một hoặc một vài cấu hình
- Một cấu hình của thiết bị USB giống như một bản hồ sơ về thiết bị
- Mỗi cấu hình thiết bị lại có một hoặc một số giao diện (interfaces). Mỗi một giao diện tương ứng với một chức năng (function) của thiết bị
- không giống như các trình điều khiển thiết bị đơn chức năng khác, trình điều khiển cho thiết bị usb thường sẽ gắn với một giao diện nào đó của thiết bị USB đấy chứ không thường cho toàn bộ chức năng của thiết bị

Cấu trúc thông tin về thiết bị USB

- thiết bị USB có thể có nhiều trình điều khiển và nhiều giao diện khác nhau có thể cùng chung một trình điều khiển, nhưng mỗi giao diện chỉ thuộc về 1 trình điều khiển
- các thiết bị USB hợp lệ đều ngầm định sử dụng endpoint 0 để truyền điều khiển (control), đây là endpoint duy nhất có 2 chiều

Cấu trúc thông tin về thiết bị USB



Quy trình phát triển trình điều khiển USB trong Linux

- Tìm hiểu thiết bị USB
- Khai báo danh sách thiết bị được điều khiển
- Khai báo cấu trúc liên quan đến thiết bị
- Đăng ký thiết bị
- Xây dựng hàm thăm dò thiết bị
- Xây dựng hàm ngắt cho thiết bị
- Xây dựng các hàm mở, đọc, ghi cho thiết bị

Tìm hiểu về thiết bị usb

- chạy lệnh lsusb trên terminal, tất cả các thiết bị USB đang kết nối với máy tính sẽ được liệt kê ra. Từ đó ta có thể biết được idVendor và idProduct của thiết bị.
- Tiếp tục, gõ lệnh lsusb –vd <idVendor>:<idProduct> để hiển thị các thông tin về cấu hình USB của thiết bị

Khai báo danh sách các thiết bị được điều khiển

- Khi viết một trình điều khiển thiết bị USB, lập trình viên cần xác định trình điều khiển này sẽ được sử dụng cho các thiết bị hoặc các lớp thiết bị nào
- Cấu trúc **usb_device_id** cung cấp một kiểu thiết bị, nó có thể là một thiết bị cụ thể cũng có thể là một lớp thiết bị. Có một số macro để khởi tạo cấu trúc này
  - Ví dụ, ta có thể sử dụng macro USB_DEVICE (vendor, product) để tạo ra một cấu trúc usb_device_id với IDvendor và IDproduct.
  - Sau khi tạo ra một cấu trúc usb_device_id, cần phải khai báo cấu trúc này với USB Core, để làm việc này ta sử dụng macro:
  - MODULE_DEVICE_TABLE(usb, usb_device_id[]);

Khai báo cấu trúc dữ liệu liên quan tới thiết bị

- Các thông tin cần thiết bao gồm:
  - thiết bị cụ thể (được xác định bởi cấu trúc usb_device), thông tin các Configuration, Interface, Endpoint của thiết bị
  - Tùy theo đặc điểm của từng phần cứng cụ thể mà ta định nghĩa một cấu trúc dữ liệu phù hợp

Khai báo cấu trúc dữ liệu liên quan tới thiết bị

Ví dụ: Với thiết bị có 1 configuration; mỗi configuration có 1 interface; mỗi interface có 2 endpoint là Bulk IN và Bulk OUT, ta có thể định nghĩa một cấu trúc dữ liệu như sau:

```
//cấu trúc dữ liệu lưu trữ các thông tin về thiết bị,
//endpoint, bộ đệm
struct usb_mydevice {
     //con trỏ tới cấu trúc mô tả thiết bị
     struct usb_device         *udev;
     struct usb_interface      *interface;
     //bộ đệm dữ liệu vào
     unsigned char              *bulk_in_buffer;
     //kích thước bộ đệm dữ liệu vào
     size_t                    bulk_in_size;
     //địa chỉ 2 Endpoint Bulk IN và OUT
     __u8                      bulk_in_endpointAddr;
     __u8                      bulk_out_endpointAddr;
};
```

Đăng ký và hủy đăng ký trình điều khiển USB

- Để tầng USB Core có thể nhận ra trình điều khiển, lập trình viên cần phải đăng ký trình điều khiển với nhân.
- Để đăng ký trình điều khiển USB, ta sử dụng hàm sau:
  - **usb_register(struct usb_driver &);**
  - Hàm này thường được gọi trong hàm khởi tạo mô-đun trình điều khiển.
  - Tham số cần truyền cho hàm này là một con trỏ tới cấu trúc **usb_driver**. Cấu trúc này bao gồm các thông tin về trình điều khiển

Đăng ký và hủy đăng ký trình điều khiển USB

- Để hủy đăng kí một trình điều khiển thiết bị USB, ta sử dụng hàm sau:
  - **usb_deregister(struct usb_driver &);**
  - Hàm này thường được gọi trong hàm kết thúc mô-đun trình điều khiển

Hàm thăm dò thiết bị

- Khi thiết bị mới được kết nối tới hệ thống, nếu trình điều khiển được chỉ định cho thiết bị đó thì **hàm thăm dò (probe) của trình điều khiển sẽ được gọi**.
- USB Core truyền tới hàm thăm dò một con trỏ tới cấu trúc **usb_interface** mô tả Interface được chọn trên thiết bị. Nguyên mẫu hàm thăm dò như sau:
  - **int (*probe) (struct usb_interface *intf, const struct usb_device_id* id);**
- Trong hàm thăm dò, trình điều khiển cần thực hiện một số công việc sau:
  - Lấy ra địa chỉ các Endpoint cần dùng, lấy ra kích thước các bộ đệm cho thiết bị
  - Cấp phát bộ đệm
  - Lưu lại các thông tin như: địa chỉ Endpoint, kích thước bộ đệm, địa chỉ bộ đệm, v.v.
  - Đăng ký lớp thiết bị cho trình điều khiển.

Đăng kí lớp thiết bị

- Trình điều khiển cũng cần đăng ký lớp thiết bị mà nó quản lý. Mỗi lớp thiết bị khác nhau, có thể có các hàm đăng ký khác nhau.
- Nếu trình điều khiển muốn làm việc với thiết bị như kiểu Input, sử dụng hàm:
  - input_register_device(struct input_dev*)
- Nếu trình điều khiển muốn làm việc với thiết bị như một thiết bị kiểu kí tự, sử dụng hàm:
  - usb_register_dev(struct usb_interface*, struct usb_class_driver*)

Đăng kí lớp thiết bị

- Tham số thứ hai là con trỏ tới cấu trúc **usb_class_driver**
  - Cấu trúc này chứa một trường quan trọng là một con trỏ tới cấu trúc **file_operations** – đã định nghĩa các thao tác trên thiết bị

Hàm ngắt kết nối thiết bị

- Khi thiết bị được gỡ bỏ ra khỏi hệ thống, hàm ngắt kết nối được gọi. Hàm này có nguyên mẫu như sau:
  - void (*disconnect) (struct usb_interface* intf);
- Trong hàm này cần thực hiện hai công việc: Hủy các dữ liệu về thiết bị đã lưu trữ từ hàm thăm dò và hủy đăng ký lớp thiết bị.
- Để hủy dữ liệu lưu trữ, ta thiết lập dữ liệu NULL cho interface intf:

usb_set_intfdata(intf, NULL);

Để hủy đăng ký lớp thiết bị, ta thực hiện:

usb_deregister_dev(struct usb_interface* , struct usb_class_driver*);

Các hàm mở, đọc và ghi thiết bị

- **Mở tệp tin thiết bị**
- Để thực hiện thao tác mở thiết bị, ta gọi hàm sau:
  - static int mydevice_open(struct inode *inode, struct file *file);
- Thao tác mở thiết bị có tác dụng chuẩn bị cho các hành động đọc, ghi sau đó.
- Trong hàm trên, ta sử dụng hàm **usb_get_intfdata()** để lấy ra các thông tin liên quan tới thiết bị đã lưu trữ từ hàm thăm dò **probe()** bằng hàm **usb_set_intfdata** và thiết lập dữ liệu này cho cấu trúc file.

```
dev = usb_get_intfdata(interface);
file->private_data = dev;
```

Các hàm mở, đọc và ghi thiết bị

- Thao tác đọc dữ liệu dược thực hiện thông qua hàm:
  - static ssize_t mydevice_read(struct file *file, char __user *buffer, size_t count, loff_t *ppos);
- Hàm này thực hiện hai việc chính: một là là tạo ra các URB (USB request block) yêu cầu dữ liệu và xác nhận nó tới tầng USB Core, hai là chuyển dữ liệu nhận được từ các URB sang không gian người dùng bằng hàm:
  - unsigned long copy_to_user(void __user * to, const void * from, unsigned long size);
- Thao tác ghi dữ liệu được thực hiện thông qua hàm:
  - static ssize_t my_device write(struct file *file, const char __user *buffer, size_t count, loff_t *ppos);

Ví dụ minh họa về xây dựng trình điều khiển cho ổ nhớ USB

- Sử dụng một thiết bị nhớ USB để tìm hiểu quá trình hệ thống Linux làm việc với thiết bị này và thử nghiệm viết một trình điều khiển đơn giản cho nó
- Các bước chính
  - Đăng ký trình điều khiển thiết bị nhớ USB
  - Thông tin USB endpoint và xây dựng hàm thăm dò
  - Cài đặt các thao tác trao đổi dữ liệu với thiết bị USB

Đăng ký trình điều khiển thiết bị nhớ USB

- Các hàm API của tầng USB core để thực hiện đăng ký và hủy đăng ký thiết bị trong tệp tiêu đề <linux/usb.h> như sau:
  - int usb_register(struct usb_driver *driver);
  - void usb_deregister(struct usb_driver *);
- Cấu trúc usb_driver chứa các trường cung cấp tên trình điều khiển, bảng ID để dò thiết bị tự động, và 2 hàm được gọi bởi USB core trong quá trình kết nối thiết bị (hot plugging) và gỡ thiết bị (hot removal) chứa trong tệp tin pen_register.c:

Thử nghiệm

- Thử nghiệm driver này theo các bước sau:
  - Biên dịch driver sử dụng Makefile (tạo file pen_driver.ko)
  - Nạp driver sử dụng insmod pen_driver.ko.
  - Kết nối thiết bị usb flash driver (cần loại bỏ driver mặc định usb-storage)
  - Kiểm tra file thiết bị được tạo ra tự động /dev/pen0
  - Kiểm tra tra thông tin ghi log bằng lệnh dmesg
  - Có thể thử đọc/ghi file thiết bị tạo ra /dev/pen0
  - Ngắt kết nối thiết bị, và kiểm tra lại file thiết bị /dev/pen0
  - Gỡ driver khởi hệ thống rmmod pen_driver.

HỎI - ĐÁP