How To Use The linux/joystick.h library

Written By: Group Tetris

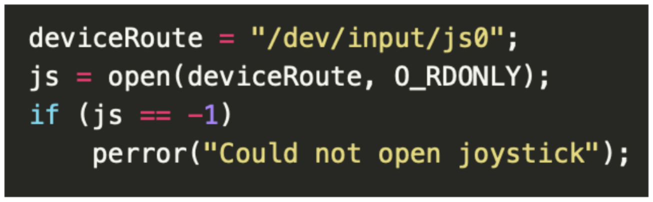
Introduction

This is a guide for how to capture inputs from the Xbox Controller using C and linux/joystick.h library. This guide will help you with getting inputs from the 4 buttons on the right (A, X, Y, B), the right bumper and left bumper.

1. Initialization

a. Include the header **linux/joystick.h**

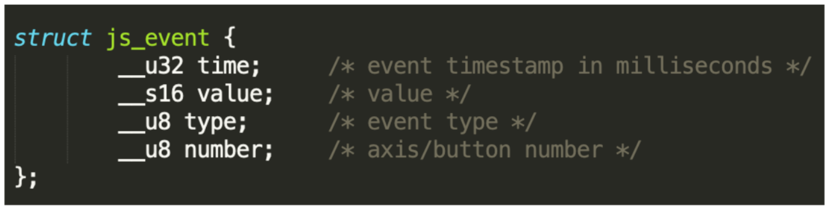
b. Open file **/dev/input/js0** in read only. If there are multiple joysticks, open files js1, js2, etc.



```
deviceRoute = "/dev/input/js0";
js = open(deviceRoute, O_RDONLY);
if (js == -1)
    perror("Could not open joystick");
```

2. js_events Explanation

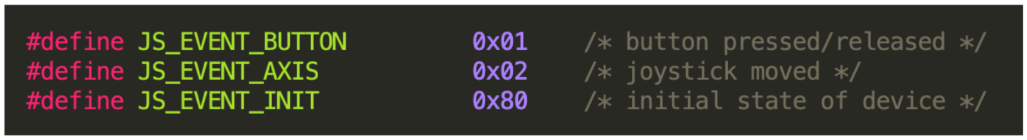
Your controller inputs will be read as a js_event. js_ events are defined as:



```
struct js_event {
        __u32 time;     /* event timestamp in milliseconds */
        __s16 value;    /* value */
        __u8 type;      /* event type */
        __u8 number;    /* axis/button number */
};
```

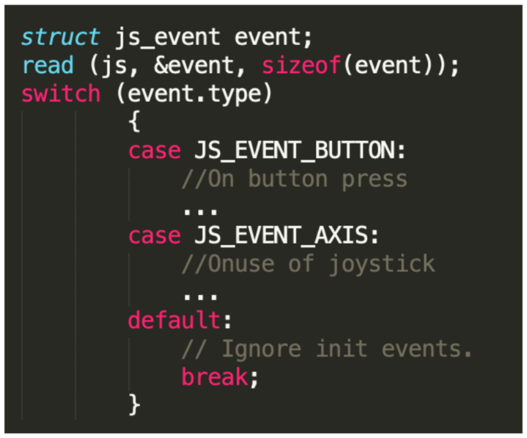
a) js_event .type

There are 3 types of js_events:



```
#define JS_EVENT_BUTTON         0x01    /* button pressed/released */
#define JS_EVENT_AXIS           0x02    /* joystick moved */
#define JS_EVENT_INIT           0x80    /* initial state of device */
```

JS_EVENT_INIT will be issued by the driver on open. The more crucial js_events are: JS_EVENT_BUTTON and JS_EVENT_AXIS, which signify whether a button or a joystick is used as an input from your controller. These event types can be checked by:



```
struct js_event event;
read (js, &event, sizeof(event));
switch (event.type)
        {
        case JS_EVENT_BUTTON:
            //On button press
            ...
        case JS_EVENT_AXIS:
            //Onuse of joystick
            ...
        default:
            // Ignore init events.
            break;
        }
```

b) js_event.number

js_event number corresponds to the button or axis which generated the event. For our XBox 360 controller, we had the following chart for js_event.type = JS_EVENT_BUTTON (From the inputs which we used)

js_event.number	**Corresponding button on controller**
0	A
1	B
2	X
3	Y
4	Left Bumper
5	Right Bumper

For js_event.type = JS_EVENT_AXIS, the corresponding axis for js_event number are listed below:

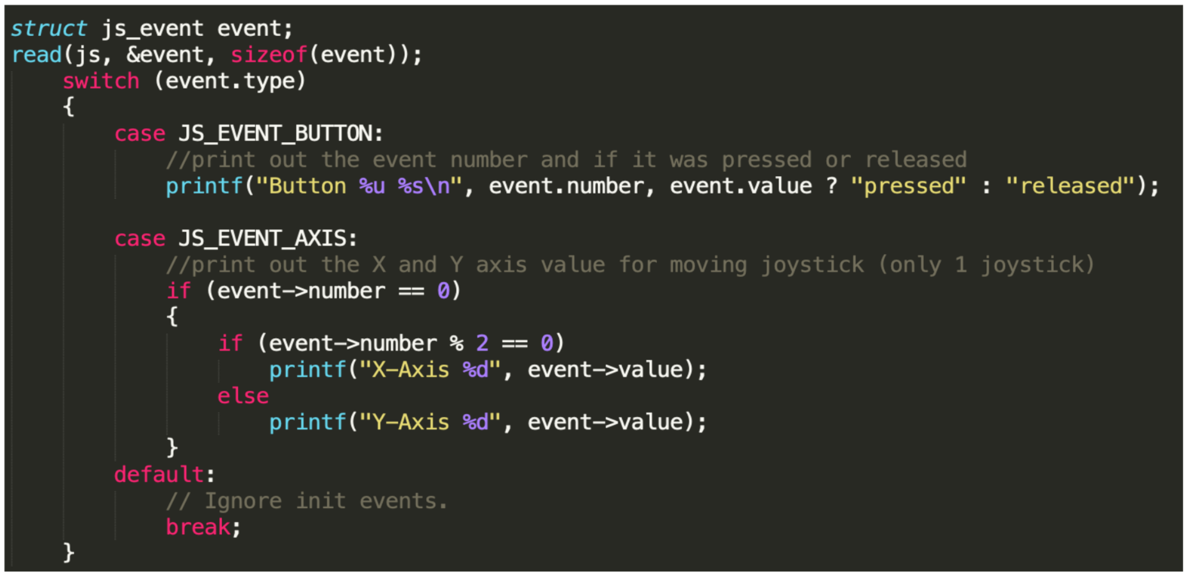
js_event number	Corresponding Axis
0	1st Axis X
1	1st Axis Y
2	2nd Axis X
3	2nd Axis Y
…	…

c) Js_event.value

When js_event.type = JS_EVENT_AXIS, the js_event value will be a signed integer between -32767 and +32767 which represents the position of the joystick along the axis.

When js_event.type = JS_EVENT_BUTTON, the js_event value will be either a 0 for releasing a button, or 1 for pressing a button.

The following sample prints out a button press, or joystick movement event:

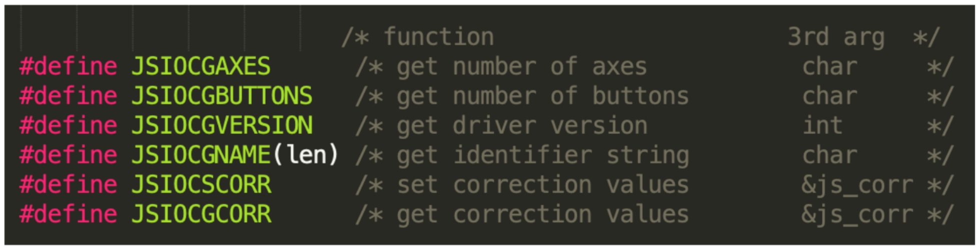


```c
struct js_event event;
read(js, &event, sizeof(event));
    switch (event.type)
    {
        case JS_EVENT_BUTTON:
            //print out the event number and if it was pressed or released
            printf("Button %u %s\n", event.number, event.value ? "pressed" : "released");

        case JS_EVENT_AXIS:
            //print out the X and Y axis value for moving joystick (only 1 joystick)
            if (event->number == 0)
            {
                if (event->number % 2 == 0)
                    printf("X-Axis %d", event->value);
                else
                    printf("Y-Axis %d", event->value);
            }
        default:
            // Ignore init events.
            break;
    }
```

3. IOCTLs

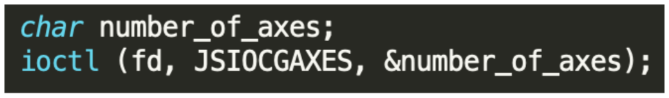
You can use ioctl (input/output control) functions in the joystick drivers. For our purposes we will use 2. But these options are available:



```
                                        /* function                         3rd arg  */
#define JSIOCGAXES                      /* get number of axes               char     */
#define JSIOCGBUTTONS                   /* get number of buttons            char     */
#define JSIOCGVERSION                   /* get driver version               int      */
#define JSIOCGNAME(len)                 /* get identifier string            char     */
#define JSIOCSCORR                      /* set correction values            &js_corr */
#define JSIOCGCORR                      /* get correction values            &js_corr */
```

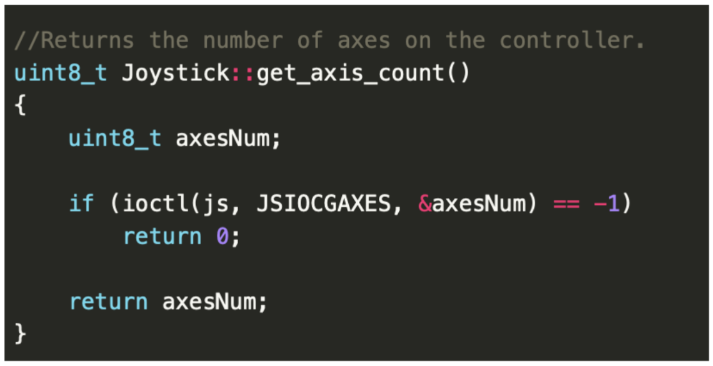
a) JSIOCGAXES

JSIOCGAXES allows you to read the number of axes used by your controllers. It can be done via:



```
char number_of_axes;
ioctl (fd, JSIOCGAXES, &number_of_axes);
```

Sample code for getting the number of axes on controller:



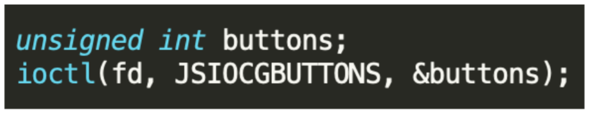
```
//Returns the number of axes on the controller.
uint8_t Joystick::get_axis_count()
{
    uint8_t axesNum;

    if (ioctl(js, JSIOCGAXES, &axesNum) == -1)
        return 0;

    return axesNum;
}
```

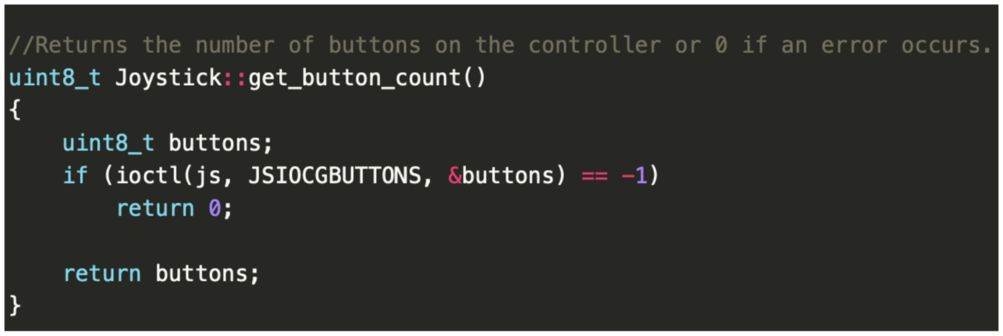
b) JSIOCGBUTTONS

JSIOCGBUTTONS are similar, in that it allows you to read the number of buttons to be used by your controller. It is done via:



```
unsigned int buttons;
ioctl(fd, JSIOCGBUTTONS, &buttons);
```

Sample code for getting the number of buttons on the controller:


```
//Returns the number of buttons on the controller or 0 if an error occurs.
uint8_t Joystick::get_button_count()
{
    uint8_t buttons;
    if (ioctl(js, JSIOCGBUTTONS, &buttons) == -1)
        return 0;

    return buttons;
}
```

Final Words

Hopefully this guide was helpful in allowing you to use a controller with linux/joystick. Results will vary based on what controller is being used. We have used an XBox 360 controller. We highly recommend that you check the references below, and especially the sample code by Jason White. Happy coding!

Troubleshooting

- If the file descriptor returned for the joystick is -1, ensure that you are opening the correct file. Try listing the items in the /dev/input directory and checking if there are js# listings. If there are none or some are missing, it may be the case that the controller is faulty.
- If js_event.type is Axis, and your js_event value is not 0 while the joystick is dead or not used, it should be recalibrated or adjusted.

References

- Linux joystick support documentation: https://www.kernel.org/doc/html/latest/input/joydev/index.html
- Joystick input with C++ (abstract SDL joystick input): https://www.youtube.com/watch?v=vbBEG6SvryA
- Reading joystick/gamepad event on Linux and displaying them: https://gist.github.com/jasonwhite/c5b2048c15993d285130