

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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 3

Дисциплина	Операционные системы.
Тема	Загружаемые модули ядра
Студент	Куприй А. А.
Группа	ИУ7-63Б
Преподаватель	Рязанова Н.Ю.

Москва, 2020 г.

1 Практическая часть

Проанализировать работу приведенных программ и объяснить результат их работы.

1.1 Задание №1

Реализовать загружаемый модуль ядра, который при загрузке записывает в системный журнал сообщение о запущенных процессах. Модуль должен собираться при помощи Make-файла. Загружаемый модуль должен содержать:

- указание лицензии GPL;
- указание автора.

Текст первой программы:

Листинг 1.1 — Текст программы первого задания

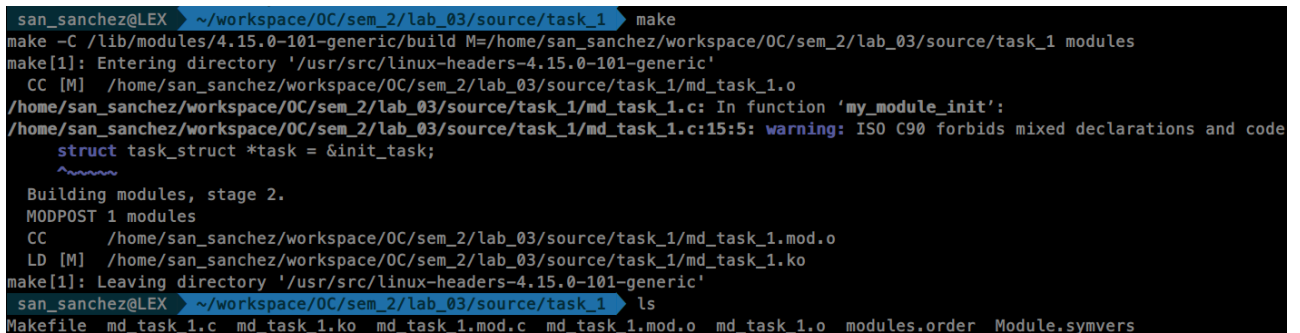
```
1 #include <linux/module.h>
2 #include <linux/init.h>
3 #include <linux/kernel.h>
4 #include <linux/sched.h>
5 #include <linux/init_task.h>
6
7 MODULE_LICENSE("GPL");
8 MODULE_AUTHOR("Alexander Kupry");
9 MODULE_DESCRIPTION("lab_03_task_1");
10
11 static int __init my_module_init(void)
12 {
13     printk(KERN_INFO "My module is loaded\n");
14
15     struct task_struct *task = &init_task;
16
17     do
18     {
19         printk(KERN_INFO "My module: process %s — %d, parent: %s — %d\n",
20             task->comm, task->pid, task->parent->comm, task->parent->pid);
21     }
22     while ((task = next_task(task)) != &init_task);
23
24
```

```

25     printk(KERN_INFO "My module: process current: %s — %d, parent: %s —
        %d\n",
26     current->comm, current->pid, current->parent->comm,
        current->parent->pid);
27
28     return 0;
29 }
30
31 static void __exit my_module_exit(void)
32 {
33     printk(KERN_INFO "My module is unloaded\n");
34 }
35
36 module_init(my_module_init);
37 module_exit(my_module_exit);

```

Результат сборки загружаемого модуля ядра md1 при помощи утилиты make:



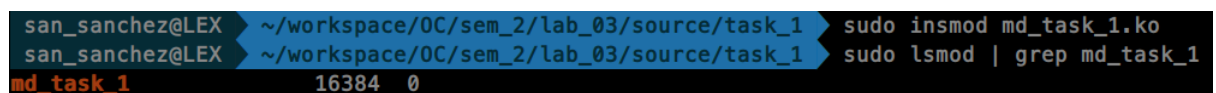
```

san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_1$ make
make -C /lib/modules/4.15.0-101-generic/build M=/home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_1 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-101-generic'
CC [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_1/md_task_1.o
/home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_1/md_task_1.c: In function 'my_module_init':
/home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_1/md_task_1.c:15:5: warning: ISO C90 forbids mixed declarations and code
    struct task_struct *task = &init_task;
    ~~~~~
Building modules, stage 2.
MODPOST 1 modules
CC      /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_1/md_task_1.mod.o
LD [M]  /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_1/md_task_1.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-101-generic'
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_1$ ls
Makefile md_task_1.c md_task_1.ko md_task_1.mod.c md_task_1.mod.o md_task_1.o modules.order Module.symvers

```

Рисунок 1.1 — Скришот результата работы make.

Загрузка модуля и демонстрация успешной загрузки:



```

san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_1$ sudo insmod md_task_1.ko
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_1$ sudo lsmod | grep md_task_1
md_task_1      16384  0

```

Рисунок 1.2 — Скришот результата работы lsmod.

Вывод dmesg:

```

[100867.928395] My module is loaded
[100867.928398] My module: process swapper/0 -- 0, parent: swapper/0 -- 0
[100867.928399] My module: process systemd -- 1, parent: swapper/0 -- 0
[100867.928401] My module: process kthreadd -- 2, parent: swapper/0 -- 0
[100867.928402] My module: process kworker/0:0H -- 4, parent: kthreadd -- 2
[100867.928404] My module: process mm_percpu_wq -- 6, parent: kthreadd -- 2
[100867.928405] My module: process ksoftirqd/0 -- 7, parent: kthreadd -- 2
[100867.928406] My module: process rcu_sched -- 8, parent: kthreadd -- 2
[100867.928408] My module: process rcu_bh -- 9, parent: kthreadd -- 2
[100867.928409] My module: process migration/0 -- 10, parent: kthreadd -- 2
[100867.928411] My module: process watchdog/0 -- 11, parent: kthreadd -- 2
[100867.928412] My module: process cpuhp/0 -- 12, parent: kthreadd -- 2
[100867.928413] My module: process cpuhp/1 -- 13, parent: kthreadd -- 2
[100867.928415] My module: process watchdog/1 -- 14, parent: kthreadd -- 2
[100867.928416] My module: process migration/1 -- 15, parent: kthreadd -- 2
[100867.928418] My module: process ksoftirqd/1 -- 16, parent: kthreadd -- 2
[100867.928419] My module: process kworker/1:0H -- 18, parent: kthreadd -- 2
[100867.928421] My module: process cpuhp/2 -- 19, parent: kthreadd -- 2
[100867.928422] My module: process watchdog/2 -- 20, parent: kthreadd -- 2
[100867.928424] My module: process migration/2 -- 21, parent: kthreadd -- 2
[100867.928425] My module: process ksoftirqd/2 -- 22, parent: kthreadd -- 2
[100867.928426] My module: process kworker/2:0H -- 24, parent: kthreadd -- 2
[100867.928427] My module: process cpuhp/3 -- 25, parent: kthreadd -- 2
[100867.928429] My module: process watchdog/3 -- 26, parent: kthreadd -- 2
[100867.928430] My module: process migration/3 -- 27, parent: kthreadd -- 2
[100867.928431] My module: process ksoftirqd/3 -- 28, parent: kthreadd -- 2
[100867.928433] My module: process kworker/3:0H -- 30, parent: kthreadd -- 2
[100867.928434] My module: process cpuhp/4 -- 31, parent: kthreadd -- 2
[100867.928436] My module: process watchdog/4 -- 32, parent: kthreadd -- 2
[100867.928437] My module: process migration/4 -- 33, parent: kthreadd -- 2
[100867.928439] My module: process ksoftirqd/4 -- 34, parent: kthreadd -- 2

```

Рисунок 1.3 — Скришот результата работы dmesg.

Демонстрация успешной выгрузки модуля.

```

san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_1 sudo rmmod md_task_1.ko
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_1 sudo dmesg | tail -1
[100986.156589] My module is unloaded

```

Рисунок 1.4 — Скришот результата работы rmmod.

1.2 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 функции. 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 функции, 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 сценарии некорректного завершения установки модуля, и возможность использования загружаемого модуля в качестве функции, выполняемой в пространстве ядра. 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.

Листинг Makefile:

Листинг 1.2 — Текст header файла md.h

```
1 ifneq ($(KERNELRELEASE),)
2     obj-m    := md1.o md2.o md3.o
3 else
4     CURRENT = $(shell uname -r)
5     KDIR = /lib/modules/$(CURRENT)/build
6     PWD = $(shell pwd)
7 default:
8     $(MAKE) -C $(KDIR) M=$(PWD) modules
9 clean:
10     @rm -f *.o *.cmd *.flags *.mod.c *.order
11     @rm -f *.*.cmd *~ *.*~ TODO.*
12     @rm -fR .tmp*
13     @rm -rf .tmp_versions
14 disclean: clean
15     @rm *.ko *.symvers
16
17 endif
```

Листинг md.h:

Листинг 1.3 — Текст header файла md.h

```
1 #ifndef MD
2 #define MD
3
```

```
4 extern char* md1_str_data;
5 extern int md1_int_data;
6 extern char* md1_get_str(int n);
7 extern int md1_factorial(int n);
8
9 #endif
```

Листинг md1.c:

Листинг 1.4 — Текст header файла md1.c

```
1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include "md.h"
4
5 MODULE_LICENSE("GPL");
6 MODULE_AUTHOR("Alexander Kupry");
7
8 char* md1_str_data = "MD1: Hello , world!";
9 int md1_int_data = 42;
10
11 extern char* md1_get_str(int n)
12 {
13     printk( "+ md1: md1_get_str() called!\n" );
14     switch (n)
15     {
16     case 1:
17         return "Message 1!\n";
18         break;
19     case 2:
20         return "Message 2!\n";
21         break;
22     default:
23         return "Other message!\n";
24         break;
25     }
26 }
27
28 extern int md1_factorial(int n)
29 {
30     int i, res;
31     res = 1;
32
33     printk( "+ md1: md1_factorial() called!\n" );
34     if (n <= 0)
35     {
36         return 0;
```

```

37     }
38     for (i = 2; i <= n; i++)
39     {
40         res *= i;
41     }
42
43     return res;
44 }
45
46 EXPORT_SYMBOL(md1_str_data);
47 EXPORT_SYMBOL(md1_int_data);
48
49 EXPORT_SYMBOL(md1_get_str);
50 EXPORT_SYMBOL(md1_factorial);
51
52
53 static int __init md_init( void )
54 {
55     printk( "+ md1: module md1 start!\n" );
56     return 0;
57 }
58
59 static void __exit md_exit( void )
60 {
61     printk( "+ md1: module md1 unloaded!\n" );
62 }
63
64 module_init( md_init );
65 module_exit( md_exit );

```

Листинг md2.c:

Листинг 1.5 — Текст header файла md2.c

```

1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include "md.h"
4
5 MODULE_LICENSE("GPL");
6 MODULE_AUTHOR("Alexander Kupry");
7
8 static int __init md_init(void)
9 {
10     printk( "+ md2: module md2 start!\n" );
11     printk( "+ md2: number from md1 : %d\n", md1_int_data );
12     printk( "+ md2: string from md1 : %s\n", md1_str_data );
13     printk( "+ md2: result md1_get_str(0) : %s\n", md1_get_str(0) );

```



```

14     printk( "+ md2: result md1_get_str(1) : %s\n", md1_get_str(1) );
15     printk( "+ md2: result md1_get_str(2) : %s\n", md1_get_str(2) );
16     printk( "+ md2: result md1_factorial(4) : %d\n", md1_factorial(4) );
17
18     return 0;
19 }
20
21 static void __exit md_exit( void )
22 {
23     printk( "+ md2: module md2 unloaded!\n" );
24 }
25
26 module_init( md_init );
27 module_exit( md_exit );

```

Листинг md3.c:

Листинг 1.6 — Текст header файла md3.c

```

1  #include <linux/init.h>
2  #include <linux/module.h>
3  #include "md.h"
4
5  MODULE_LICENSE("GPL");
6  MODULE_AUTHOR("Alexander Kupry");
7
8  static int __init md_init( void )
9  {
10     printk( "+ md3: module md3 start!\n" );
11     printk( "+ md3: number from md1 : %d\n", md1_int_data );
12     printk( "+ md3: string from md1 : %s\n", md1_str_data );
13     printk( "+ md3: result md1_get_str(0) : %s\n", md1_get_str(0) );
14     printk( "+ md3: result md1_get_str(1) : %s\n", md1_get_str(1) );
15     printk( "+ md3: result md1_get_str(2) : %s\n", md1_get_str(2) );
16     printk( "+ md3: result md1_factorial(4) : %d\n", md1_factorial(4) );
17
18     return -1;
19 }
20
21 module_init( md_init );

```

Результат.

Результат сборки при помощи утилиты make:

```

san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 make
make -C /lib/modules/4.15.0-101-generic/build M=/home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-101-generic'
CC [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md1.o
CC [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md2.o
CC [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md3.o
Building modules, stage 2.
MODPOST 3 modules
CC /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md1.mod.o
LD [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md1.ko
CC /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md2.mod.o
LD [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md2.ko
CC /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md3.mod.o
LD [M] /home/san_sanchez/workspace/OC/sem_2/lab_03/source/task_2/md3.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-101-generic'
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 ls
Makefile md1.ko md1.mod.o md2.c md2.mod.c md2.o md3.ko md3.mod.o md.h Module.symvers
md1.c md1.mod.c md1.o md2.ko md2.mod.o md3.c md3.mod.c md3.o modules.order

```

Рисунок 1.5 — Скришот результата работы make.

Попытка загрузки модулей в неправильном порядке:

```

✗ san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo insmod md2.ko
insmod: ERROR: could not insert module md2.ko: Unknown symbol in module
✗ san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo dmesg | tail -4
[107714.846133] md2: Unknown symbol md1_int_data (err 0)
[107714.846152] md2: Unknown symbol md1_factorial (err 0)
[107714.846169] md2: Unknown symbol md1_get_str (err 0)
[107714.846184] md2: Unknown symbol md1_str_data (err 0)

```

Рисунок 1.6 — Скришот результата работы insmod.

Ошибка возникла по причине того, что модуль содержит ссылки на неизвестные ядру имена.

Для правильной работы необходимо сначала загрузить md1.ko, а уже после этого - md2.ko.

Результат загрузки модулей в правильном порядке:

```

✗ san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo insmod md1.ko
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo insmod md2.ko
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo insmod md3.ko
insmod: ERROR: could not insert module md3.ko: Operation not permitted

```

Рисунок 1.7 — Скришот результата работы insmod.

Демонстрация успешной загрузки:

```
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo lsmod
Module          Size Used by
md2              16384 0
md1              16384 1 md2
```

Рисунок 1.8 — Скришот результата работы lsmod.

Вывод dmesg:

```
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo dmesg | grep +
[107451.609331] + md1: module md1 start!
[107454.260111] + md2: module md2 start!
[107454.260113] + md2: number from md1 : 42
[107454.260113] + md2: string from md1 : MD1: Hello, world!
[107454.260114] + md1: md1_get_str() called!
[107454.260114] + md2: result md1_get_str(0) : Other message!
[107454.260115] + md1: md1_get_str() called!
[107454.260115] + md2: result md1_get_str(1) : Message 1!
[107454.260115] + md1: md1_get_str() called!
[107454.260116] + md2: result md1_get_str(2) : Message 2!
[107454.260117] + md1: md1_factorial() called!
[107454.260117] + md2: result md1_factorial(4) : 24
[107456.185034] + md3: module md3 start!
[107456.185036] + md3: number from md1 : 42
[107456.185037] + md3: string from md1 : MD1: Hello, world!
[107456.185038] + md1: md1_get_str() called!
[107456.185038] + md3: result md1_get_str(0) : Other message!
[107456.185039] + md1: md1_get_str() called!
[107456.185040] + md3: result md1_get_str(1) : Message 1!
[107456.185040] + md1: md1_get_str() called!
[107456.185041] + md3: result md1_get_str(2) : Message 2!
[107456.185042] + md1: md1_factorial() called!
[107456.185043] + md3: result md1_factorial(4) : 24
```

Рисунок 1.9 — Скришот результата работы dmesg.

Попытка выгрузки модулей в неправильном порядке также вызовет ошибку:

```
san_sanchez@LEX ~/workspace/OC/sem_2/lab_03/source/task_2 sudo rmmod md1.ko
rmmod: ERROR: Module md1 is in use by: md2
```

Рисунок 1.10 — Скришот результата работы rmmod.

Результат успешной выгрузки модулей:

```
san_sanchez@LEX > ~/workspace/0C/sem_2/lab_03/source/task_2 > sudo rmmod md2.ko  
san_sanchez@LEX > ~/workspace/0C/sem_2/lab_03/source/task_2 > sudo rmmod md1.ko  
san_sanchez@LEX > ~/workspace/0C/sem_2/lab_03/source/task_2 > sudo dmesg | tail -2  
[109909.557162] + md2: module md2 unloaded!  
[109912.111957] + md1: module md1 unloaded!
```

Рисунок 1.11 — Скришот результата работы `rmmod`.

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были проанализированный особенности работы функции ввода-вывода в UNIX/LINUX.