Department of Computer Engineering

BLG 413E
System Programming
Project 1 Report

Muhammed Raşit EROL 150150023
Samet Pilav 150160806
Joshgun Rzabayli 150160901

Spring 2019

REPORT

Introduction:

In this project, we are required to implement a character device driver that will play the board game “Master Mind”. In order to implement this project, we modified scull device to achieve goals these are stated in project file. We changed mainly device layout, write operation, read operation and ioctl function.

Implementation:

Firstly, we changed the “ioctl” file in order to define our commands as below (see Figure 1). In here, in order to start new game it is needed to pass argument to the function. Hence, we use write operation on this ioctl function. Others perform its duty and return -1, if function fails.

```
#define MASTERMIND_IOC_MAGIC  'k'
#define MASTERMIND_MMIND_REMAINING  _IO(MASTERMIND_IOC_MAGIC,  0)
#define MASTERMIND_MMIND_ENDGAME    _IO(MASTERMIND_IOC_MAGIC,   1)
#define MASTERMIND_MMIND_NEWGAME    _IOW(MASTERMIND_IOC_MAGIC,  2, char *)
#define MASTERMIND_IOC_MAXNR 2
```

Figure 1 : mastermind_ioctl.h

After defining commands, we implemented these commands in main file as follow (see Figure 2). In this function remaining part return remaining guess number using our device data structure. End game function calls trim function and it deletes all values in the device. The new game function clears line number for new game and checks given parameter in order get proper argument to the game. For example, 9999 and 999a are not proper parameters, so the function returns -1.

```
306     switch(cmd) {
307         case MASTERMIND_MMIND_REMAINING:
308             if (! capable (CAP_SYS_ADMIN)){
309                 return -EPERM;
310             }
311             return mmind_max_guesses - dev->line_number;
312             break;
313         case MASTERMIND_MMIND_ENDGAME:
314             if (! capable (CAP_SYS_ADMIN)){
315                 return -EPERM;
316             }
317             mastermind_trim(mastermind_devices);
318             break;
319         case MASTERMIND_MMIND_NEWGAME:
320 
321             if (! capable (CAP_SYS_ADMIN)){
322                 return -EPERM;
323             }
324 
325             dev->line_number_each_game = 0;
326 
327             if(strlen((char *)arg) != 4){
328                 printk(KERN_ALERT "Invalid magic number! The Number must be 4 digit!\n");
329                 return -EINVAL;
330             }
331 
332             for(i = 0 ; i < 4; i++){
333                 if( *((char *)arg + i) - '0' < 0 || *((char *)arg + i) - '0' > 9){
334                     printk(KERN_ALERT "Invalid magic number! The Number must be digit!\n");
335                     return -EINVAL;
336                 }
337             }
338             strncpy(dev->mmind_number,  (char *)arg, 4);
339 
340             break;
341         default:  /* redundant, as cmd was checked against MAXNR */
342             return -ENOTTY;
343     }
344     return retval;
345 }
```

Figure 2 : mastermind_ioctl function

The Player A has to start the game by defining secret number as in format: “mmind_number=“4283”” module parameter. The Player B can guess the number as echo “1234” format. The report will be stored in an internal buffer in the form “xxxx m+ n-abcd\n”. For this purpose, we implement structure to hold guess, number of in place digits, number of out of place digits and number of guesses. The line structure as below (see Figure 3):

```
struct line
{
    char *guess;
    int number_of_in_place_digits;
    int number_of_out_of_place_digits;
    int number_of_guesses;
    int line_order;
};
```

Figure 3 : line struct

The device structure is below (see Figure 4). In here, mmind_number represent magic number. line_number variable is used for total number of guesses in order to check game is finished when 256 line is reached. line_number_each_game is used for number of guesses for each game. If new game is started, the variable is assigned to zero as in the ioctl new game function.

```
struct mastermind_dev
{
    struct line **data;
    char *mmind_number;
    int line_number; // total line number
    int line_number_each_game; //current line number each game
    struct semaphore sem;
    struct cdev cdev;
};
```

Figure 4: mastermind_dev struct

The write and read functions are implemented in order perform game requirements. In then write function, mmind_max_guesses is used for check maximum number of guesses for each game and also restriction of 256 line is checked with related conditions. Also, input validation is performed in the write function. For each write operation data structure allocate new memory location and used it in order store given parameter during the game flow. The m and n variables described in the project pdf are calculated in the write function.

In the read function, proper conditions are created in order to perform correct reading. The data is read from data structure written in the write function into local buffer. After that, data is sent to read function buffer from local buffer at the end of the function.

In the cleanup function all reserved memory locations is returned back to operating system. Also, in the init module function, major number is taken from system and device is created with default variables.

The test programs written for ioctl functions are also attached the zip file. Some of the program screenshots are added the following part.

Program Flow:

In order to active device, it is needed to become root using sudo su command. After that insmod command device is created. In here it is possible to give two parameters to the program. One is mmind_number which is used for store magic number and its default value is 4283 and mmind_max_guesses which is used for maximum number of guesses each game and its default value is 10. The major number which is given us from system can be checked using grep mastermind /proc/devices command. After that using mknod node is created (see Figure 5).

```
root@mrerol-virtual-machine: /home/mrerol/Desktop/project1
File  Edit  Tabs  Help
mrerol@mrerol-virtual-machine:~/Desktop/project1$ sudo su
[sudo] password for mrerol:
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# insmod mastermind.ko
mmind_number="9999" mmind_max_guesses=10
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# grep mastermind /proc/devices
250 mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# mknod /dev/mastermind c 250 0
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1#
```

Figure 5 : installation of device

```
root@mrerol-virtual-machine: /home/mrerol/Desktop/project1
File  Edit  Tabs  Help
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "12345" > /dev/mastermind
bash: echo: write error: Invalid argument
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234a" > /dev/mastermind
bash: echo: write error: Invalid argument
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "123" > /dev/mastermind
bash: echo: write error: Invalid argument
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# cat /dev/mastermind
9999 4+ 0- 0001
6789 1+ 3- 0002
1234 0+ 0- 0003
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# echo "1234" > /dev/mastermind
bash: echo: write error: Cannot allocate memory
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1# tail /dev/mastermind
9999 4+ 0- 0001
6789 1+ 3- 0002
1234 0+ 0- 0003
1234 0+ 0- 0004
1234 0+ 0- 0005
1234 0+ 0- 0006
1234 0+ 0- 0007
1234 0+ 0- 0008
1234 0+ 0- 0009
1234 0+ 0- 0010
root@mrerol-virtual-machine:/home/mrerol/Desktop/project1#
```

Figure 6: example of program work with writing and reading

In the Figure 6, some of the read and write operations can be seen. Also, some input validation error messages can be seen in there. In the following figures test functions of ioct can be seen (see Figure 7-9).

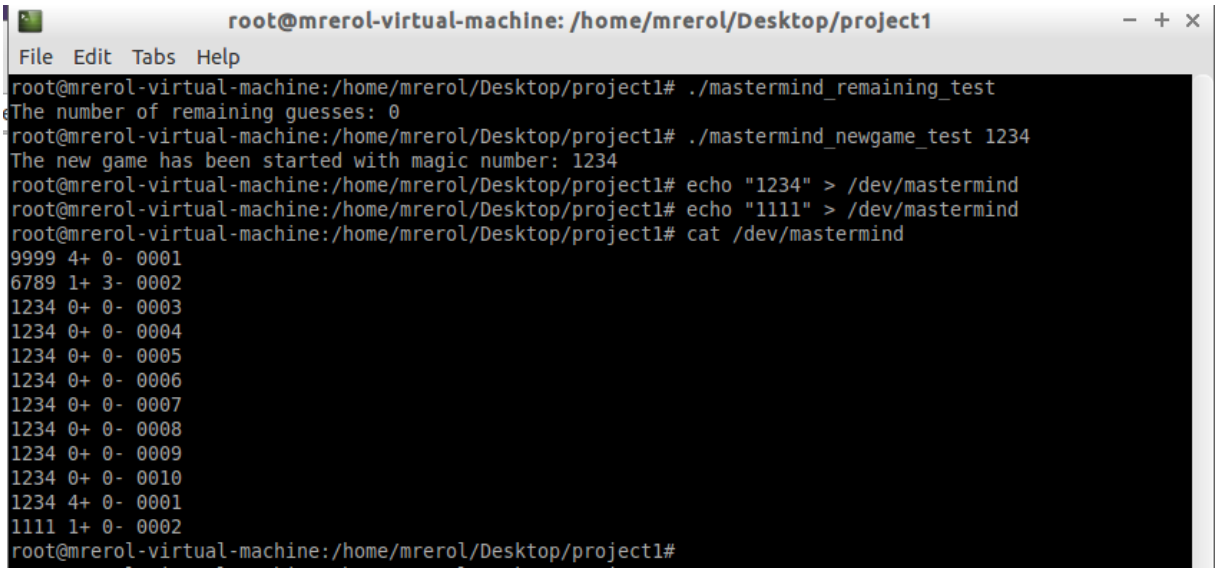

Figure 7: new game and remaining ioctl functions example

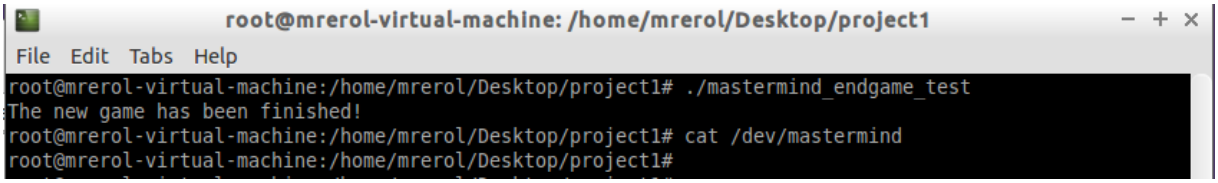

Figure 8: end game test example

In the end of the program, using rmmod and rm -r commands device can be deleted from system (see Figure 9).

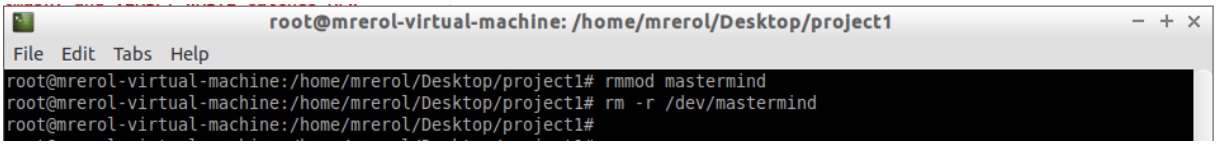

Figure 9: cleaning up the device

In the zip file all source codes and object files of test programs are added. Also, using makefile is attached to the zip file. The program is tested in the Lubuntu 32-bit 14.04 version required in the software_installation.pdf file.