Sam Salin

ECE 373

5/16/16

1)

**Driver Source Code**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <getopt.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <pci/pci.h>

#include <stdint.h>

#include <linux/types.h>

#define MEM\_WINDOW\_SZ 0x0200000

#define LEDCTL 0x00e00//led control registers

#define GPRC 0x4074 //good packets received

#define MSK 0x000000 //mask/turn all off

#define LED0 0x4e

#define LED1 0x4e00

#define LED2 0x4e0000

#define TWOGREEN 0x4e4e0e

static struct pci\_filter filter; /\* Device filter \*/

static u32 contets; /\*hold original LED\_CTL contents\*/

static struct option opts[] = {

{"address", 1, NULL, 'a' },

{"debug", 0, NULL, 'D' },

{"device", 1, NULL, 'd' },

{"slot", 1, NULL, 's' },

{ 0, 0, NULL, '0' }

};

static void usage(char \*progname, char \*idfile)

{

printf( "Usage: %s [options] [device] (%s)\n\n"

"Options:\n"

"-D\t\tPCI debugging\n"

"-o\t\tRegister offset\n"

"Device:\n"

"-d [<vendor>]:[<device>]\t\t\tShow selected devices\n"

"-s [[[[<domain>]:]<bus>]:][<slot>][.[<func>]]"

"\tShow devices in selected slots\n\n",

progname, idfile);

}

static int print\_register(struct pci\_dev \*dev, u32 offset)

{

volatile void \*mem;

int dev\_mem\_fd;

dev\_mem\_fd = open("/dev/mem", O\_RDONLY);

if (dev\_mem\_fd < 0) {

perror("open");

return -1;

}

mem = mmap(NULL, MEM\_WINDOW\_SZ, PROT\_READ, MAP\_SHARED, dev\_mem\_fd,

(dev->base\_addr[0] & PCI\_ADDR\_MEM\_MASK));

if (mem == MAP\_FAILED) {

perror("mmap/readable - try rebooting with iomem=relaxed");

close(dev\_mem\_fd);

return -1;

}

printf("0x%x\n", \*((u32 \*)(mem + offset)));

close(dev\_mem\_fd);

munmap((void \*)mem, MEM\_WINDOW\_SZ);

return 0;

}

static int lightshow(struct pci\_dev \*dev)

{

int i;

u32 reg;

u32 tmp;

volatile void \*mem;

int dev\_mem\_fd;

dev\_mem\_fd = open("/dev/mem", O\_RDWR);

mem = mmap(NULL, MEM\_WINDOW\_SZ, PROT\_READ | PROT\_WRITE, MAP\_SHARED, dev\_mem\_fd, (dev->base\_addr[0] & PCI\_ADDR\_MEM\_MASK));

//save LEDCTL

tmp = \*((u32 \*)(mem + LEDCTL));

printf("LED REGISTER CONTAINS %x\n", tmp);

reg = tmp;

/\*turn on both green LED 2 sec\*/

reg = reg & MSK;

reg = reg | TWOGREEN;

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(2);

reg = reg & MSK;

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(2);

/\*turn all LED off 2 sec\*/

/\*do light sequence\*/

for (i=0; i<5; ++i){

//amber

reg = LEDCTL;

reg = reg & MSK;

reg = reg | LED0; //on

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(1);

reg = reg & MSK; //off

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(1);

//green on right(led0)

reg = reg | LED1; //on, should still be cleared

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(1);

reg = reg & MSK; //off

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(1);

//green on left

reg = reg | LED2; //on, should still be cleared

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(1);

reg = reg & MSK;

\*((u32 \*)(mem + LEDCTL)) = reg;

sleep(1);

}

\*((u32 \*)(mem + LEDCTL)) = tmp; //restore ledctl

tmp = \*((u32 \*)(mem + GPRC)); //grab good packet register

printf("device has received %d good packets\n",tmp);

close(dev\_mem\_fd);

munmap((void \*)mem, MEM\_WINDOW\_SZ);

}

int main(int argc, char \*\*argv)

{

int ch;

struct pci\_access \*pacc;

struct pci\_dev \*dev;

char \*errmsg;

char buf[128];

u32 offset = 0;

u32 value = 0;

u64 lvalue = 0;

int device\_specified = 0;

int got\_offset = 0;

int ret;

if (getuid() != 0) {

printf("%s: must be run as root\n", argv[0]);

exit(1);

}

pacc = pci\_alloc(); /\* Get the pci\_access structure \*/

if (pacc == NULL) {

perror("pci\_alloc");

exit(1);

}

pci\_filter\_init(pacc, &filter);

while ((ch = getopt\_long(argc, argv, "o:d:s:", opts, NULL)) != -1) {

switch (ch) {

case 'D':

pacc->debugging++;

break;

case 'o':

offset = strtol(optarg, NULL, 0);

got\_offset++;

break;

case 'd':

/\* Show only selected devices \*/

if ((errmsg = pci\_filter\_parse\_id(&filter, optarg))) {

printf("%s\n", errmsg);

exit(1);

}

device\_specified++;

break;

case 's':

/\* Show only devices in selected slots \*/

if ((errmsg = pci\_filter\_parse\_slot(&filter, optarg))) {

printf("%s\n", errmsg);

exit(1);

}

device\_specified++;

break;

case '?':

default:

usage(argv[0], pacc->id\_file\_name);

exit(1);

break;

}

}

if (!device\_specified) {

printf("No device given\n");

usage(argv[0], pacc->id\_file\_name);

exit(1);

}

if (!got\_offset) {

printf("No offset given\n");

usage(argv[0], pacc->id\_file\_name);

exit(1);

}

pci\_init(pacc); /\* Initialize the PCI library \*/

pci\_scan\_bus(pacc); /\* Get the list of devices \*/

if (pacc->debugging)

printf( "filter: "

"bus=0x%x slot=0x%x func=0x%x\n"

"\tvendor=0x%x device=0x%x\n\n",

filter.bus, filter.slot, filter.func,

filter.vendor, filter.device);

/\* Iterate over all devices to find the single one we want \*/

for (dev = pacc->devices; dev; dev = dev->next) {

if (pci\_filter\_match(&filter, dev))

break;

}

if (!dev) {

printf("no device found\n");

ret = -1;

goto out;

}

/\* Fill in header info we need \*/

pci\_fill\_info(dev, PCI\_FILL\_IDENT | PCI\_FILL\_BASES | PCI\_FILL\_SIZES);

printf( "%02x:%02x.%d (%04x:%04x)\n%s\n",

dev->bus, dev->dev, dev->func,

dev->vendor\_id, dev->device\_id,

pci\_lookup\_name(pacc, buf, sizeof(buf),

PCI\_LOOKUP\_VENDOR|PCI\_LOOKUP\_DEVICE,

dev->vendor\_id, dev->device\_id, 0, 0));

lightshow(dev); //do the light show

out:

pci\_cleanup(pacc); /\* Close everything \*/

return ret;

}

**Makefile**

I didn’t actually use a makefile since it was easy enough to compile at the command line with gcc, as can be seen in the typescript.

2) Since there is no kernel object involved in writing a userspace driver, a few conclusions can be drawn. First and foremost, things probably will not be quite as fast since user processes generally do not take precedence over kernel processes. Secondly, fewer permissions are needed since we don’t need to mess around with insmod/mknod/binding/unbinding drivers.

3) Typescript:

root@ece373:/media/usb/373/HW\_5# gcc HW\_5.c -lpci -lz

root@ece373:/media/usb/373/HW\_5# ./a.out -s 03:00.0 -o 0xE00

03:00.0 (8086:150c)

Intel Corporation 82583V Gigabit Network Connection

LED REGISTER CONTAINS 78406

device has received 0 good packets

root@ece373:/media/usb/373/HW\_5# ./a.out -s 03:00.0 -o 0xE00

03:00.0 (8086:150c)

Intel Corporation 82583V Gigabit Network Connection

LED REGISTER CONTAINS 78406

device has received 0 good packets

root@ece373:/media/usb/373/HW\_5# ./a.out -s 03:00.0 -o 0xE00

03:00.0 (8086:150c)

Intel Corporation 82583V Gigabit Network Connection

LED REGISTER CONTAINS 78406

device has received 3 good packets