BLG 413E System Programming

Project 2 Report

(Device Driver)

Group 47

Kağan Özgün - 150130055

Mehmet Taha Çorbacıoğlu – 150130017

05/12/2017

Contents

1 Objective of the Project

The objective of the project is developing a device driver for build a message box between users on system. Write operation done on device (/dev/messagebox). Messages start with "@USERNAME" that specify the message receiver. User can only see the message addressed to him/her. Device driver messages cab write with echo method and can read with cat command.

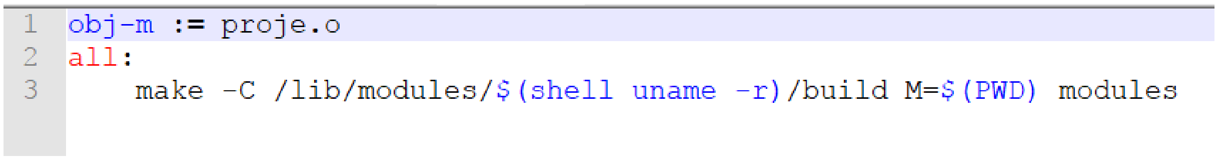
2 Method

In this project we firstly write a function for taking user name from linux passwd file. For taking user from messages we parse the messages and differentiate the user and message that way we build the basic of the messagebox.

2.1 Source Code

We take the scull example from lecture source and build project on scull example project.

2.1.1 Make File for Module



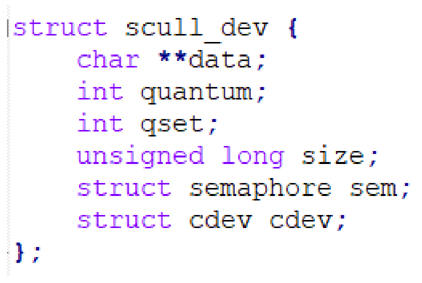
```
obj-m := proje.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
```

Figure 1 Makefile file for module

For adding module to linux kernel Makefile used to do settings.

2.1.2 Data Structure for Device

Struct include char pointer to take data, quantum and size for driver and semaphore for operations.

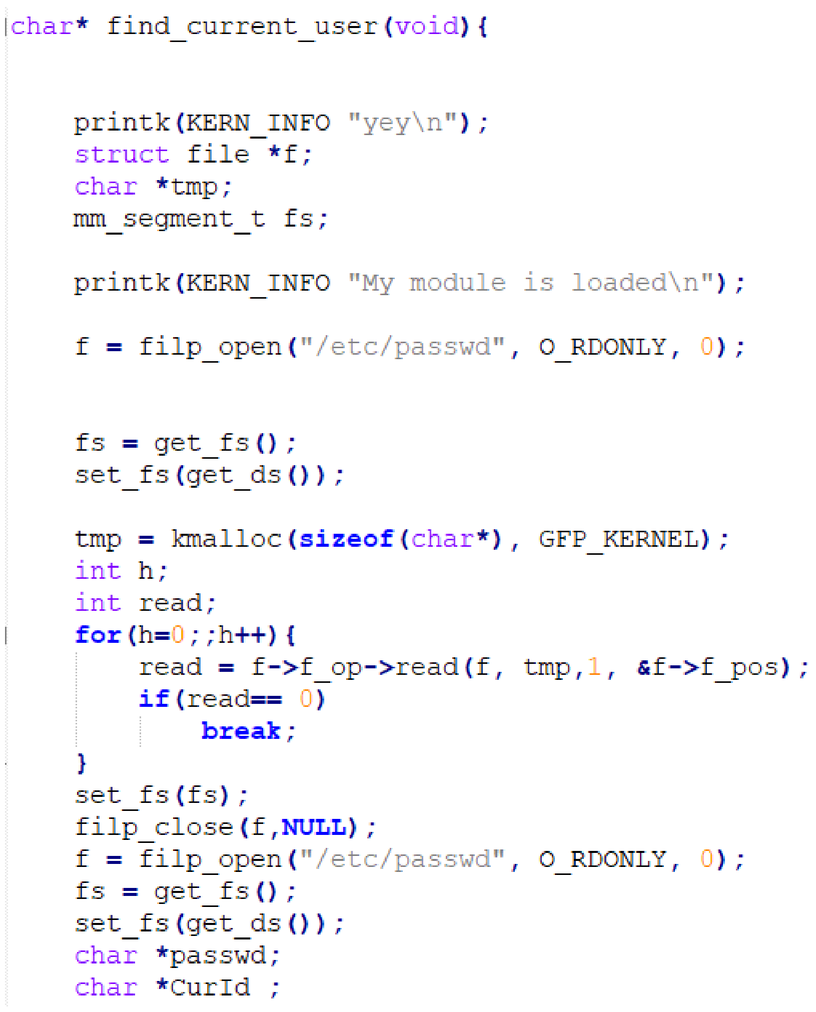


```
struct scull_dev {
    char **data;
    int quantum;
    int qset;
    unsigned long size;
    struct semaphore sem;
    struct cdev cdev;
};
```

Figure 2 Struct for manupilating data

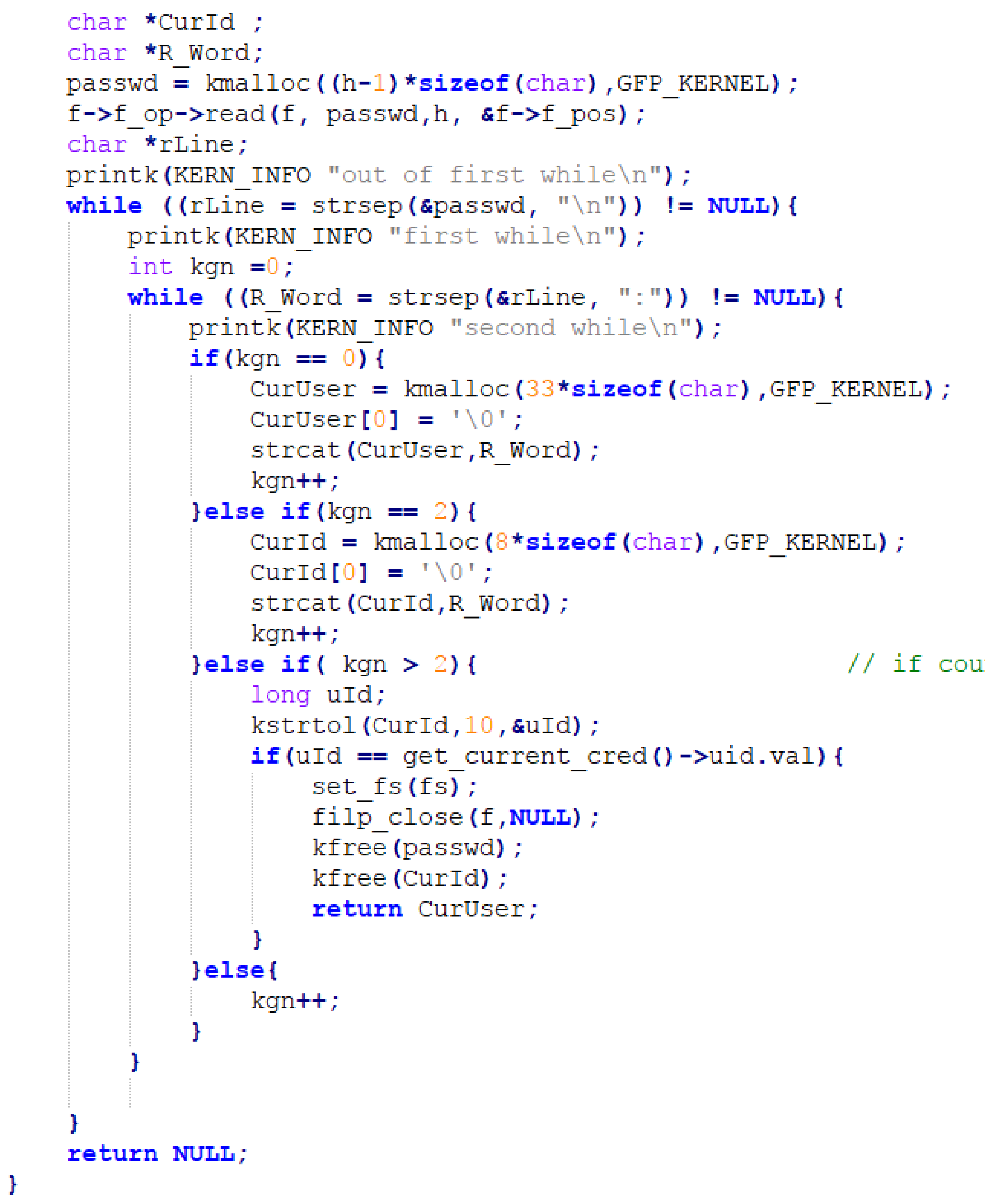
2.1.3 Find Current User Function

We implement function for taking all user from linux passwd file. Function read /etc/passwd file and parse the file string to take user id, username information and compare current user id with users in the file if current user find in file function return the username of the current user else return NULL.



```
char* find_current_user(void){


    printk(KERN_INFO "yey\n");
    struct file *f;
    char *tmp;
    mm_segment_t fs;

    printk(KERN_INFO "My module is loaded\n");

    f = filp_open("/etc/passwd", O_RDONLY, 0);


    fs = get_fs();
    set_fs(get_ds());

    tmp = kmalloc(sizeof(char*), GFP_KERNEL);
    int h;
    int read;
    for(h=0;;h++){
        read = f->f_op->read(f, tmp,1, &f->f_pos);
        if(read== 0)
            break;
    }
    set_fs(fs);
    filp_close(f,NULL);
    f = filp_open("/etc/passwd", O_RDONLY, 0);
    fs = get_fs();
    set_fs(get_ds());
    char *passwd;
    char *CurId ;
```

Figure 3 Find current username function part1

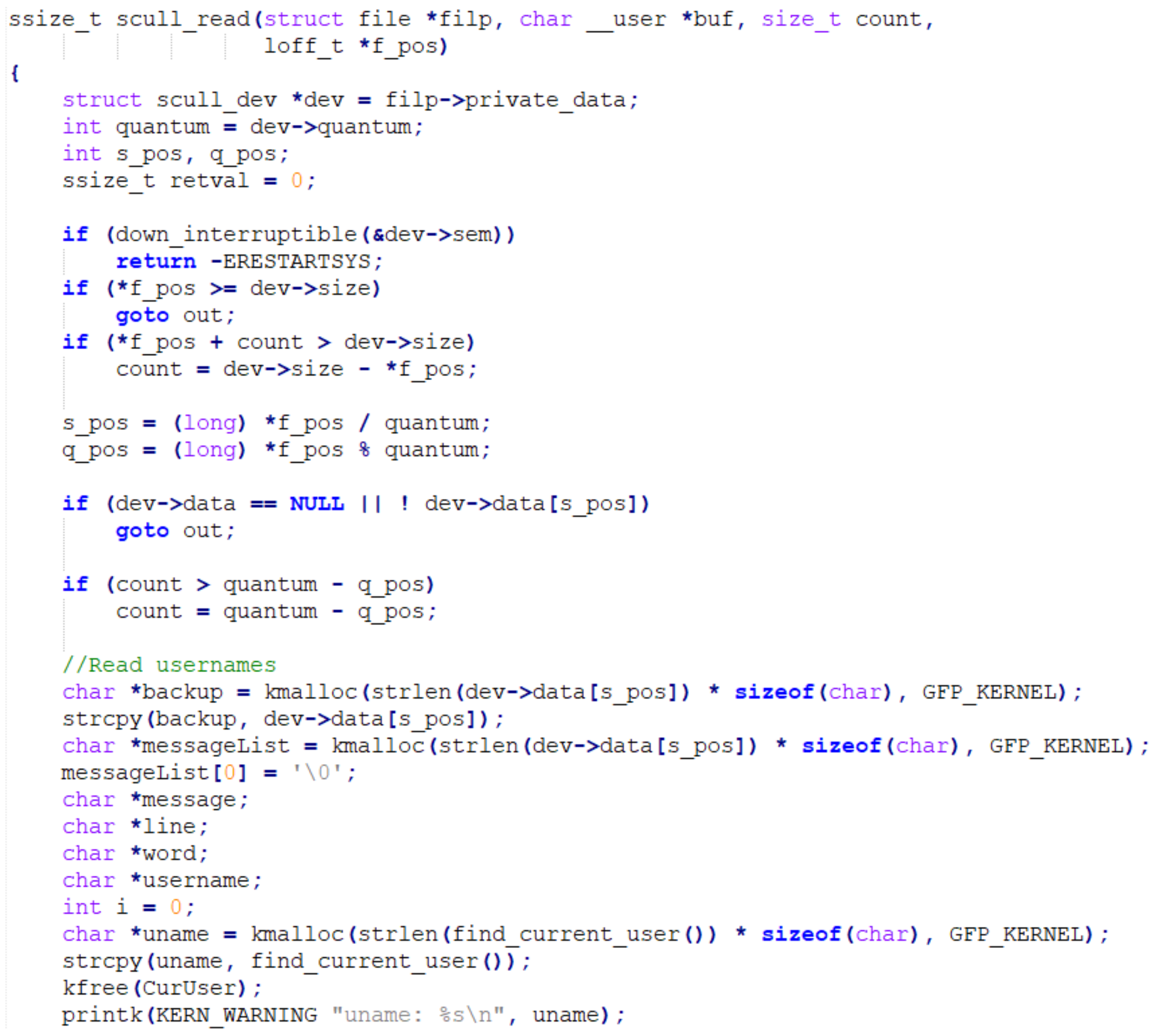


```
    char *CurId ;
    char *R_Word;
    passwd = kmalloc((h-1)*sizeof(char),GFP_KERNEL);
    f->f_op->read(f, passwd,h, &f->f_pos);
    char *rLine;
    printk(KERN_INFO "out of first while\n");
    while ((rLine = strsep(&passwd, "\n")) != NULL){
        printk(KERN_INFO "first while\n");
        int kgn =0;
        while ((R_Word = strsep(&rLine, ":")) != NULL){
            printk(KERN_INFO "second while\n");
            if(kgn == 0){
                CurUser = kmalloc(33*sizeof(char),GFP_KERNEL);
                CurUser[0] = '\0';
                strcat(CurUser,R_Word);
                kgn++;
            }else if(kgn == 2){
                CurId = kmalloc(8*sizeof(char),GFP_KERNEL);
                CurId[0] = '\0';
                strcat(CurId,R_Word);
                kgn++;
            }else if( kgn > 2){                         // if cou
                long uId;
                kstrtol(CurId,10,&uId);
                if(uId == get_current_cred()->uid.val){
                    set_fs(fs);
                    filp_close(f,NULL);
                    kfree(passwd);
                    kfree(CurId);
                    return CurUser;
                }
            }else{
                kgn++;
            }
        }

    }
    return NULL;
}
```

Figure 4 Find current username function part2

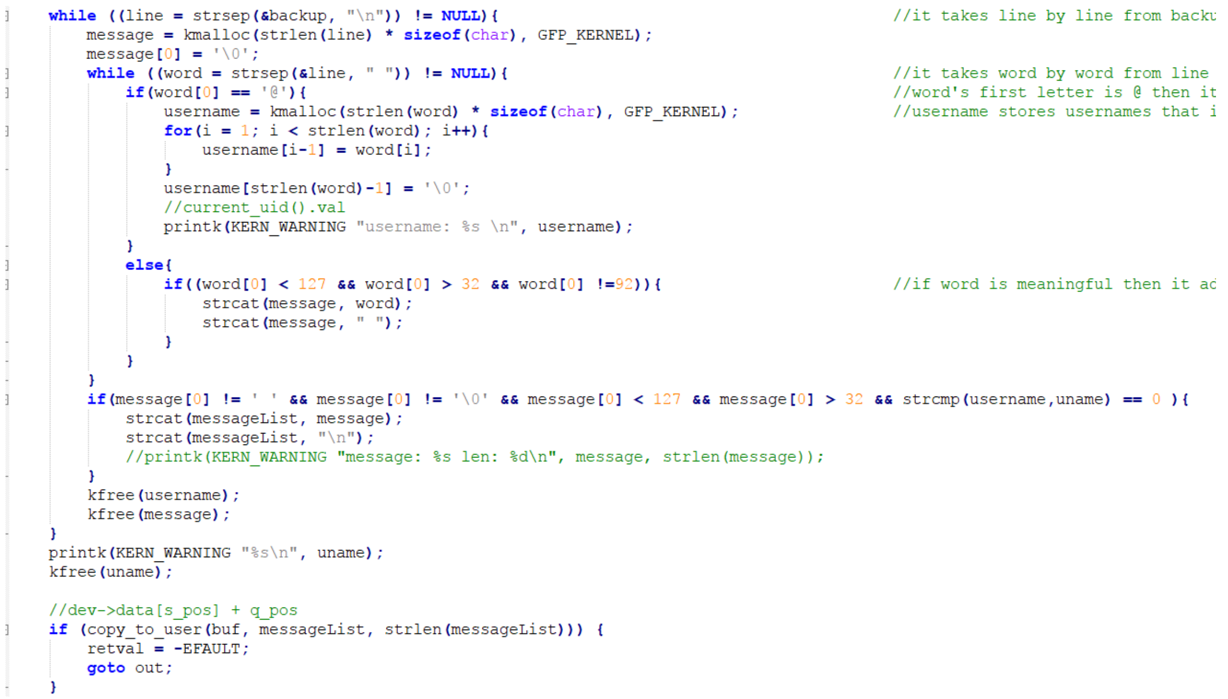
2.1.4 Read Function for Driver

Read function takes messages from driver and parse the messages to separate target user and the message content. Function takes current username from find_current_user function and write the messages of the current user.



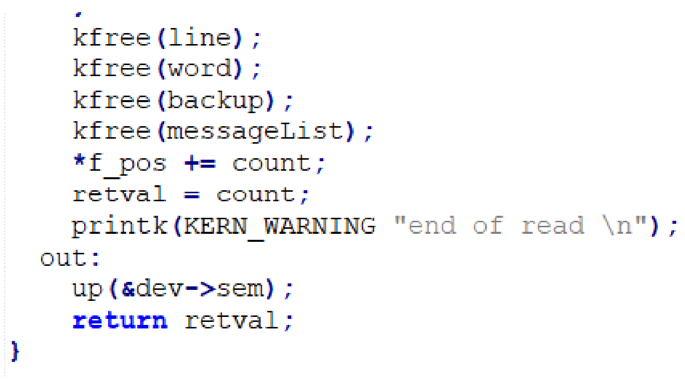
```c
ssize_t scull_read(struct file *filp, char __user *buf, size_t count,
                   loff_t *f_pos)
{
    struct scull_dev *dev = filp->private_data;
    int quantum = dev->quantum;
    int s_pos, q_pos;
    ssize_t retval = 0;

    if (down_interruptible(&dev->sem))
        return -ERESTARTSYS;
    if (*f_pos >= dev->size)
        goto out;
    if (*f_pos + count > dev->size)
        count = dev->size - *f_pos;

    s_pos = (long) *f_pos / quantum;
    q_pos = (long) *f_pos % quantum;

    if (dev->data == NULL || ! dev->data[s_pos])
        goto out;

    if (count > quantum - q_pos)
        count = quantum - q_pos;

    //Read usernames
    char *backup = kmalloc(strlen(dev->data[s_pos]) * sizeof(char), GFP_KERNEL);
    strcpy(backup, dev->data[s_pos]);
    char *messageList = kmalloc(strlen(dev->data[s_pos]) * sizeof(char), GFP_KERNEL);
    messageList[0] = '\0';
    char *message;
    char *line;
    char *word;
    char *username;
    int i = 0;
    char *uname = kmalloc(strlen(find_current_user()) * sizeof(char), GFP_KERNEL);
    strcpy(uname, find_current_user());
    kfree(CurUser);
    printk(KERN_WARNING "uname: %s\n", uname);
```

Figure 5 Read Function part1



```c
    while ((line = strsep(&backup, "\n")) != NULL){                       //it takes line by line from backu
        message = kmalloc(strlen(line) * sizeof(char), GFP_KERNEL);
        message[0] = '\0';
        while ((word = strsep(&line, " ")) != NULL){                      //it takes word by word from line
            if(word[0] == '@'){                                           //word's first letter is @ then it
                username = kmalloc(strlen(word) * sizeof(char), GFP_KERNEL); //username stores usernames that i
                for(i = 1; i < strlen(word); i++){
                    username[i-1] = word[i];
                }
                username[strlen(word)-1] = '\0';
                //current_uid().val
                printk(KERN_WARNING "username: %s \n", username);
            }
            else{
                if((word[0] < 127 && word[0] > 32 && word[0] !=92)){      //if word is meaningful then it ac
                    strcat(message, word);
                    strcat(message, " ");
                }
            }
        }
        if(message[0] != ' ' && message[0] != '\0' && message[0] < 127 && message[0] > 32 && strcmp(username,uname) == 0 ){
            strcat(messageList, message);
            strcat(messageList, "\n");
            //printk(KERN_WARNING "message: %s len: %d\n", message, strlen(message));
        }
        kfree(username);
        kfree(message);
    }
    printk(KERN_WARNING "%s\n", uname);
    kfree(uname);

    //dev->data[s_pos] + q_pos
    if (copy_to_user(buf, messageList, strlen(messageList))) {
        retval = -EFAULT;
        goto out;
    }
```

Figure 6 Read function part2

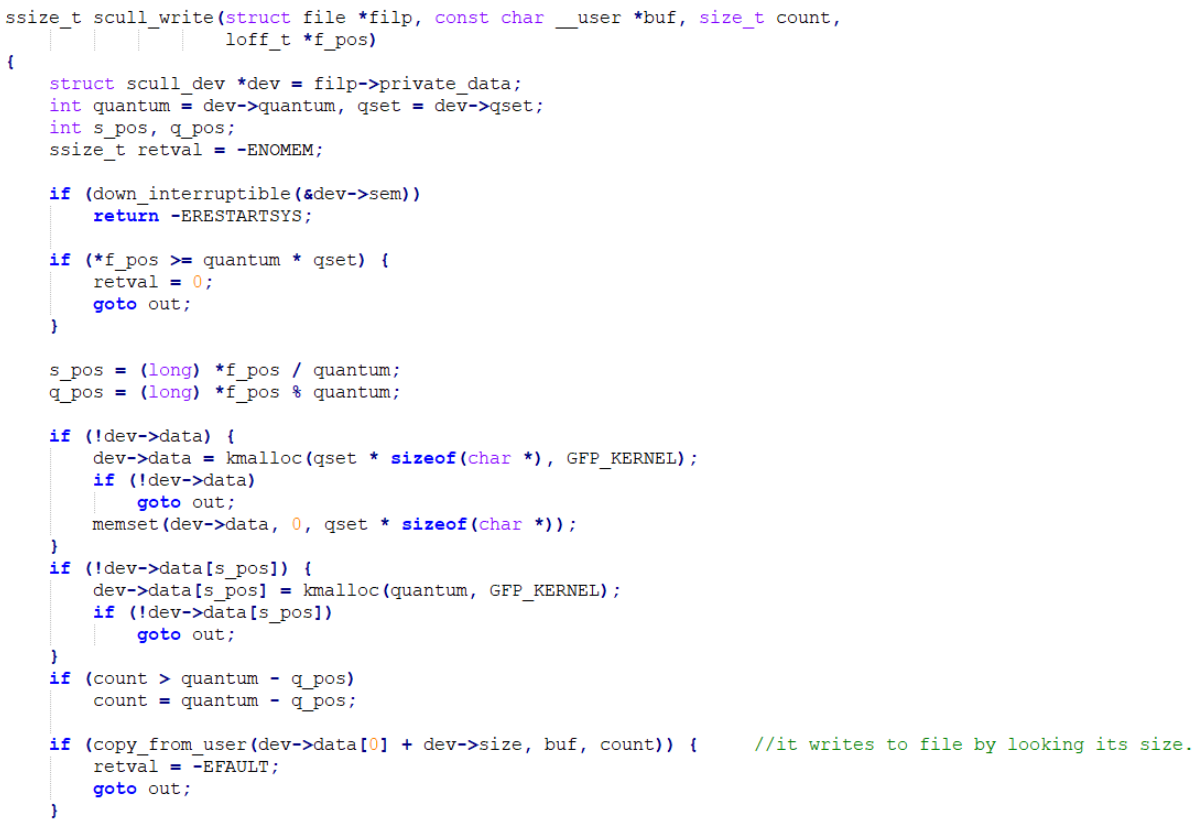


```c
    kfree(line);
    kfree(word);
    kfree(backup);
    kfree(messageList);
    *f_pos += count;
    retval = count;
    printk(KERN_WARNING "end of read \n");
  out:
    up(&dev->sem);
    return retval;
}
```

Figure 7 Read Function part3 free the memory

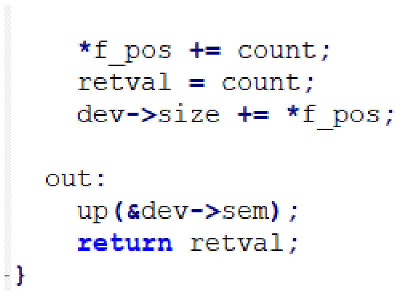
2.1.5 Write Function

We use the scull project write function and only change the function for writing messages to target user.



```c
ssize_t scull_write(struct file *filp, const char __user *buf, size_t count,
                    loff_t *f_pos)
{
    struct scull_dev *dev = filp->private_data;
    int quantum = dev->quantum, qset = dev->qset;
    int s_pos, q_pos;
    ssize_t retval = -ENOMEM;

    if (down_interruptible(&dev->sem))
        return -ERESTARTSYS;

    if (*f_pos >= quantum * qset) {
        retval = 0;
        goto out;
    }

    s_pos = (long) *f_pos / quantum;
    q_pos = (long) *f_pos % quantum;

    if (!dev->data) {
        dev->data = kmalloc(qset * sizeof(char *), GFP_KERNEL);
        if (!dev->data)
            goto out;
        memset(dev->data, 0, qset * sizeof(char *));
    }
    if (!dev->data[s_pos]) {
        dev->data[s_pos] = kmalloc(quantum, GFP_KERNEL);
        if (!dev->data[s_pos])
            goto out;
    }
    if (count > quantum - q_pos)
        count = quantum - q_pos;

    if (copy_from_user(dev->data[0] + dev->size, buf, count)) {     //it writes to file by looking its size.
        retval = -EFAULT;
        goto out;
    }
```

Figure 8 Write function part1

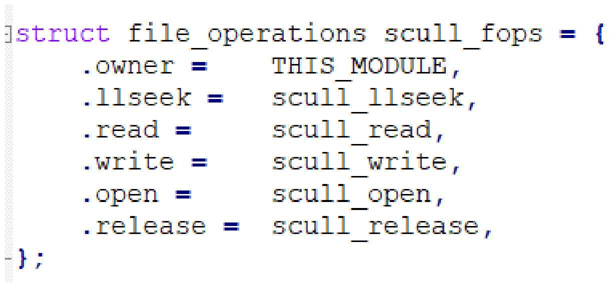


```
    *f_pos += count;
    retval = count;
    dev->size += *f_pos;

  out:
    up(&dev->sem);
    return retval;
}
```

Figure 9 Write function part2

2.1.6 File Operations Definitions

Ioctl functions definition for driver. We only use the read, write, open and release.



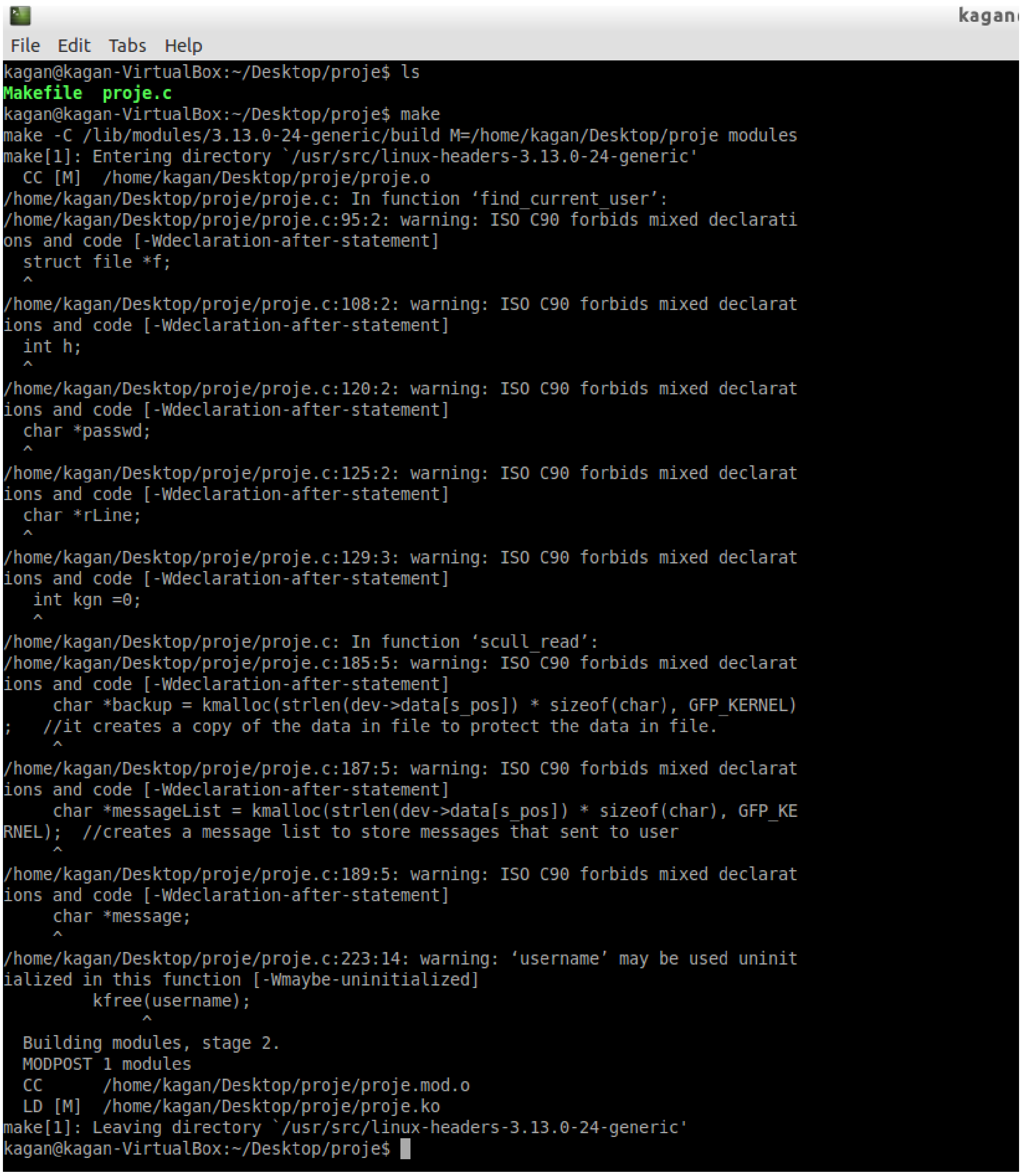
```
struct file_operations scull_fops = {
     .owner =    THIS_MODULE,
     .llseek =   scull_llseek,
     .read =     scull_read,
     .write =    scull_write,
     .open =     scull_open,
     .release =  scull_release,
};
```

Figure 10 File operations definition for driver

2.2 Test of the Driver

2.2.1 Compiling Source Code

For compiling the source code we use the make command in directory.

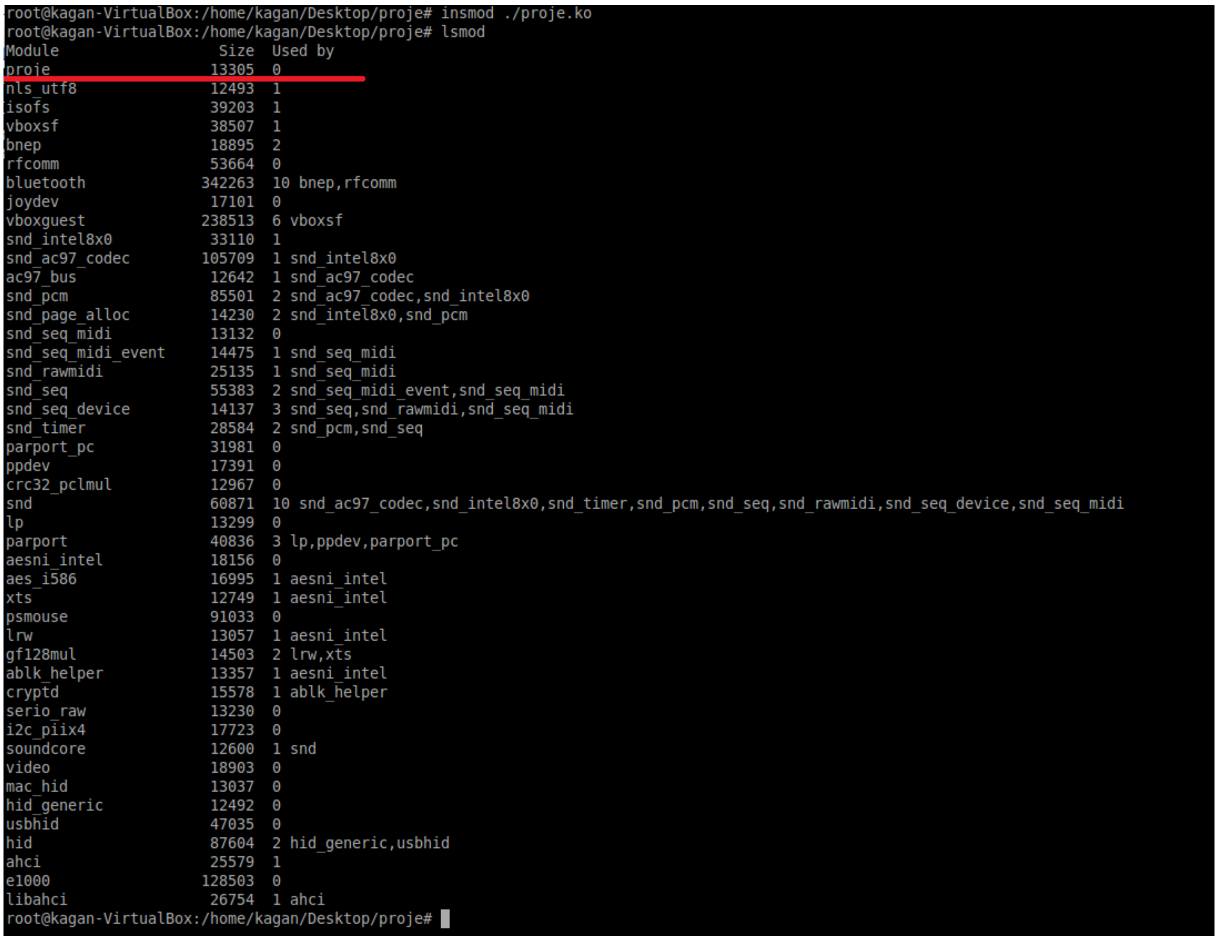


```
kagan@kagan-VirtualBox:~/Desktop/proje$ ls
Makefile  proje.c
kagan@kagan-VirtualBox:~/Desktop/proje$ make
make -C /lib/modules/3.13.0-24-generic/build M=/home/kagan/Desktop/proje modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-24-generic'
  CC [M]  /home/kagan/Desktop/proje/proje.o
/home/kagan/Desktop/proje/proje.c: In function ‘find_current_user’:
/home/kagan/Desktop/proje/proje.c:95:2: warning: ISO C90 forbids mixed declarati
ons and code [-Wdeclaration-after-statement]
  struct file *f;
  ^
/home/kagan/Desktop/proje/proje.c:108:2: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
  int h;
  ^
/home/kagan/Desktop/proje/proje.c:120:2: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
  char *passwd;
  ^
/home/kagan/Desktop/proje/proje.c:125:2: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
  char *rLine;
  ^
/home/kagan/Desktop/proje/proje.c:129:3: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
   int kgn =0;
   ^
/home/kagan/Desktop/proje/proje.c: In function ‘scull_read’:
/home/kagan/Desktop/proje/proje.c:185:5: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
     char *backup = kmalloc(strlen(dev->data[s_pos]) * sizeof(char), GFP_KERNEL)
;   //it creates a copy of the data in file to protect the data in file.
     ^
/home/kagan/Desktop/proje/proje.c:187:5: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
     char *messageList = kmalloc(strlen(dev->data[s_pos]) * sizeof(char), GFP_KE
RNEL);  //creates a message list to store messages that sent to user
     ^
/home/kagan/Desktop/proje/proje.c:189:5: warning: ISO C90 forbids mixed declarat
ions and code [-Wdeclaration-after-statement]
     char *message;
     ^
/home/kagan/Desktop/proje/proje.c:223:14: warning: ‘username’ may be used uninit
ialized in this function [-Wmaybe-uninitialized]
         kfree(username);
              ^
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/kagan/Desktop/proje/proje.mod.o
  LD [M]  /home/kagan/Desktop/proje/proje.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-24-generic'
kagan@kagan-VirtualBox:~/Desktop/proje$ 
```

Figure 11 Compiling code with Make command

2.2.2 Loading Module

For the install module to linux kernel we use the insmod command and list modules with lsmod.

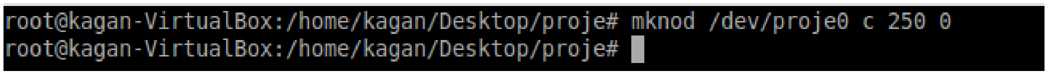


```
root@kagan-VirtualBox:/home/kagan/Desktop/proje# insmod ./proje.ko
root@kagan-VirtualBox:/home/kagan/Desktop/proje# lsmod
Module                  Size  Used by
proje                  13305  0
nls_utf8               12493  1
isofs                  39203  1
vboxsf                 38507  1
bnep                   18895  2
rfcomm                 53664  0
bluetooth             342263  10 bnep,rfcomm
joydev                 17101  0
vboxguest             238513  6 vboxsf
snd_intel8x0           33110  1
snd_ac97_codec        105709  1 snd_intel8x0
ac97_bus               12642  1 snd_ac97_codec
snd_pcm                85501  2 snd_ac97_codec,snd_intel8x0
snd_page_alloc         14230  2 snd_intel8x0,snd_pcm
snd_seq_midi           13132  0
snd_seq_midi_event     14475  1 snd_seq_midi
snd_rawmidi            25135  1 snd_seq_midi
snd_seq                55383  2 snd_seq_midi_event,snd_seq_midi
snd_seq_device         14137  3 snd_seq,snd_rawmidi,snd_seq_midi
snd_timer              28584  2 snd_pcm,snd_seq
parport_pc             31981  0
ppdev                  17391  0
crc32_pclmul           12967  0
snd                    60871  10 snd_ac97_codec,snd_intel8x0,snd_timer,snd_pcm,snd_seq,snd_rawmidi,snd_seq_device,snd_seq_midi
lp                     13299  0
parport                40836  3 lp,ppdev,parport_pc
aesni_intel            18156  0
aes_i586               16995  1 aesni_intel
xts                    12749  1 aesni_intel
psmouse                91033  0
lrw                    13057  1 aesni_intel
gf128mul               14503  2 lrw,xts
ablk_helper            13357  1 aesni_intel
cryptd                 15578  1 ablk_helper
serio_raw              13230  0
i2c_piix4              17723  0
soundcore              12600  1 snd
video                  18903  0
mac_hid                13037  0
hid_generic            12492  0
usbhid                 47035  0
hid                    87604  2 hid_generic,usbhid
ahci                   25579  1
e1000                 128503  0
libahci                26754  1 ahci
root@kagan-VirtualBox:/home/kagan/Desktop/proje#
```

Figure 12 List of modules using lsmod command

2.2.3 Add Node to System

Add nod for reaching driver by path and give the minor and major number to device.

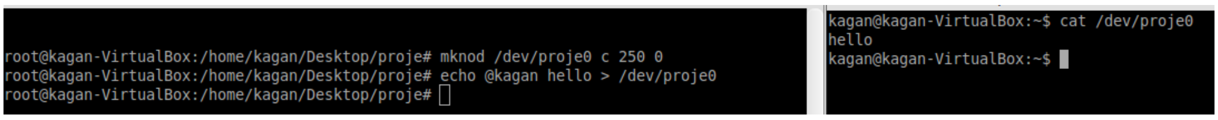


```
root@kagan-VirtualBox:/home/kagan/Desktop/proje# mknod /dev/proje0 c 250 0
root@kagan-VirtualBox:/home/kagan/Desktop/proje#
```

Figure 13 mknod command for adding driver node

2.2.4 Test1

Root user send message to user kagan and user kagan receive the message.



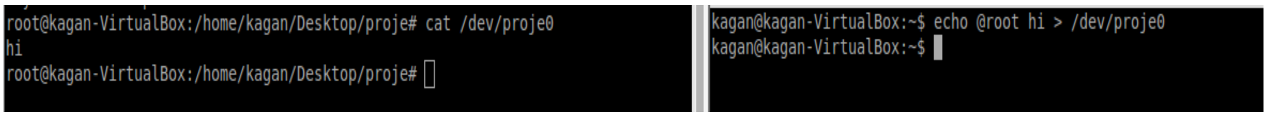
```
root@kagan-VirtualBox:/home/kagan/Desktop/proje# mknod /dev/proje0 c 250 0
root@kagan-VirtualBox:/home/kagan/Desktop/proje# echo @kagan hello > /dev/proje0
root@kagan-VirtualBox:/home/kagan/Desktop/proje#
```

```
kagan@kagan-VirtualBox:~$ cat /dev/proje0
hello
kagan@kagan-VirtualBox:~$
```

Figure 14 Test 1 Send message Root to kagan

2.2.5 Test2

User kagan send message to root user and root read the messages.



```
root@kagan-VirtualBox:/home/kagan/Desktop/proje# cat /dev/proje0
hi
root@kagan-VirtualBox:/home/kagan/Desktop/proje#
```

```
kagan@kagan-VirtualBox:~$ echo @root hi > /dev/proje0
kagan@kagan-VirtualBox:~$
```

Figure 15 Test2 send message kagan to root

3 Conclusion

We spent a lot of time for getting username from user id part of the project and last few days we have some memory errors and segmentation faults and we cannot implement delete functions for messages and ioctl operations of the project. We only implement message sending and message receiving part of the project.