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»

Студент: Батбилэг Н.

Группа: ИУ7-71Б

Оценка (баллы) _____

Преподаватель: Рязанова Н.Ю.

Москва.
2021 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 информацию о процессах. Модуль должен собираться при помощи Make-файла.

Листинг 1.1: 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
```

Листинг 1.2: 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("GPL");
MODULE_AUTHOR("Nomuundalai");
MODULE_DESCRIPTION("lab_03");

static int __init md_init(void)
{
	printk("LAB3.2 : Hello world!\n");
	struct task_struct* task = &init_task;
	printk("LAB3.2 : %s-%d\tparent %s-%d\n", current→comm, current→pid,
					current→parent→comm, current→parent→pid);

	Do
	{
		printk("LAB3.2 : %s-%d\tparent %s-%d\n", task→comm, task→pid,
					task→parent→comm, task→parent→pid);
		task = next_task(task);
	} while (task != &init_task);
```

```
	return 0;
}

static void __exit md_exit(void)
{
	printk("LAB3.2 : Goodbye.\n");
}

module_init(md_init);
module_exit(md_exit);
```

На рисунке 1.1 представлена загрузка модуля ядра.

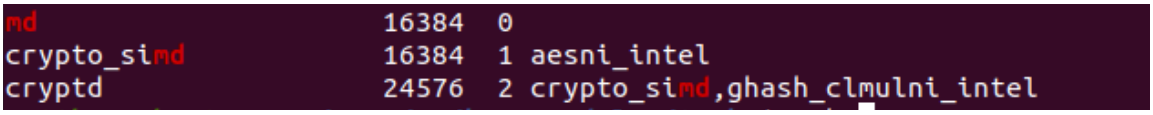

Рисунок 1.1. Загрузка модуля ядра.

Результат dmesg на Рисунке 1.2.

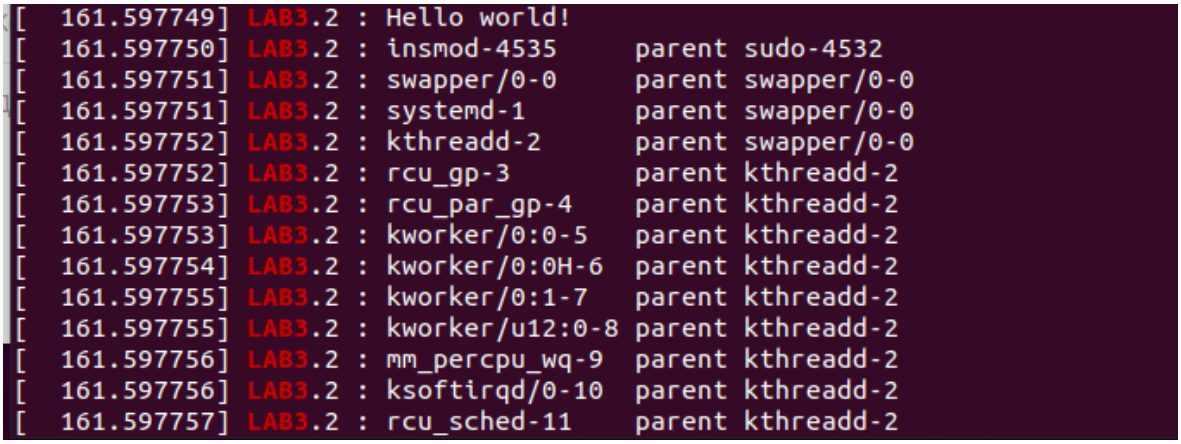

Рисунок 1.2: Сообщения в системном журнале от «md»

Результат rmmod md на Рисунке 1.3.

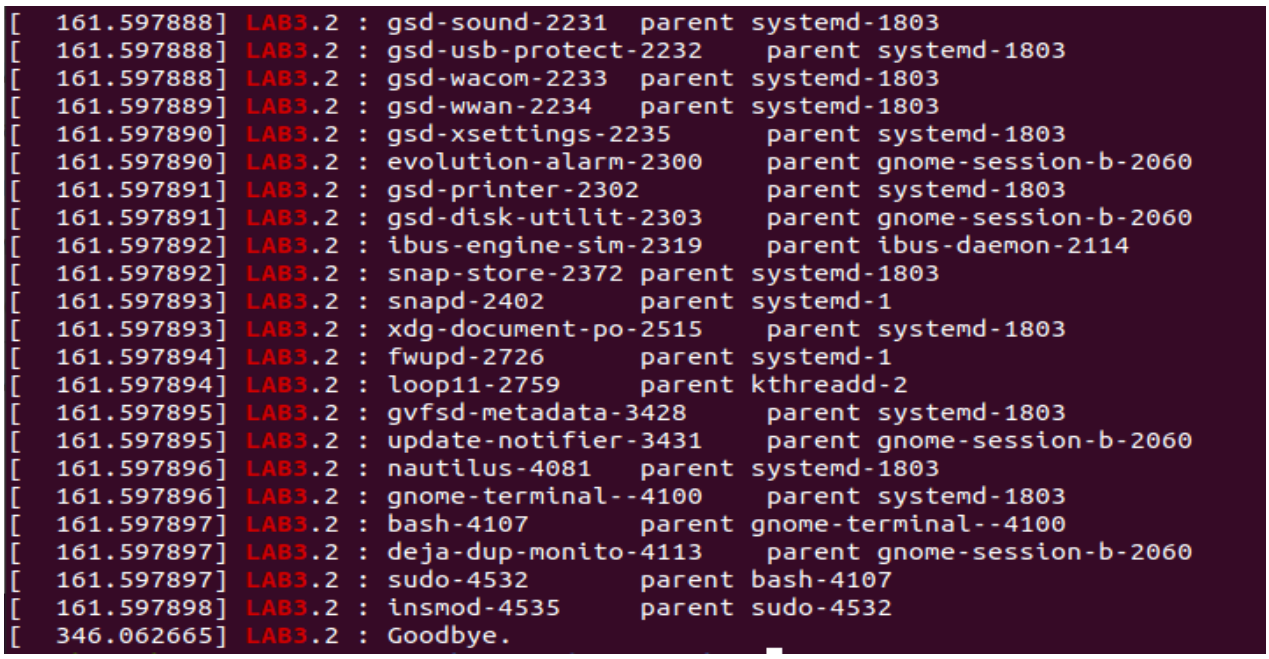

Рисунок 1.3: Системный журнал после выполнения «rmmod md»

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.

Листинг 2.1: Makefile

```
KBUILD_EXTRA_SYMBOLS = $(shell pwd)/Module.symverscd
ifneq ($(KERNELRELEASE),)
	obj-m := md1.o md2.o md3.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
```

Листинг 2.2: md.h

```
#ifndef MD_H
#define MD_H 1

extern char* md1_data;
extern char* md1_proc(void);
extern char* md1_noexport(void);

#endif
```

Листинг 2.3: md1.c

```
#include <linux/init.h>
#include <linux/module.h>
#include "md.h"

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Nomuundalai");
MODULE_DESCRIPTION("lab_03");

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("LAB3.3 : module md1 start!\n");
        return 0;
}

static void __exit md_exit(void)
{
        printk("LAB3.3 : module md1 unloaded!\n");
}

module_init(md_init);
module_exit(md_exit);
```

Листинг 2.4: md2.c

```
#include <linux/init.h>
#include <linux/module.h>
#include "md.h"

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Nomuundalai");
MODULE_DESCRIPTION("lab_03");


static int __init md_init(void)
{
        printk("LAB3.3 : module md2 start!\n");
        printk("LAB3.3 : data string exported from md1 : %s\n", md1_data);
        printk("LAB3.3 : string returned md1_proc() is : %s\n", md1_proc());
        return 0;
}

static void __exit md_exit(void)
{
        printk("LAB3.3 : module md2 unloaded!\n");
}

module_init(md_init);
module_exit(md_exit);
```

Листинг 2.5: md3.c

```
#include <linux/init.h>
#include <linux/module.h>
#include "md.h"

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Nomuundalai");
MODULE_DESCRIPTION("lab_03");

static int __init md_init(void)
{
        printk("LAB3.3 : module md3 start!\n");
        printk("LAB3.3 : data string exported from md1 : %s\n", md1_data);
        printk("LAB3.3 : string returned md1_proc() is : %s\n", md1_proc());
        return -1;
}

static void __exit md_exit(void)
{
        printk("LAB3.3 : module md3 unloaded!\n");
}

module_init(md_init);
module_exit(md_exit);
```

Результат работы программы:
Попробуем загрузить 2 модуль без загрузки 1, результат на Рисунке 2.1.

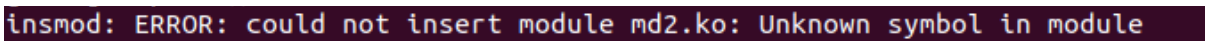

Рисунок 2.1: Ошибка при неправильном порядке загрузки

Загрузим 1 модуль, результат на Рисунке 2.1.

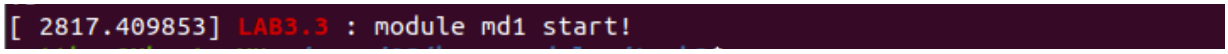

Рисунок 2.2: Журнал после загрузки 1 модуля

Загрузим 2 модуль, после загрузки 1, результат на Рисунке 2.3.

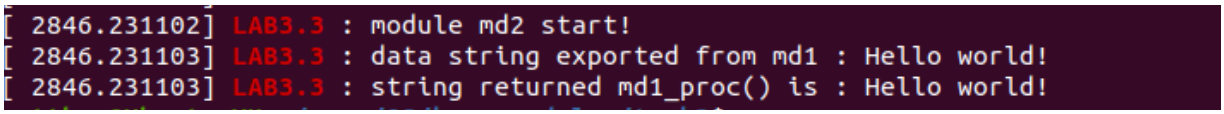

Рисунок 2.3: Журнал после загрузки 2 модуля

Ошибка при загрузке 3 модуля, так как «static int __init md_init(void)» намеренно возвращает ненулевое значение, результат на Рисунке 2.4 и 2.5.

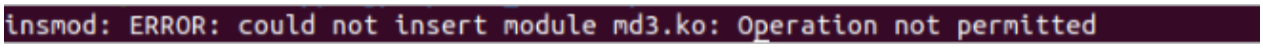

Рисунок 2.4: Ошибка загрузки 3 модуля

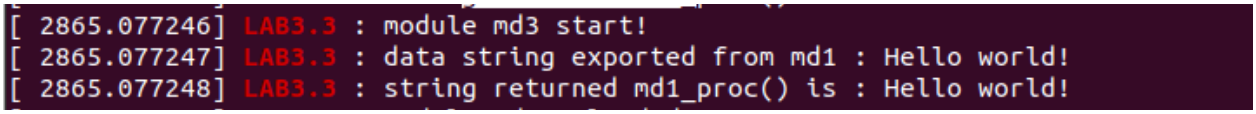

Рисунок 2.5: Журнал после загрузки 3 модуля, так как md_init выполнилась

Попытаемся удалить 1 модуль, без удаления 2 (использует символы из 1), результат на Рисунке 2.6.

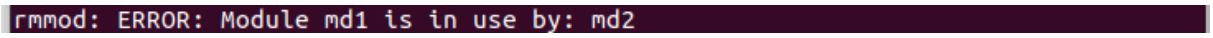

Рисунок 2.6: Ошибка удаления 1 модуля

Произошло это из-за того, что счетчик ссылок на 1 модуль ненулевой.
Удалим модули в правильном порядке, результат на Рисунке 2.7.

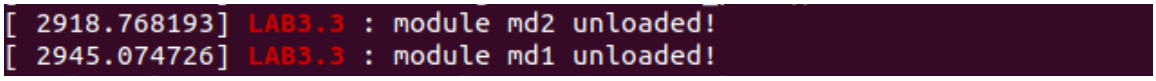

Рисунок 2.7: Журнал после удаления 2 и 1 модулей