Ross Spencer

Project 4 Report

Files modified:

chompdrv.cpp:

Changes: I added a send helper method and a main where my driver daemon happens.

Functions:

void send(int uinputFile, int typeParam, int codeParam, int valParam):

This helper function creates a new input_event struct, sets its attributes to the parameters, and then writes it to the open uinputFile that uinput is monitoring. I used reference 2 as a guide for this to simplify the 10 lines each button value needed.

int main():

I utilize libusb to connect to the USB subsystem and uinput to emulate a joystick in userspace.

In my main, I first set up the libusb session (a libusb_context) and initialize it and find the USB device with the VendorID 0x9A7A and ProductID 0xBA17 as given in the spec, using reference 1 as my resource for everything libusb. I then make a uinput device to pipe the signals being read from libusb into Linux and have the VM actually register those signals, using reference 2 for all things uinput. Then I set up all of the uinput I/O control calls to allow the daemon to control the Linux joystick buttons and begin my while-loop. From here, I read into a char (8 bits) 1 byte of data using libusb_bulk_transfer to get the state of chompapp, and then, if sucessful, parse this using bitshifting (reference 3) to get the bit values in the byte. Since the byte is stored backwards in Little Endian, I converting the first 2 bits (technically last 2) from binary to decimal to get yaxis and do the same for the corresponding bits to get xaxis and button. I then have 3 switch cases which send the correct value of the ABS_Y axis, ABS_X axis, and BTN_JOYSTICK (reference 4) from the spec based on the value from chompapp. If the read is unsuccessful, it makes the while-loop terminate and the program end.

Testing:

I ran my driver and added a bunch of code to print out the bulk_transfer result to make sure it matched the hex number in chompapp and then parsed the bits and made sure the y, x, and button values also matched. Once I got that part working completely, I ran jstest and made sure that the resulting values corresponded to both the chompapp state and the pdf spec. To test for memory leaks, I attempted to use valgrind, but even after changing the permissions of my executable with chmod I couldn't get it to run correctly and gave up on that idea since people in the Slack said we'd have memory leaks from libusb regardless. I also tried to get GatorRaider working, but Chrome kept crashing and Gmail wouldn't let me send over an apk file for security reasons. The app crashed once I got it over Firefox; I also don't have a joystick, so it wouldn't have helped anyways.

Words: 487

Link to Unlisted Video:

https://youtu.be/1memAhO6uTA

References:

[1] https://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/ for teaching me how to do a bulk transfer from the device in libusb

[2] https://www.kernel.org/doc/html/v4.12/input/uinput.html for teaching me how uinput works and how to make keyboard/joystick events, and for the idea behind my send function (theirs was emit)

[3] https://stackoverflow.com/questions/37487528/how-to-get-the-value-of-every-bit-of-an-unsigned-char for guiding me on bitshifting to get the individual bits of the byte into an array in Little Endian format

[4] https://stackoverflow.com/questions/39559063/libsuinput-creating-joystick-with-more-than-one-button and https://elixir.bootlin.com/linux/latest/source/include/uapi/linux/input-event-codes.h#L364 for helping me figure out how to mess with the axes