Project Report: COP 4600 Operating Systems

P4: Device Drivers

For the Project 4, driver's basic purpose is to connect with virtual joystick device (chompstick) and read, interpret and submit the device status to Linux kernel as input device. For doing the same, a new device driver is written in the C language.

Function Description

Step1: Setup device. First thing first, initialize the uinput device. (/dev/uinput). Set it up for joystick variables like X-Asix, Y-Axis or Button. Setup the vendor id, product id of virtual joystick.

Step2: Open USB device (virtual joystick) using vendor id and product id. Function used is libusb_open_device_with_vid_pid from libusb library.

Step3: Lock the device interface for read / write operations**.** This is again a function from libusb library as libusb_claim_interface.

Step 4: Wait for interrupt from virtual joy stick device. libusb_interrupt_transfer function from libusb is used for data fetch / read from virtual joystick device. It's going to give us 1 byte of data. We will parse it into bits to know the state of virtual joystick device.

Step 5: Once data is received from device on its state changed, parse the byte into bits and find out the state of x-axis, y-axis, and button state.

Step 6: Here we are going to emit the events as per deriver specifications. For example, if state of x-axis is 1, publish -32767 as value for joystick device.

The libusb open source library that allows you to communicate with USB devices from userspace is used. It is the core library used for device connection (open, read interrupt etc). Also the uinput is a kernel module that makes it possible to emulate input devices from userspace. By writing to /dev/uinput (or /dev/input/uinput) device, a process can create a virtual input device with specific capabilities. Hence in this case, to submit events to linux kernel input device, uinput input_event is used.z

The Driver connects with virtual device (chompapp) using vendor id and product id. Now, driver program used libusb function libusb_interrupt_transfer to listen for interrupts raised by virtual device on its state change. For example, if the x-axis state changes or y-axis state changes or button is in the on/off state. When a user performs an action on virtual device, driver catches the interrupt, reads the input byte as per device specification, interpret the state (bit processing as per given specification of virtual state device) and emits an input event as true joystick event to device input system.

Finally the jstest utility was used to verify the input events via /dev/input/js0. The corresoponding button and axes information was hence displayed. Hence on running the jstest utility, it depicts the true state of virtual joystick device (Chompapp) as per its design specifications.

Bugs

No device driver registration to kernel

Challenges Faced

- How to connect with USB devices connected with the system?
- How to read the data from USB device?
- Printing of bits of a byte?

Screencast:

https://youtu.be/ZSDY0ZaFG-c

External References

- https://libusb.info/
- https://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/
- https://www.kernel.org/doc/html/v4.12/input/uinput.html
- https://stackoverflow.com/questions/9280654/c-printing-bits