|  |  |
| --- | --- |
|  | **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»

**ОТЧЕТ**

*к лабораторной работе №3*

*По курсу: «Операционные системы»*

*На тему: «Загружаемые модули ядра»*

Студентка ИУ7-65Б

Оберган Т.М

Преподаватель

Рязанова Н.Ю.

*2020 г.*

Оглавление

[Часть 1 3](#_Toc42115081)

[Листинг 3](#_Toc42115082)

[Результат работы программы 4](#_Toc42115083)

[Часть 2 5](#_Toc42115084)

[Задание 5](#_Toc42115085)

[Листинг 5](#_Toc42115086)

[Результат работы программы 7](#_Toc42115087)

# Часть 1

## Листинг

Содержимое makefile:

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

make clean

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

Содержимое md.c:

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/sched.h>

#include <linux/init\_task.h>

MODULE\_LICENSE("Dual BSD/GPL");

MODULE\_AUTHOR("Obergan T.M");

**static** **int** \_\_init md\_init(**void**)

{

printk("LAB3: Hello world.\n");

**struct** task\_struct \*task = &init\_task;

**do**

{

printk("LAB3:---%s-%d, parent %s-%d", task->comm,

task->pid, task->parent->comm, task->parent->pid);

} **while** ((task = next\_task(task)) != &init\_task);

printk("LAB3:---%s-%d, parent %s-%d", current->comm,

current->pid, current->parent->comm, current->parent->pid);

**return** 0;

}

**static** **void** \_\_exit md\_exit(**void**)

{

printk("LAB3: Goodbye.\n");

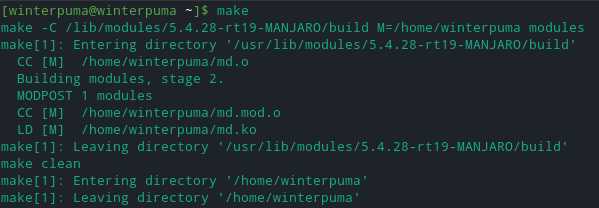
}

module\_init(md\_init);

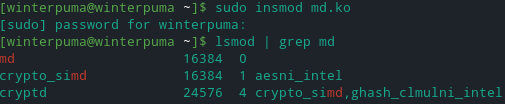
module\_exit(md\_exit);

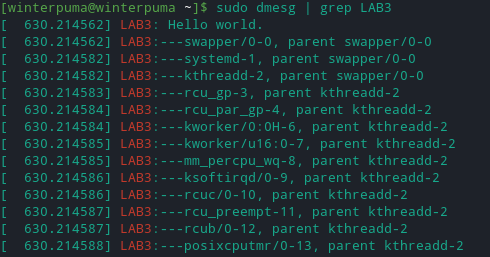
## Результат работы программы

Компиляция модуля при помощи makefile:

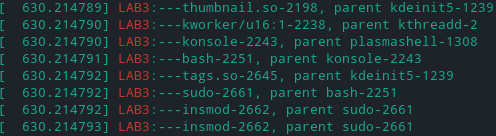


Загрузка модуля при помощи команды insmod:

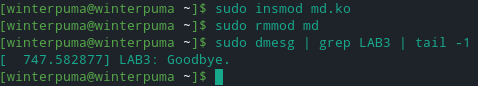




…



Выгрузка модуля при помощи команды rmmod:



# Часть 2

## Задание

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.

## Листинг

md.h:

//#ifndef MY\_MD

//#define MY\_MD 1

**extern** **char**\* md1\_data;

**extern** **char**\* md1\_proc( **void** );

**extern** **char**\* md1\_noexport( **void** );

//#endif

md1.c

#include <linux/init.h>

#include <linux/module.h>

#include "md.h"

MODULE\_LICENSE("Dual BSD/GPL");

MODULE\_AUTHOR("Obergan T.M");

**char**\* md1\_data = "Hello world!";

**extern** **char**\* md1\_proc(**void**)

{

**return** md1\_data;

}

**static** **char**\* md1\_local(**void**)

{

**return** md1\_data;

}

**extern** **char**\* md1\_noexport(**void**)

{

**return** md1\_data;

}

EXPORT\_SYMBOL(md1\_data);

EXPORT\_SYMBOL(md1\_proc);

**static** **int** \_\_init md\_init(**void**)

{

printk("LAB3: module md1 start!\n");

**return** 0;

}

**static** **void** \_\_exit md\_exit(**void**)

{

printk("LAB3: module md1 unloaded!\n");

}

module\_init(md\_init);

module\_exit(md\_exit);

md2.c

#include <linux/init.h>

#include <linux/module.h>

#include "md.h"

MODULE\_LICENSE("Dual BSD/GPL");

MODULE\_AUTHOR("Obergan T.M");

**static** **int** \_\_init md\_init(**void**)

{

printk("LAB3: module md2 start!\n");

printk("LAB3: data string exported from md1 : %s\n", md1\_data);

printk("LAB3: string returned md1\_proc() is : %s\n", md1\_proc());

**return** 0;

}

**static** **void** \_\_exit md\_exit(**void**)

{

printk("LAB3: module md2 unloaded!\n");

}

module\_init(md\_init);

module\_exit(md\_exit);

md3.c

#include <linux/init.h>

#include <linux/module.h>

#include "md.h"

MODULE\_LICENSE("Dual BSD/GPL");

MODULE\_AUTHOR("Obergan T.M");

**static** **int** \_\_init md\_init(**void**)

{

printk("LAB3: module md3 start!\n");

printk("LAB3: data string exported from md1 : %s\n", md1\_data);

printk("LAB3: string returned md1\_proc() is : %s\n", md1\_proc());

**return** -1;

}

**static** **void** \_\_exit md\_exit(**void**)

{

printk("LAB3: module md3 unloaded!\n");

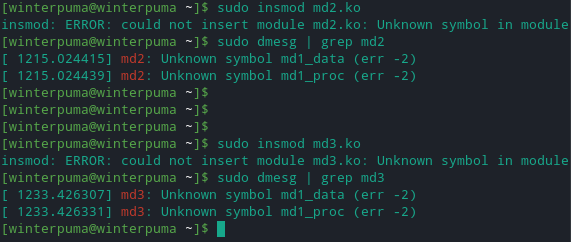
}

module\_init(md\_init);

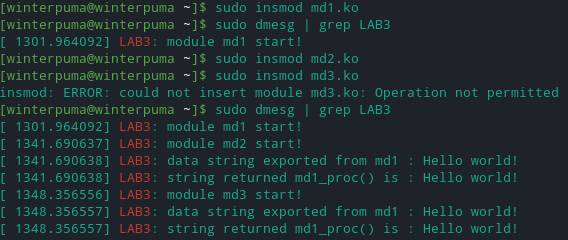
module\_exit(md\_exit);

## Результат работы программы

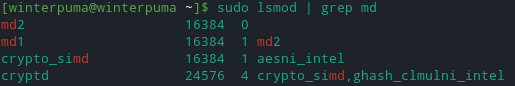
Так как md2 и md3 используют данные из md1, то при попытке загрузки этих модулей до загрузки md1 возникнет ошибка.



При загрузке модулей в правильном порядке md1 и md2 будут успешно загружены, а md3 не будет загружен т.к. функция инициализации md3 вернула ненулевое значение (но код инициализирующей функции будет выполнен, что можно наблюдать в буфере сообщений ядра).



Модуль md1 используется модулем md2:



При попытке выгрузки md2 до md1 возникнет следующая ошибка:



Для успешной выгрузки, модули нужно выгружать в обратном порядке (сначала md2, а потом md1).

