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»

Дисциплина: «Операционные системы»

Лабораторная работа №3

Тема работы: «Загружаемые модули ядра»

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватель: Рязанова Н. Ю.

Москва, 2020 г.

Цель работы

Знакомство с базовыми принципами разработки и взаимодействия с загружаемыми модулями ядра ОС Linux.

Задание 1

Реализовать загружаемый модуль ядра, который при загрузке записывает в системный журнал сообщение “Hello world!”, а при выгрузке “Good by”. Модуль должен собираться при помощи Make-файла. Загружаемый модуль должен содержать:

- Указание лицензии GPL
- Указание автора

Листинг кода программы

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/sched.h>

#include <linux/init_task.h>


MODULE_LICENSE("GPL");
MODULE_AUTHOR("Levushkin Ilya");

static int __init my_module_init(void)
{
    printk(KERN_INFO "Module is now loaded.\n");
    struct task_struct *task = &init_task;

    do
    {
        printk(KERN_INFO "---%s-%d, parent %s-%d", task->comm,
            task->pid, task->parent->comm, task->parent->pid);
    } while ((task = next_task(task)) != &init_task);

    printk(KERN_INFO "---%s-%d, parent %s-%d", current->comm,
        current->pid, current->parent->comm, current->parent->pid);
    return 0;
}


static void __exit my_module_cleanup(void) {
    printk(KERN_INFO "Module is now unloaded.\n");
}


module_init(my_module_init);
module_exit(my_module_cleanup);

```

Рис. 1: md.c

Демонстрация работы программы

Ниже продемонстрированы загрузка модуля ядра, вывод списка загруженных модулей ядра (команда lsmod), чье название содержит строку «md», последние 20 сообщений, выведенных модулями ядра и информация о модуле.

```
ilalevuskin@ubuntu:~/Desktop/3_os/1$ sudo insmod md.ko
ilalevuskin@ubuntu:~/Desktop/3_os/1$ lsmod | grep md
md                     16384  0
crypto_simd            16384  1 aesni_intel
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
ilalevuskin@ubuntu:~/Desktop/3_os/1$ dmesg | tail -20
[ 3653.538649] ---update-notifier-2637, parent gnome-session-b-1869
[ 3653.538650] ---gnome-software-2641, parent gnome-session-b-1869
[ 3653.538651] ---fwupd-2682, parent systemd-1
[ 3653.538652] ---zeitgeist-daemo-2877, parent systemd-1817
[ 3653.538653] ---zeitgeist-fts-2885, parent systemd-1817
[ 3653.538654] ---gnome-terminal--2929, parent systemd-1817
[ 3653.538654] ---bash-2942, parent gnome-terminal--2929
[ 3653.538655] ---nautilus-2963, parent systemd-1817
[ 3653.538656] ---deja-dup-monito-3000, parent gnome-session-b-1869
[ 3653.538657] ---kworker/u64:0-10529, parent kthreadd-2
[ 3653.538658] ---kworker/1:1-11701, parent kthreadd-2
[ 3653.538659] ---gdm-session-wor-12203, parent gdm-session-wor-1730
[ 3653.538660] ---kworker/0:1-12312, parent kthreadd-2
[ 3653.538661] ---kworker/0:2-13339, parent kthreadd-2
[ 3653.538662] ---kworker/u64:1-14528, parent kthreadd-2
[ 3653.538663] ---kworker/1:0-15161, parent kthreadd-2
[ 3653.538665] ---kworker/0:0-15360, parent kthreadd-2
[ 3653.538666] ---kworker/u64:2-15711, parent kthreadd-2
[ 3653.538667] ---sudo-17594, parent bash-2942
[ 3653.538667] ---insmod-17595, parent sudo-17594
ilalevuskin@ubuntu:~/Desktop/3_os/1$ modinfo md.ko
filename:       /home/ilalevuskin/Desktop/3_os/1/md.ko
author:         Levushkin Ilya
license:        GPL
srcversion:     6E945F0AD5935233E3033CE
depends:
retpoline:      Y
name:           md
vermagic:       5.3.0-51-generic SMP mod_unload
```

Видно, что модуль успешно загружен.

Ниже продемонстрированы выгрузка модуля ядра и последние 5 сообщений, выведенных модулями ядра.

```
ilalevuskin@ubuntu:~/Desktop/3_os/1$ sudo rmmod md
ilalevuskin@ubuntu:~/Desktop/3_os/1$ lsmod | grep md
crypto_simd            16384  1 aesni_intel
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
ilalevuskin@ubuntu:~/Desktop/3_os/1$ dmesg | tail -5
[ 3653.538666] ---kworker/u64:2-15711, parent kthreadd-2
[ 3653.538667] ---sudo-17594, parent bash-2942
[ 3653.538667] ---insmod-17595, parent sudo-17594
[ 3653.538668] ---insmod-17595, parent sudo-17594
[ 3713.004085] Module is now unloaded.
```

Видно, что модуль успешно выгружен.

Задание 2

Реализовать три загружаемых модуля ядра:

- Вызываемый модуль md1
- Вызывающий модуль md2
- «Отладочный» модуль md3

Каждый загружаемый модуль должен содержать:

- Указание лицензии GPL
- Указание автора

Загружаемые модули должны собираться при помощи Make-файла (сборка командой make). **Вызов каждой функции модуля должен сопровождаться записью в системный журнал** информации, какая функция какого модуля была вызвана.

Модуль md1

Модуль md1 демонстрирует возможность создания экспортируемых данных и функций.

Данный модуль ядра должен содержать:

- Экспортируемые строковые (char *) и численные (int) данные.

- Экспортируемые функции возвращающие строковые и числовые значения.

Например:

- Функция, возвращающая в зависимости от переданного целочисленного параметра различные строки (на усмотрение студента);
- Функция, производящая подсчет факториала переданного целочисленного параметра;
- Функция возвращающая 0;

Модуль md2

Модуль md2 демонстрирует использование данных и функций экспортируемых первым модулем (md1).

Данный модуль должен при загрузке:

- Вызывать все экспортированные модулем md1 процедуры и вывести в системный журнал возвращаемые ими значения с указанием имени вызванной процедуры.
- Вывести в системный журнал все экспортированные модулем md1 данные.

Модуль md3

Модуль md3 демонстрирует сценарий некорректного завершения установки модуля, и возможность использования загружаемого модуля в качестве функции выполняемой в пространстве ядре.

Процедура инициализации этого загружаемого модуля должна возвращать ненулевое значение и выводить в системный журнал данные и возвращаемые значения экспортированных модулем md1 процедур (аналогично md2).

Данный модуль включен в работу для проработки вопросов, связанных с отладкой модулей ядра.

Make-файл

Make-файл должен быть написан так, чтобы при вызове команды make происходила компиляция всех реализованных загружаемых модулей. Это позволит упростить процесс компиляции. Также Make-файл должен содержать правило clean для очистки директории от промежуточных файлов компиляции.

Пример Make-файла предназначенного для сборки и компиляции загружаемого модуля ядра:

```
ifneq ($(KERNELRELEASE),)
        obj-m   := md.o
else
        CURRENT = $(shell uname -r)
        KDIR = /lib/modules/$(CURRENT)/build
        PWD = $(shell pwd)
default:
        $(MAKE) -C $(KDIR) M=$(PWD) modules
clean:
        @rm -f *.o .*.cmd .*.flags *.mod.c *.order
        @rm -f .*.*.cmd *~ *.*~ TODO.*
        @rm -fR .tmp*
        @rm -rf .tmp_versions
disclean: clean
        @rm *.ko *.symvers

endif
// $(MAKE) - вызов MAKE в режиме ядра.
```

Листинг кода программы

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>


MODULE_LICENSE("GPL");
MODULE_AUTHOR("Levushkin Ilya");

char* export_char = "Hello from md1!";
int export_int = 52;


extern char* md1_str(int n)
{
    printk(KERN_INFO "md1_str() called!\n");
    switch(n)
    {
        case 1:
            return "First message.\n";
            break;
        case 2:
            return "Second message.\n";
            break;
        case 3:
            return "Please, print 1 to get first message or 2 to get second
            break;
    }
}

extern int md1_fact(int n)
{
    int i, res;
    res = 1;
    printk(KERN_INFO "md1_fact() called!\n");
    for (i = 2; i <= n; i++)
    {
        res *= i;
    }
    return res;
}
```

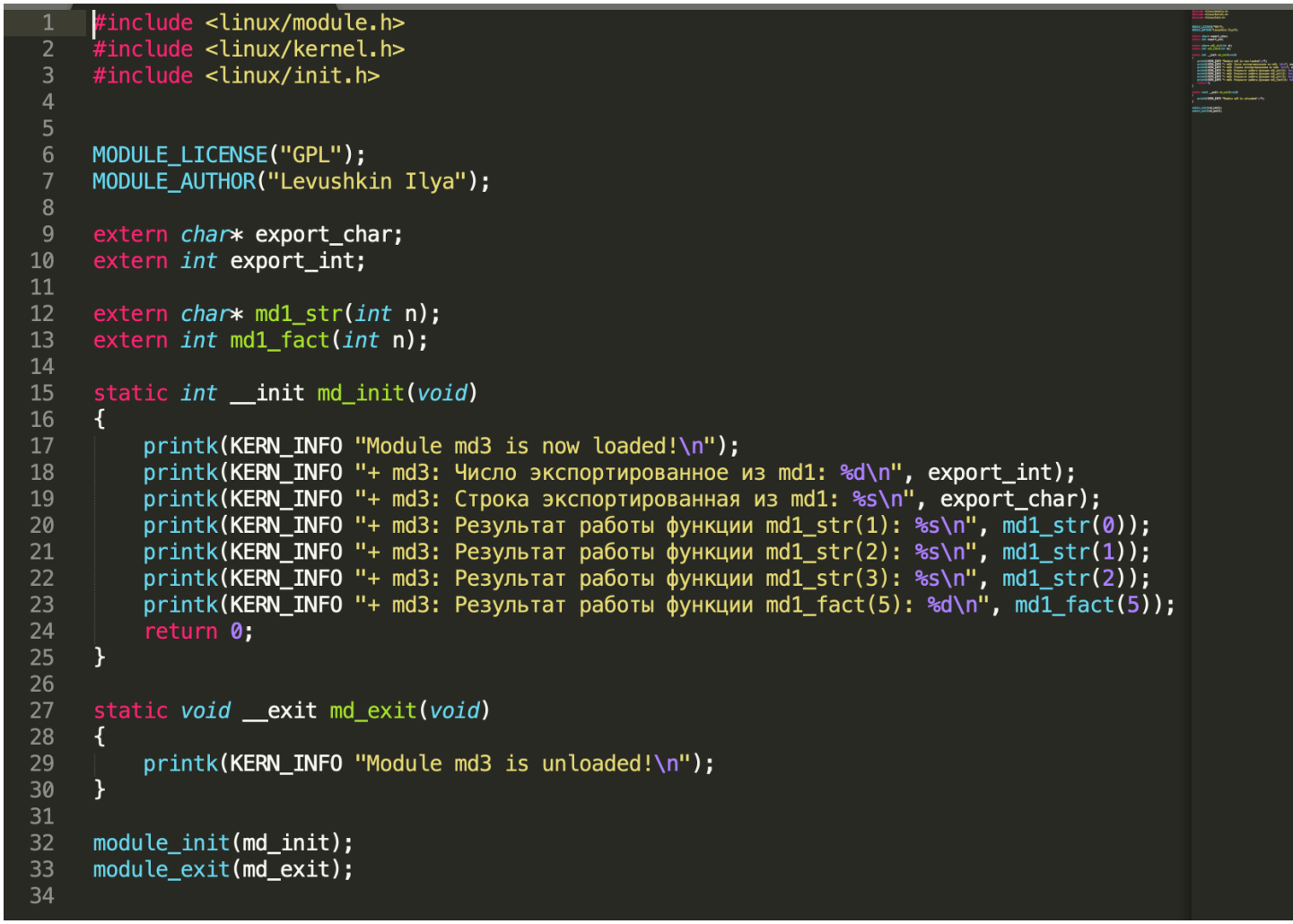
Рис. 2: md1.c

```
43
44  EXPORT_SYMBOL(md1_str);
45  EXPORT_SYMBOL(md1_fact);
46
47  EXPORT_SYMBOL(export_char);
48  EXPORT_SYMBOL(export_int);
49
50
51  static int __init md_init(void)
52  {
53      printk(KERN_INFO "Module md1 loaded!\n");
54      return 0;
55  }
56
57  static void __exit md_exit(void)
58  {
59      printk(KERN_INFO "Module md1 unloaded!\n");
60  }
61
62  module_init(md_init);
63  module_exit(md_exit);
```

Рис. 3: md1.c

```
1   #include <linux/module.h>
2   #include <linux/kernel.h>
3   #include <linux/init.h>
4
5
6   MODULE_LICENSE("GPL");
7   MODULE_AUTHOR("Levushkin Ilya");
8
9   extern char* export_char;
10  extern int export_int;
11
12  extern char* md1_str(int n);
13  extern int md1_fact(int n);
14
15  static int __init md_init(void)
16  {
17      printk(KERN_INFO "Module md2 is now loaded!\n");
18      printk(KERN_INFO "+ md2: Число экспортированное из md1: %d\n", export_int);
19      printk(KERN_INFO "+ md2: Строка экспортированная из md1: %s\n", export_char);
20      printk(KERN_INFO "+ md2: Результат работы функции md1_str(1): %s\n", md1_str(1));
21      printk(KERN_INFO "+ md2: Результат работы функции md1_str(2): %s\n", md1_str(2));
22      printk(KERN_INFO "+ md2: Результат работы функции md1_str(3): %s\n", md1_str(3));
23      printk(KERN_INFO "+ md2: Результат работы функции md1_fact(5): %d\n", md1_fact(5));
24      return 0;
25  }
26
27  static void __exit md_exit(void)
28  {
29      printk(KERN_INFO "Module md2 is unloaded!\n");
30  }
31
32  module_init(md_init);
33  module_exit(md_exit);
34
```

Рис. 4: md2.c



```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>


MODULE_LICENSE("GPL");
MODULE_AUTHOR("Levushkin Ilya");

extern char* export_char;
extern int export_int;

extern char* md1_str(int n);
extern int md1_fact(int n);

static int __init md_init(void)
{
    printk(KERN_INFO "Module md3 is now loaded!\n");
    printk(KERN_INFO "+ md3: Число экспортированное из md1: %d\n", export_int);
    printk(KERN_INFO "+ md3: Строка экспортированная из md1: %s\n", export_char);
    printk(KERN_INFO "+ md3: Результат работы функции md1_str(1): %s\n", md1_str(0));
    printk(KERN_INFO "+ md3: Результат работы функции md1_str(2): %s\n", md1_str(1));
    printk(KERN_INFO "+ md3: Результат работы функции md1_str(3): %s\n", md1_str(2));
    printk(KERN_INFO "+ md3: Результат работы функции md1_fact(5): %d\n", md1_fact(5));
    return 0;
}

static void __exit md_exit(void)
{
    printk(KERN_INFO "Module md3 is unloaded!\n");
}

module_init(md_init);
module_exit(md_exit);

```

Рис. 5: md3.c

Демонстрация работы программы

Ниже продемонстрированы загрузка вначале модуля ядра «md1» и вывод списка загруженных модулей ядра (команда lsmod), чье название содержит строку «md1», затем модуля ядра «md2» и вывод списка загруженных модулей ядра (команда lsmod), чье название содержит строку «md2», и модуля ядра «md3» и вывод списка загруженных модулей ядра (команда lsmod), чье название содержит строку «md3».

```
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo insmod md1.ko
ilalevuskin@ubuntu:~/Desktop/3_os/2$ dmesg | grep md1
[   50.786575] Module md1 loaded!
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo insmod md2.ko
ilalevuskin@ubuntu:~/Desktop/3_os/2$ dmesg | grep md2
[   66.279872] Module md2 is now loaded!
[   66.279874] + md2: Число экспортированное из md1: 52
[   66.279878] + md2: Строка экспортированная из md1: Hello from md1!
[   66.279880] + md2: Результат работы функции md1_str(1): First message.
[   66.279880] + md2: Результат работы функции md1_str(2): Second message.
[   66.279881] + md2: Результат работы функции md1_str(3): Please, print 1 to ge
t first message or 2 to get second.
[   66.279882] + md2: Результат работы функции md1_fact(5): 120
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo insmod md3.ko
ilalevuskin@ubuntu:~/Desktop/3_os/2$ dmesg | grep md3
[   72.481241] Module md3 is now loaded!
[   72.481243] + md3: Число экспортированное из md1: 52
[   72.481244] + md3: Строка экспортированная из md1: Hello from md1!
[   72.481244] + md3: Результат работы функции md1_str(1): First message.
[   72.481245] + md3: Результат работы функции md1_str(2): Second message.
[   72.481246] + md3: Результат работы функции md1_str(3): Please, print 1 to ge
t first message or 2 to get second.
[   72.481247] + md3: Результат работы функции md1_fact(5): 120
```

Видно, что модули успешно загружены.

При попытке загрузить вначале модуль «md2» без модуля «md1», возникнет ошибка, поскольку программа не может обратиться к данным из модуля «md1»:

```
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo insmod md2.ko
insmod: ERROR: could not insert module md2.ko: Unknown symbol in module
```

Если же удалить из md1 экспорт данных:

```
EXPORT_SYMBOL(md1_str);
EXPORT_SYMBOL(md1_fact);

EXPORT_SYMBOL(export_char);
EXPORT_SYMBOL(export_int);
```

И попытаться последовательно загрузить модули «md1», «md2», то мы также получим ошибку обращения к данным, которые недоступны:

```
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo insmod md1.ko
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo insmod md2.ko
insmod: ERROR: could not insert module md2.ko: Unknown symbol in module
```

Ниже продемонстрирована попытка выгрузить модули ядра, начиная с md1.

```
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo rmmod md1
rmmod: ERROR: Module md1 is in use by: md2 md3
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo rmmod md2
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo rmmod md1
rmmod: ERROR: Module md1 is in use by: md3
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo rmmod md3
ilalevuskin@ubuntu:~/Desktop/3_os/2$ sudo rmmod md1
```

Видно, что выдается сообщение об ошибке, поскольку модуль «md1» используется в модулях «md2» и «md3».

Таким образом, корректный порядок выгрузки модулей имеет два варианта:

- md2, md3, md1
- md3, md2, md1