АВРОРА
СВОЯ СИСТЕМА

Lab 5 USB

Ildar Kamaletdinov – team lead, Open Mobile Platform

with Dmitrii Alekhin – junior software developer as TA

Writing driver

› Information needed by the usb core, to call the right **probe()** and **disconnect()** driver functions. Such information is declared in a *usb_device_id* structure by the usb core's **init()** function.

```
struct usb_device_id {
     ...
     /* Used for product specific matches; range is inclusive */
     __u16          idVendor;
     __u16          idProduct;
     __u16          bcdDevice_lo;
     __u16          bcdDevice_hi;

     /* Used for device class matches */
     __u8      bDeviceClass;
     __u8      bDeviceSubClass;
     __u8      bDeviceProtocol;
     ...
};
```

Provide information for usb core

› We can use few useful macro here.

› **USB_DEVICE(vend, prod)** – to define idVendor and idProduct for our device.

› **MODULE_DEVICE_TABLE(type, name)** – to define our table.

```
static struct usb_device_id pen_table[] =
{
    { USB_DEVICE(0x058F, 0x6387) },
    {} /* Terminating entry */
};
MODULE_DEVICE_TABLE (usb, pen_table);
```

Registering driver in usb core

- To register driver in usb core few additional macros should be used also.
- **usb_register(struct device_driver *drv);**
  usb_deregister(struct device_driver *drv);

```
static struct usb_driver pen_driver =
{
    ...
};

static int __init pen_init(void)
{
    return usb_register(&pen_driver);
}

static void __exit pen_exit(void)
{
    usb_deregister(&pen_driver);
}

module_init(pen_init);
module_exit(pen_exit);
```

struct usb_driver

```
struct usb_driver {
     const char *name;                     //virtual unique name

     int (*probe) (struct usb_interface *intf,
                const struct usb_device_id *id);    //probe function

     void (*disconnect) (struct usb_interface *intf);    //disconnect function

     int (*unlocked_ioctl) (struct usb_interface *intf, unsigned int code,
               void *buf);

     int (*suspend) (struct usb_interface *intf, pm_message_t message);
     int (*resume) (struct usb_interface *intf);
     int (*reset_resume)(struct usb_interface *intf);

     int (*pre_reset)(struct usb_interface *intf);
     int (*post_reset)(struct usb_interface *intf);

     const struct usb_device_id *id_table;         //id_table is used for hotplugging
     ...
};
```

struct usb_driver

- To register driver in usb core few additional macros could be used also.
- **usb_register(struct device_driver *drv);**
  usb_deregister(struct device_driver *drv);

```
static struct usb_driver pen_driver =
{
    .name = "pen_driver",
    .id_table = pen_table,
    .probe = pen_probe,
    .disconnect = pen_disconnect,
};
```

Finally

- **probe()** function will be called when usb core detects `registered device`. We must return 0 if we are sure that kernel detected proper device. We can additionally ensure that it is our device but we must not spend much time there.
- **disconnect()** function well be called in device plug out from usb port.

```
static int pen_probe(struct usb_interface *interface, const struct usb_device_id *id)
{
    printk(KERN_INFO "Pen drive (%04X:%04X) plugged\n", id->idVendor,
                                id->idProduct);
    return 0;
}
```

Note: you might face conflict with default kernel driver for mass storage devices (usb-storage). Usually it is compiled as module so please don't forget to unload it first.

TASK

- Implement Lab 4 first (it will be used for further improvement).
- Add any USB device as an electronic key for your chardev (from lab 4). You can use any VID/PID: mouse, keyboard, usb stick, etc.
- Chardev (from lab 4) must not appear in the system unless electronic key is not inserted into USB port.
- In case of usb device removal (from lab 4) chardev must be also removed from **/dev** list but stack must not be destroyed.
- Add `error: USB key not inserted` to your userspace wrapper (**kernel_stack**) from lab 4.
- Graded output: source code with report including screenshots. (in PDF)

Acceptance criteria

- **A (20 points) –** app meets all listed criteria.
- **B (15-19 points)** – minor issues (for ex. Presence of electronic key is not checked on kernel module loading).
- **C (10-14 points)** – major issues (for ex. Stack is destroyed in case of USB device removal).

АВРОРА
СВОЯ СИСТЕМА

Thanks for your attention!

About US

Open Mobile Platform, LLC

Shortly:

- Founded in 2016
- Offices in Moscow, Nizhny Novgorod, Innopolis and St.Petersburg
- 300+ qualified IT specialists

Main products:

- OS Aurora + Aurora SDK
- Cloud Platform
  Aurora Center (Enterprise Mobility Management)
- Aurora TEE & Trusted Boot