Project 4: `/dev/deck`

CS/COE 0449 — Introduction to System Software

See Due Date on CourseWeb

Description

Standard UNIX and Linux systems come with a few special files like `/dev/zero`, which returns nothing but zeros when it is read, and `/dev/random`, which returns random bytes. In this project, you will write a device driver to create a new device, `/dev/deck` which returns playing cards from 52 standard playing card, from the top of the deck to the bottom of the deck, one card at a time. The order of the cards can be either in an order or in random (shuffled).

How It Will Work

For this project, we will need to create two programs; (1) the device driver module (`deck_dev.c`), and (2) a test program to convince ourselves it works (`poker.c`). The test program will be a very simple implementation of the game of poker.

Driver Implementation

For this project, a card will be represented using two-byte data. The first byte is a number represents the face of the card (2, 3, ..., 13, 14) where a 2 represents a two, a 13 represents a king, and a 14 represents an ace. You get the idea right? The second byte is a character represents a suit of a card (e.g., `'S'` for spade, `'H'` for heart, `'D'` for diamond, and `'C'` for club). Our device driver will be a character device that will implement two functions as follows:

- `read` function (which is the implementation of the read() syscall) and returns an appropriate two-byte representing a playing card (2, 3, ..., 13, or 14 for the first byte and `'S'`, `'H'`, `'D'`, or `'C'` for the second byte). Note that the order of the card should start from the top of the deck down to the bottom (depending on how cards are rearranged in the deck). Note that all cards are unique inside a deck of card.
- `write` function (which is the implementation of the write() syscall). This allows user to specify how playing card should be arranged in the deck.
  - If user sends a value 0 to the deck, the deck should reset itself and rearrange all 52 cards as 2S, 3S, ..., 13S, 14S, 2H, 3H, ..., 13H, 14H, 2D, 3D, ..., 13D, 14D, 2C, 3C, ..., 13C, 14C where 2S is at the top of the deck.

– If user sends a value 1 to the deck, the deck should reset itself as in previous case and then shuffle. As we discussed in class, the kernel does not have the full C Standard library available to it and so we need to get use a different function to get random numbers. By including `<linux/random.h>` we can use the function get_random_bytes() as follows:

```
unsigned char c;
get_random_byte(&c, 1);
c = c % max;
```

where `max` is the maximum number (exclusive).

Poker Implementation

Your poker program should consists of two parts:

- Testing part: The purpose of this part is to check the order of the cards on a deck. For this part, your program sends 0 to `deck` and obtains all 52 cards and display them on the console screen. Then sends 1 to `deck` and obtains all 52 cards and display them again on the console screen. The output of this part should look some what like the following:

```
Cards in order
 2S  3S  4S  5S  6S  7S  8S  9S 10S 11S 12S 13S 14S
 2H  3H  4H  5H  6H  7H  8H  9H 10H 11H 12H 13H 14H
 2D  3D  4D  5D  6D  7D  8D  9D 10D 11D 12D 13D 14D
 2C  3C  4C  5C  6C  7C  8C  9C 10C 11C 12C 13C 14C
Cards after reset and shuffled
10D 14D 12H  9S  2H  3C  5D 12C  7S  9D 12S  4H 10H
 8H  4C  7H  4S  3S 11C 10S 13D  4D 13H 13C 14H  8S
 6H  9C  2C 14C  8D  7D  3H  2S 11H 11S  5S 12D  2D
 6D 14S  6C  9H  7C  3D 11D  5H 13S  8C  5C  6S 10C
```

- Playing part: For simplicity, this will be one player game. No opponent. For this part, your program should perform the following steps:

  1. Reset the deck and shuffle (send 1 to `deck`).
  2. Obtain the first 5 cards and display them on the console screen
  3. Ask user which card(s) user wants to change (1 to 5 separate by a space or 0 for no change). Note that user can change no card or all the way to all 5 cards.
  4. Deal another x cards (depending on the number of cards user wants to change) and display user's hand on the console screen.
  5. Sort the card by ascending order (first by face then by suit) and display user's hand on the console screen.
  6. States what kind of hand user has (only the highest one). In poker there are 9 hands from highest to lowest: straight flush, four of a kind, full house, flush, straight, three of a kind, two pair, one pair, and high card. See `http://en.wikipedia.org/wiki/List_of_poker_hands` for more detail.
  7. Go back to step 1.

Example of Game Play

The following is an output that your `poker` program should resemble:

```
Cards in order
 2S  3S  4S  5S  6S  7S  8S  9S 10S 11S 12S 13S 14S
 2H  3H  4H  5H  6H  7H  8H  9H 10H 11H 12H 13H 14H
 2D  3D  4D  5D  6D  7D  8D  9D 10D 11D 12D 13D 14D
 2C  3C  4C  5C  6C  7C  8C  9C 10C 11C 12C 13C 14C
Cards after reset and shuffled
10D 14D 12H  9S  2H  3C  5D 12C  7S  9D 12S  4H 10H
 8H  4C  7H  4S  3S 11C 10S 13D  4D 13H 13C 14H  8S
 6H  9C  2C 14C  8D  7D  3H  2S 11H 11S  5S 12D  2D
 6D 14S  6C  9H  7C  3D 11D  5H 13S  8C  5C  6S 10C

Start Playing...

Your hand:
14H 12D  4S  7S 14S
Select cards to be changed: 2 3 4
Your hand:
14H  7D  7H  5H 14S
Your sorted hand:
 5H  7D  7H 14H 14S
You got Two Pair

Would you like to play again (y/n): y

Your hand:
 8H  7C  4H  6D 12C
Select cards to be changed: 5
Your hand:
 8H  7C  4H  6D  5S
Your sorted hand:
 4H  5S  6D  7C  8H
You got Straight

Would you like to play again (y/n): n
```

Hint

It is a good idea to learn from example. First, you should try to make sure that you are able to compile and run the example device driver that a tester program. The example file `adderExample.tar.gz` can be found in the directory `/afs/cs.pitt.edu/usr0/tkosiyat/public/cs0449/project4`.

Compile the Example on `thot` and Run on Virtual Machine

Copy the example file to one of your directory and extract it. You will find a new directory called `adderExample`. Inside that directory, there are three files `adder_dev.c`, `Makefile`, and `adderTester.c`.

1. Open the `Makefile` and change the line

   ```
   KDIR := /lib/modules/$(shell uname -r)/build
   ```

   to

   ```
   KDIR := /u/SysLab/shared/linux-2.6.23.1
   ```

2. Compile the module using the command

   ```
   make ARCH=i386
   ```

   Note that the `ARCH=i386` is important because we are building a 32-bit kernel on a 64-bit machine.

3. Compile the `adderTester.c` file using the command

   ```
   gcc -m32 -o adderTester adderTester.c -static
   ```

4. At this point, you should have a kernel object file `adder_dev.ko` and an executable file `adderTester`.

5. Use `qemu` to boot up your virtual machine `tty.qcow2`. When Linux boots under `qemu`, login using the root/root account (username/password).

6. Download the kernel module `adder_dev.ko` and the tester file `adderTester` you just built into the virtual machine using the `scp` command.

7. Insert the module into the kernel using the command

   ```
   insmod ./adder_dev.ko
   ```

8. Use the command `lsmod` to verify that it is successfully inserted

9. We need to make the device file in `/dev`. First, we need to find the MAJOR and MINOR numbers that identify the new device. Use the command

   ```
   cat /sys/class/misc/adder/dev
   ```

   The output should be a number like `10:63`. The 10 is the MAJOR and the 63 is the MINOR.

10. Use `mknod` to make a special file. The name will be `adder` and it is a character device. The 10 and the 63 correspond to the MAJOR and MINOR numbers we discovered above (if different, use the ones you saw.)

```
cd /dev
mknod adder c 10 63
```

11. You should see `adder` in `/dev` directory.
12. Go back to the directory where your `adderTester` is located and run.
13. To remove the module, use the following commands:

```
rm /dev/adder
rmmod adder_dev
```

What to turn in

- The module file (`deck_dev.c`)
- The `Makefile` for your `deck` device driver
- The poker file (`poker.c`)
- Any documentation you provide to help us grade your project
- All in a tar.gz file, named with your user id (`USERNAME_project4.tar.gz`)
- Copy your archive to the appropriate directory:

```
/afs/cs.pitt.edu/public/incoming/CS0449/tkosiyat/sec1
```

No late submission will be accepted.