Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа №4.2

Группа: ИУ7-64Б

Студент: Садулаева Т. Р.

Оценка (баллы): _____

Преподаватель: Рязанова Н.Ю.

Москва, 2021

Задание

Написать программу – загружаемый модуль ядра (LKM) – которая поддерживает чтение из пространства пользователя и запись в пространство пользователя из пространства ядра.

После загрузки модуля пользователь должен иметь возможность загружать в него строки с помощью команды echo, а затем считывать их с помощью команды cat.

Содержимое файла md.c:

```
#include<linux/module.h>
#include<linux/proc_fs.h>
#include<linux/slab.h>
#include<linux/vmalloc.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Sadulaeva Teona");

#define COOKIE_BUF_SIZE PAGE_SIZE
#define TEMP_BUF_SIZE 256

ssize_t fortune_read(struct file *file, char *buf, size_t count, loff_t
*f_pos); ssize_t fortune_write(struct file *file, const char *buf, size_t
count, loff_t *f_pos); int fortune_release(struct inode *node, struct file
*file);
int fortune_open(struct inode *node, struct file *file);

int fortune_init(void);
void
fortune_exit(void);

static const struct proc_ops pops = {
   .proc_read = fortune_read,
   .proc_write = fortune_write,
   .proc_release = fortune_release,
   .proc_open = fortune_open,
};

static char *cookie_buf;
static struct proc_dir_entry
*proc_entry; static unsigned
read_index;
static unsigned write_index;
```

```
char temp[TEMP_BUF_SIZE]; struct

task_struct *task = &init_task; int

len, t_len;

int fortune_init(void)
{
  cookie_buf = (char *) vmalloc(COOKIE_BUF_SIZE); if
  (!cookie_buf)
  {
    printk(KERN_INFO "not enough memory\n");
    return -ENOMEM;
  }
  memset(cookie_buf, 0, COOKIE_BUF_SIZE);
  proc_entry = proc_create("fortune", 0666, NULL, &pops);
  if (!proc_entry)
  {
    vfree(cookie_buf);
    printk(KERN_INFO "Couldn't create proc entry\n");
    return -ENOMEM;
  }
  read_index = 0;
  write_index = 0;
  proc_mkdir("my_dir_fortune", NULL);
  proc_symlink("my_symbolic_fortune", NULL, "/proc/fortune");
  printk(KERN_INFO "fortune module loaded.\n");
  return 0;
}


ssize_t fortune_read(struct file *file, char *buf, size_t count, loff_t *f_pos)
{
  if (*f_pos > 0 || write_index ==
    0) return 0;

  if (read_index >=
    write_index) read_index = 0;

  len = copy_to_user(buf, "%s\n", &cookie_buf[read_index]);
  read_index += len;
  *f_pos += len;


  printk(KERN_INFO "proc called read \n"); return
  len;
}


int fortune_release(struct inode *node, struct file *file)
{
  printk(KERN_INFO "proc called release \n");
```

```
  return 0;
}

int fortune_open(struct inode *node, struct file *file)
{
  printk(KERN_INFO "proc called open \n"); return
  0;
}

ssize_t fortune_write(struct file *file, const char *buf, size_t count, loff_t *f_pos)
{
  int space_available = (COOKIE_BUF_SIZE - write_index) + 1;

  if (count > space_available)
  {
    printk(KERN_INFO "+_+ buf is full\n");
    return -ENOSPC;
  }

  if (copy_from_user(&cookie_buf[write_index], buf, count))
    return -EFAULT;

  write_index += count;
  cookie_buf[write_index - 1] = 0;
  printk(KERN_INFO "proc called write \n");
  return count;
}

void fortune_exit(void)
{
  remove_proc_entry("fortune", NULL);

  if (cookie_buf)
    vfree(cookie_buf);

  printk(KERN_INFO "fortune module unloaded.\n");
}

module_init(fortune_init);
module_exit(fortune_exit);
```

Загрузка модуля – проверка загрузки – демонстрация работы загружаемого модуля – выгрузка модуля:

```
          VirtualBox:~/Downloads/lab4_os_part2$ sudo insmod md.ko
          VirtualBox:~/Downloads/lab4_os_part2$ lsmod | grep md
md                     16384  0
crypto_simd            16384  1 aesni_intel
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
          VirtualBox:~/Downloads/lab4_os_part2$ echo "here we go again" > pro
c/fortune
bash: proc/fortune: No such file or directory
          VirtualBox:~/Downloads/lab4_os_part2$ echo "here we go again" > /pr
oc/fortune
          VirtualBox:~/Downloads/lab4_os_part2$ cat /proc/fortune
here we go again
```

Содержимое файла /var/log/syslog:

```
[ 5995.617453] fortune module loaded.
[ 6151.947252] proc called open
[ 6151.947288] proc called write
[ 6151.947293] proc called release
[ 6252.730481] proc called open
[ 6252.730507] proc called read
[ 6252.730533] proc called release
[ 6412.091555] fortune module unloaded.
```

Вывод содержимого /proc (с созданной директорией -- my_dir_fortune и символической ссылкой -- my_symbolic_fortune):

```
          VirtualBox:~/Downloads/lab4_os_part2$ ls -a /proc
.       20421  3444  5349  5957  80          bus             mounts
..      20473  3574  5353  5971  804         cgroups         mtrr
1       20779  369   5355  599   819         cmdline         my_dir_fortune
10      20861  3755  5401  5991  82          consoles        my_symbolic_fortune
1016    20873  4     5528  6     821         cpuinfo         net
1021    20882  4060  5542  6009  84          crypto          pagetypeinfo
1033    20894  4362  5555  601   85          devices         partitions
105     21     4364  5584  603   864         diskstats       pressure
108     21162  4410  5613  604   87          dma             sched_debug
109     21235  4413  5619  605   870         driver          schedstat
11      21293  4418  563   618   88          dynamic_debug   scsi
114     21294  4422  564   620   880         execdomains     self
115     215    4428  5832  628   89          fb              slabinfo
12      2179   4429  5845  634   8902        filesystems     softirqs
13      21872  4439  5901  6366  892         fs              stat
14      21873  4454  5905  637   895         interrupts      swaps
15      22     4470  5907  639   9           iomem           sys
154     23     4476  5908  640   90          ioports         sysrq-trigger
155     24     4486  5915  653   9058        irq             sysvipc
```

РЕАЛИЗАЦИЯ ПРИ ПОМОЩИ SEQUENCE

Содержимое файла fortune_seq.c:

```
#include<linux/module.h>
#include<linux/proc_fs.h>
#include<linux/slab.h>
#include<linux/vmalloc.h>
#include<linux/seq_file.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Sadulaeva Teona");

static char *str = NULL;

unsigned int
write_index; unsigned int
read_index;

#define COOKIE_POT_SIZE PAGE_SIZE

static struct proc_dir_entry *proc_entry, *proc_dir, *proc_sym;

ssize_t my_write(struct file* file, const char ____user *buffer, size_t count, loff_t
*f_pos); int my_release(struct inode *node, struct file *file);
int my_open(struct inode *inode, struct file *file);

static const struct proc_ops pops = {
   .proc_read = seq_read,
   .proc_write = my_write,
   .proc_release = my_release,
   .proc_open = my_open,
};

static int ____init fortune_init(void)
{
   printk(KERN_INFO "fortune init\n");
   write_index = 0;
   read_index = 0;
   str = vmalloc(COOKIE_POT_SIZE); if
   (!str)
   {
      printk(KERN_INFO "Error: can't malloc cookie buffer\n");
      return -ENOMEM;
   }
   memset(str, 0, COOKIE_POT_SIZE);
   proc_entry = proc_create("fortune", S_IRUGO | S_IWUGO, NULL, &pops);
   if(!proc_entry)
   {
      vfree(str);
      printk(KERN_INFO "Error: can't create fortune
      file\n"); return -ENOMEM;
```

```
  }
  proc_dir = proc_mkdir("fortune_dir", NULL);
  if(!proc_dir)
  {
    vfree(str);
    printk(KERN_INFO "Error: can't create fortune
    file\n"); return -ENOMEM;
  }
  proc_sym = proc_symlink("fortune_symlink", NULL, "/proc/fortune");
  if(!proc_sym)
  {
    vfree(str);
    printk(KERN_INFO "Error: can't create fortune
    file\n"); return -ENOMEM;
  }
  printk(KERN_INFO "fortune: Fortune module loaded successfully\n");
  return 0;
}



static int my_show(struct seq_file *m, void *v)
{
  printk(KERN_INFO "fortune called my_show\n");
  seq_printf(m, "%s", str + read_index);
  int len = strlen(str + read_index);
  if (len)
    read_index += len + 1;
  return 0;
}

ssize_t my_write(struct file* file, const char____user *buffer, size_t count, loff_t *f_pos)
{
  printk(KERN_INFO "fortune called my_write\n");
  if (copy_from_user(&str[write_index], buffer, count))
  return -EFAULT;
  write_index  +=  count;
  str[write_index-1]   =
  0; return count;
}



int my_open(struct inode *inode, struct file *file)
{
  printk(KERN_INFO "fortune called my_open\n");
  return single_open(file, my_show, NULL);
}



int my_release(struct inode *inode, struct file *file)
{
```

```
  printk(KERN_INFO "fortune called my_release\n");
  return single_release(inode, file);
}



static void ____exit fortune_exit(void)
{
  printk(KERN_INFO "fortune exit\n");
  if (proc_entry)
    remove_proc_entry("fortune", NULL); if
  (proc_dir)
    remove_proc_entry("fortune_dir", NULL); if
  (proc_sym)
    remove_proc_entry("fortune_symlink", NULL); if
  (str)
    vfree(str);
  printk(KERN_INFO "fortune: Fortune module unloaded\n");
}

module_init(fortune_init);
module_exit(fortune_exit);
```

Загрузка модуля

```
VirtualBox:~/Downloads/lab4_os_part2$ ls -a
.   fortune_seq.c   fortune_seq.mod  md.c   md.mod
..  fortune_seq.ko  Makefile         md.ko  Module.symvers
VirtualBox:~/Downloads/lab4_os_part2$ sudo insmod fortune_seq.ko
VirtualBox:~/Downloads/lab4_os_part2$ lsmod | grep fortune_seq
fortune_seq            16384  0
VirtualBox:~/Downloads/lab4_os_part2$ ls -a /proc
```

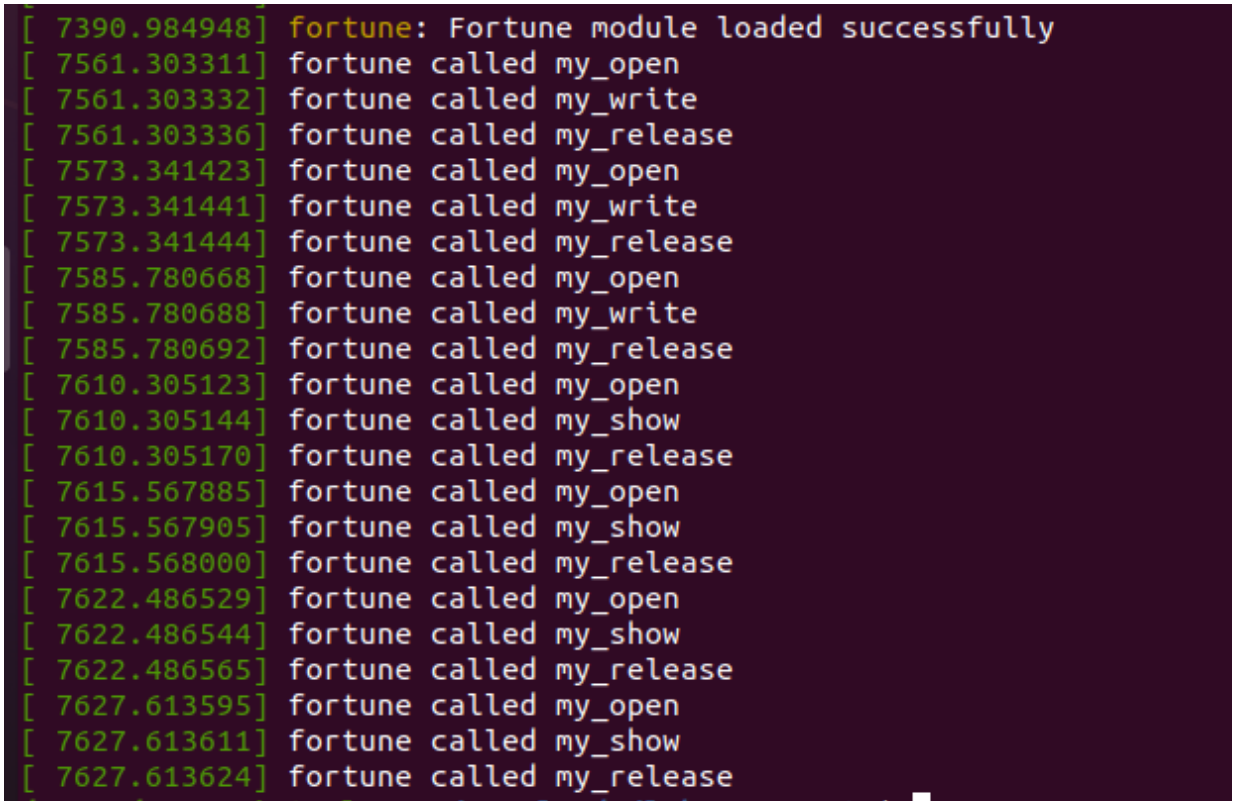
Вывод содержимого /proc, в том числе созданной директории и символической ссылки (fortune_dir и fortune_symlink):

```
.       20421  285   5331  5950  78    bootconfig
..      20473  3     5347  5955  79    buddyinfo
1       20779  3444  5349  5957  80    bus
10      20861  3574  5353  5971  804   cgroups
1016    20873  369   5355  599   819   cmdline
1021    20882  3755  5401  5991  82    consoles
1033    20894  4     5528  6     821   cpuinfo
105     21     4060  5542  6009  84    crypto
108     21235  4362  5555  601   85    devices
109     21294  4364  5584  603   864   diskstats
11      215    4410  5613  604   87    dma
114     2179   4413  5619  605   870   driver
115     21872  4418  563   618   88    dynamic_debug
12      21922  4422  564   620   880   execdomains
13      21991  4428  5832  628   89    fb
14      21992  4429  5845  634   8902  filesystems
15      21993  4439  5901  6366  892   fortune
154     22     4454  5905  637   895   fortune_dir
155     22442  4470  5907  639   9     fortune_symlink
16      23     4476  5908  640   90    fs
16882   24     4486  5915  653   9058  interrupts
17      242    4492  5917  683   91    iomem
17370   243    4498  5919  689   92    ioports
175     244    4502  5922  72    923   irq
176     25     4527  5923  724   932   kallsyms
```

Демонстрация работы загруженного модуля:

```
VirtualBox:~/Downloads/lab4_os_part2$ echo "hi!" > /proc/fortune
VirtualBox:~/Downloads/lab4_os_part2$ echo "hola!" > /proc/fortune
VirtualBox:~/Downloads/lab4_os_part2$ echo "privet!" > /proc/fortun
e
VirtualBox:~/Downloads/lab4_os_part2$ cat /proc/fortune
hi!
   VirtualBox:~/Downloads/lab4_os_part2$ cat /proc/fortune
hola!
     VirtualBox:~/Downloads/lab4_os_part2$ cat /proc/fortune
privet!
       VirtualBox:~/Downloads/lab4_os_part2$
```

Содержимое файла /var/log/syslog:


[7390.984948] fortune: Fortune module loaded successfully
[7561.303311] fortune called my_open
[7561.303332] fortune called my_write
[7561.303336] fortune called my_release
[7573.341423] fortune called my_open
[7573.341441] fortune called my_write
[7573.341444] fortune called my_release
[7585.780668] fortune called my_open
[7585.780688] fortune called my_write
[7585.780692] fortune called my_release
[7610.305123] fortune called my_open
[7610.305144] fortune called my_show
[7610.305170] fortune called my_release
[7615.567885] fortune called my_open
[7615.567905] fortune called my_show
[7615.568000] fortune called my_release
[7622.486529] fortune called my_open
[7622.486544] fortune called my_show
[7622.486565] fortune called my_release
[7627.613595] fortune called my_open
[7627.613611] fortune called my_show
[7627.613624] fortune called my_release