chardev.c Makefile

chardev.c

/\*

 \* chardev.c: Creates a read-only char device that says how many times

 \* you have read from the dev file

 \*/

#include <linux/cdev.h>

#include <linux/delay.h>

#include <linux/device.h>

#include <linux/fs.h>

#include <linux/init.h>

#include <linux/irq.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/poll.h>

/\* Prototypes - this would normally go in a .h file \*/

static int device\_open(struct inode \*, struct file \*);

static int device\_release(struct inode \*, struct file \*);

static ssize\_t device\_read(struct file \*, char \_\_user \*, size\_t, loff\_t \*);

static ssize\_t device\_write(struct file \*, const char \_\_user \*, size\_t, loff\_t \*);

#define SUCCESS 0

#define DEVICE\_NAME "chardev" /\* Dev name as it appears in /proc/devices \*/

#define BUF\_LEN 80 /\* Max length of the message from the device \*/

/\* Global variables are declared as static, so are global within the file. \*/

static int major; /\* major number assigned to our device driver \*/

enum {

    CDEV\_NOT\_USED = 0,

    CDEV\_EXCLUSIVE\_OPEN = 1,

};

/\* Is device open? Used to prevent multiple access to device \*/

static atomic\_t already\_open = ATOMIC\_INIT(CDEV\_NOT\_USED);

static char msg[BUF\_LEN + 1]; /\* The msg the device will give when asked \*/

static struct class \*cls;

static struct file\_operations chardev\_fops = {

    .read = device\_read,

    .write = device\_write,

    .open = device\_open,

    .release = device\_release,

};

static int \_\_init chardev\_init(void)

{

    major = register\_chrdev(0, DEVICE\_NAME, &chardev\_fops);

    if (major < 0) {

        pr\_alert("Registering char device failed with %d\n", major);

        return major;

    }

    pr\_info("I was assigned major number %d.\n", major);

    cls = class\_create(THIS\_MODULE, DEVICE\_NAME);

    device\_create(cls, NULL, MKDEV(major, 0), NULL, DEVICE\_NAME);

    pr\_info("Device created on /dev/%s\n", DEVICE\_NAME);

    return SUCCESS;

}

static void \_\_exit chardev\_exit(void)

{

    device\_destroy(cls, MKDEV(major, 0));

    class\_destroy(cls);

    /\* Unregister the device \*/

    unregister\_chrdev(major, DEVICE\_NAME);

}

/\* Methods \*/

/\* Called when a process tries to open the device file, like

 \* "sudo cat /dev/chardev"

\*/

static int device\_open(struct inode \*inode, struct file \*file)

{

    static int counter = 0;

    if (atomic\_cmpxchg(&already\_open, CDEV\_NOT\_USED, CDEV\_EXCLUSIVE\_OPEN))

        return -EBUSY;

    sprintf(msg, "I already told you %d times Hello Hou!\n", counter++);

    try\_module\_get(THIS\_MODULE);

    return SUCCESS;

}

/\* Called when a process closes the device file. \*/

static int device\_release(struct inode \*inode, struct file \*file)

{

/\* We're now ready for our next caller \*/

    atomic\_set(&already\_open, CDEV\_NOT\_USED);

/\* Decrement the usage count, or else once you opened the file, you will

\* never get rid of the module.

\*/

    module\_put(THIS\_MODULE);

    return SUCCESS;

}

 /\* Called when a process, which already opened the dev file, attempts to

 \* read from it.

 \*/

static ssize\_t device\_read(struct file \*filp, /\* see include/linux/fs.h \*/

    char \_\_user \*buffer, /\* buffer to fill with data \*/

    size\_t length, /\* length of the buffer \*/

    loff\_t \*offset)

{

/\* Number of bytes actually written to the buffer \*/

    int bytes\_read = 0;

    const char \*msg\_ptr = msg;

    if (!\*(msg\_ptr + \*offset)) { /\* we are at the end of message \*/

        \*offset = 0; /\* reset the offset \*/

        return 0; /\* signify end of file \*/

    }

    msg\_ptr += \*offset;

    /\* Actually put the data into the buffer \*/

    while (length && \*msg\_ptr) {

    /\* The buffer is in the user data segment, not the kernel

    \* segment so "\*" assignment won't work. We have to use

    \* put\_user which copies data from the kernel data segment to

    \* the user data segment.

    \*/

    put\_user(\*(msg\_ptr++), buffer++);

    length--;

    bytes\_read++;

}

    \*offset += bytes\_read;

    /\* Most read functions return the number of bytes put into the buffer. \*/

    return bytes\_read;

}

    /\* Called when a process writes to dev file: echo "hi" > /dev/hello \*/

    static ssize\_t device\_write(struct file \*filp, const char \_\_user \*buff,

    size\_t len, loff\_t \*off)

    {

        pr\_alert("Sorry, this operation is not supported.\n");

        return -EINVAL;

    }

module\_init(chardev\_init);

module\_exit(chardev\_exit);

MODULE\_LICENSE("GPL");

Makefile

obj-m += chardev.o

PWD := $(CURDIR)

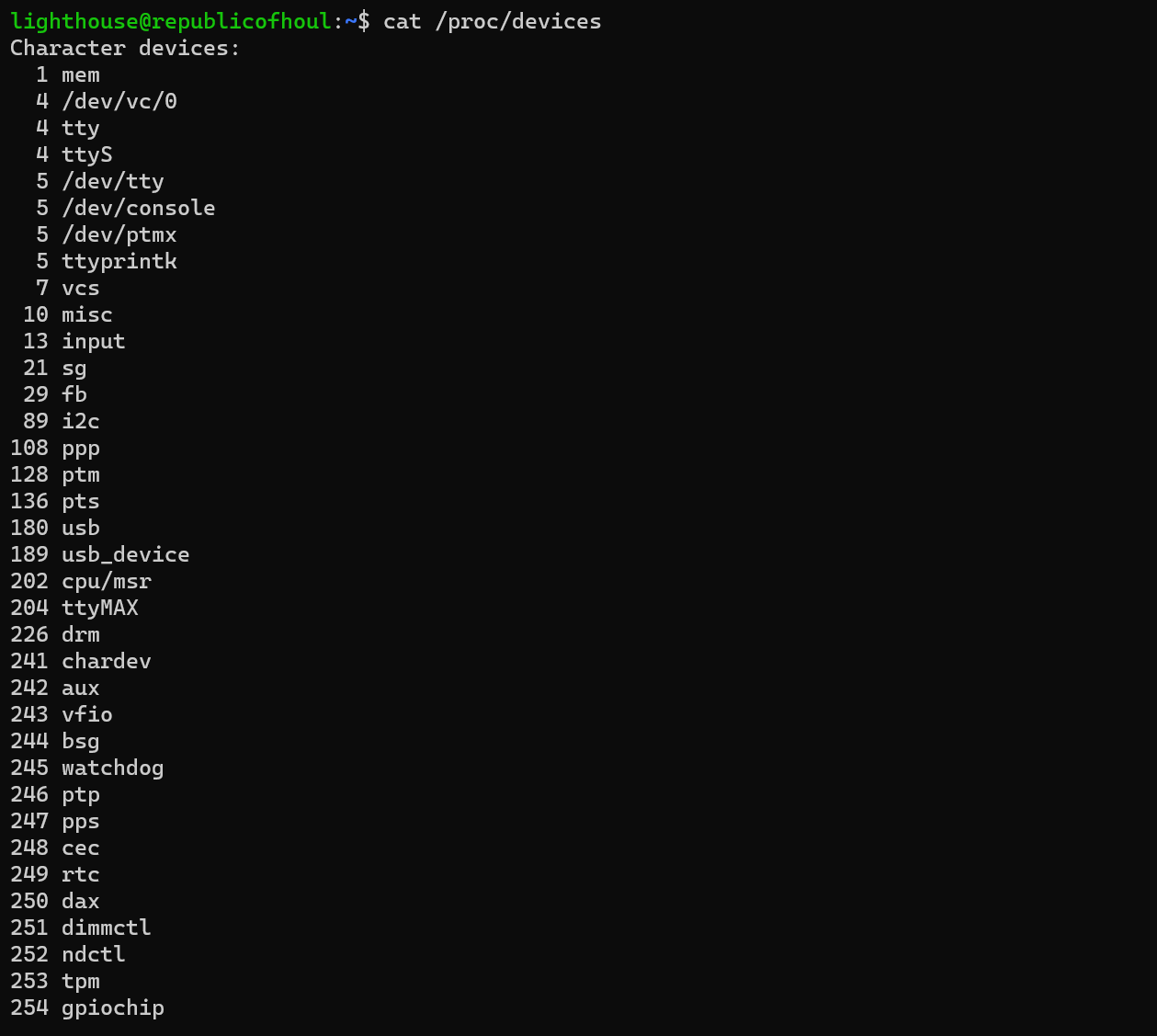
all:

    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:

    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

cat /proc/devices



dmseg output

