|  |  |
| --- | --- |
|  | **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»*

Студент ИУ7-66Б

Турсунов Ж.Р.

Преподаватель

Рязанова Н.Ю.

*Москва, 2021 г.*

# Часть 1

1. Реализовать загружаемый модуль ядра, который при загрузке записывает в системный журнал информацию о процессах. О каждом процессе записать: название, идентификатор, название предка, идентификатор предка. Модуль должен собираться при помощи make файла.

# Листинг

|  |
| --- |
|  |

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/sched.h>

#include <linux/init\_task.h>

// чтобы сообщить ядру, под какой лицензией распространяется исходный код модуля.

// Если лицензия не соотв. GPL то загрузка модуля приведет к установке в ядре флага tainted (испорчено).

MODULE\_LICENSE**(**"GPL"**);**

// позволяет сообщить ядру автора модуля, только для информационных целей

MODULE\_AUTHOR**(**"Tursunov Jasur"**);**

// \_\_init - действие, на которое в системе надо обратить внимание

// \_\_init используются для обозначения функций, встраиваемых в ядро,

// указание о том, что функция используется только на этапе инициализации

// и освобождает использованные ресурсы после.

// Это позволяет экономить ресурсы системы

// static - т.к. функцию не нужно экспортировать

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

**{**

// printk - функция ядра (определена в ядре Linux) - выводит переданную строку в var/log/message

// записывает сообщ в буфер ядра (в системный журнал)

printk**(**"LAB3: Hello World\n"**);**

// &init\_task - точка входа в кольцевой список task\_struct

// task\_struct - дескриптор процесса

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

**do**

**{**

// KERN\_INFO - уровень протоколирования - информационные сообщения

// ЗАПЯТАЯ НЕ СТАВИТСЯ тк уровень протоколирования - часть

// строки форматирования (для экономии памяти стека при вызове функции)

printk**(**"LAB3:---%s-%d, parent: %s-%d"**,**

task**->**comm**,** task**->**pid**,** // task->comm - символическое имя исполняемого файла в файловой подсистеме

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

// в таск\_стракт есть ссылка на файловую подсистему, которой принадлежить процесс

**}** **while** **((**task **=** next\_task**(**task**))** **!=** **&**init\_task**);** // поскольку список кольцевой

// current - указатель на task\_struct текущего процесса

printk**(**"LAB3:---%s-%d, parent: %s-%d"**,**

current**->**comm**,** current**->**pid**,**

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

**return** 0**;**

**}**

// пропуск функции когда модуль встроен (built) в ядро - built-in drivers

// has no effect for loadable modules

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

**{**

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

**}**

// загружаемые модули ядра должны содержать два макроса

module\_init**(**md\_init**);**

// для регистрации функции инициализации модуля - функция md\_init будет вызываться при загрузке модуля в ядро

module\_exit**(**md\_exit**);**

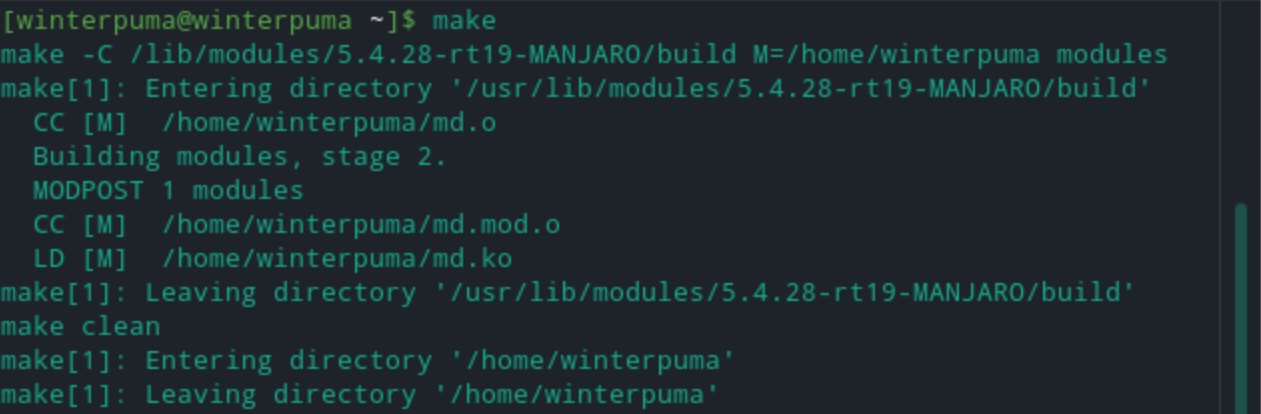
// для регистрации функции которая вызывается при удалении модуля из ядра

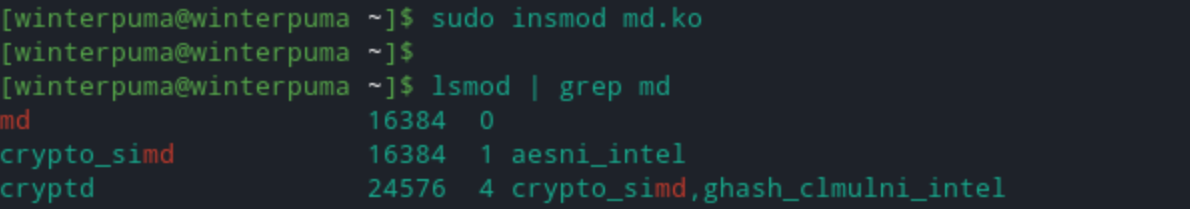
// как правило, выполняет освобождение ресурсов

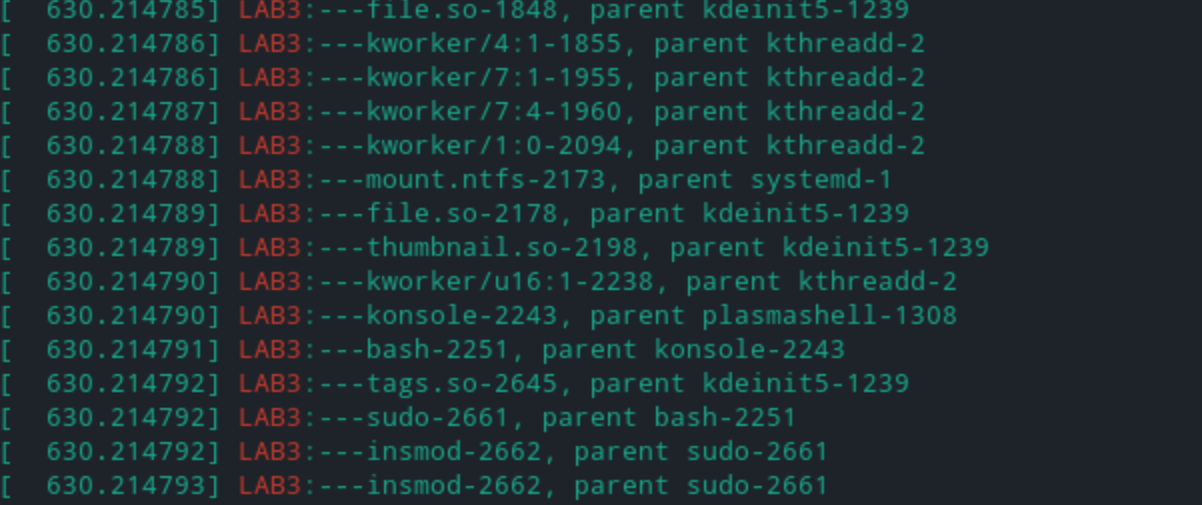
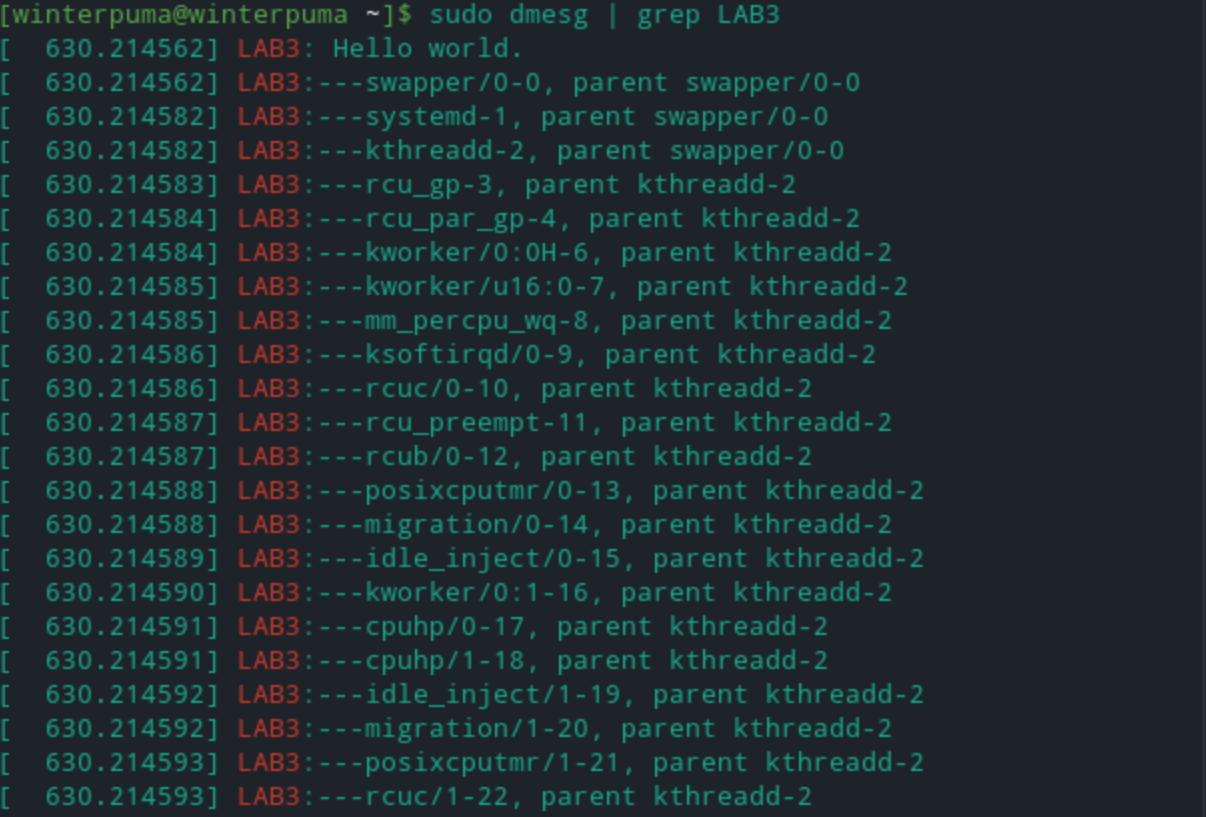
// после завершения этой функции модуль выгружается из ядра

**Результат работы программы**

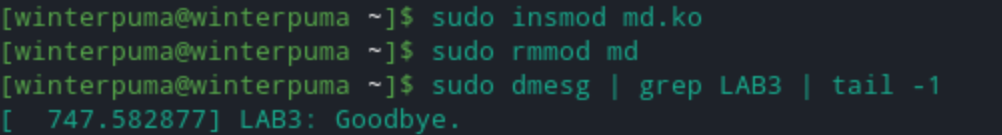
**Компиляция модуля при помощи makefile:**

****

**Загрузка модуля при помощи команды insmod:**



**Выгрузка модуля при помощи команды rmmod:**

****

# Часть 2

1. 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

# 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

**Листинг md1**

**Взаимодействие загружаемых модулей ядра. Модуль 1 демонстрирует возможность создания экспортируемых данных и функций**

#include <linux/module.h> // макросы MODULE\_<...>

#include <linux/init.h> // \_\_init и \_\_exit

#include "md.h"

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_AUTHOR**(**"Tursunov Jasur"**);**

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**);**

/\*

Модуль md2 связывается с экспорт-м именем по абсолютному адресу.

-> бессмысленно предоставлять модуль в собранном виде —

он должен собираться на месте использования.

\*/

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**);**

**Листинг md2**

**Взаимодействие загружаемых модулей ядра. Модуль 2 демонстрирует использование данных и функций экспортируемых первым модулем**

#include <linux/module.h> // макросы MODULE\_<...>

#include <linux/init.h> // \_\_init и \_\_exit

#include "md.h"

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_AUTHOR**(**"Tursunov Jasur"**);**

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: proc md1\_noexport returns: %s\n"**,** md1\_noexport**());**

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

**}**

module\_init**(**md\_init**);**

module\_exit**(**md\_exit**);**

**Листинг md3**

**Взаимодействие загружаемых модулей ядра. Модуль 3 демонстрирует сценарий некорректного завершения установки модуля, и возможность использования загружаемого модуля в качестве функции выполняемой в пространстве ядре**

#include <linux/module.h>

#include <linux/init.h>

#include "md.h"

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_AUTHOR**(**"Tursunov Jasur"**);**

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**());**

// преднамеренно возвращает ненулевое значение,

// что означает ошибку инициализации модуля;

// тогда такой модуль не будет подгружен к ядру, но произойдёт это

// уже после выполнения кода инициализирующей функции модуля в пространстве ядра;

**return** **-**1**;**

**}**

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

**{**

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

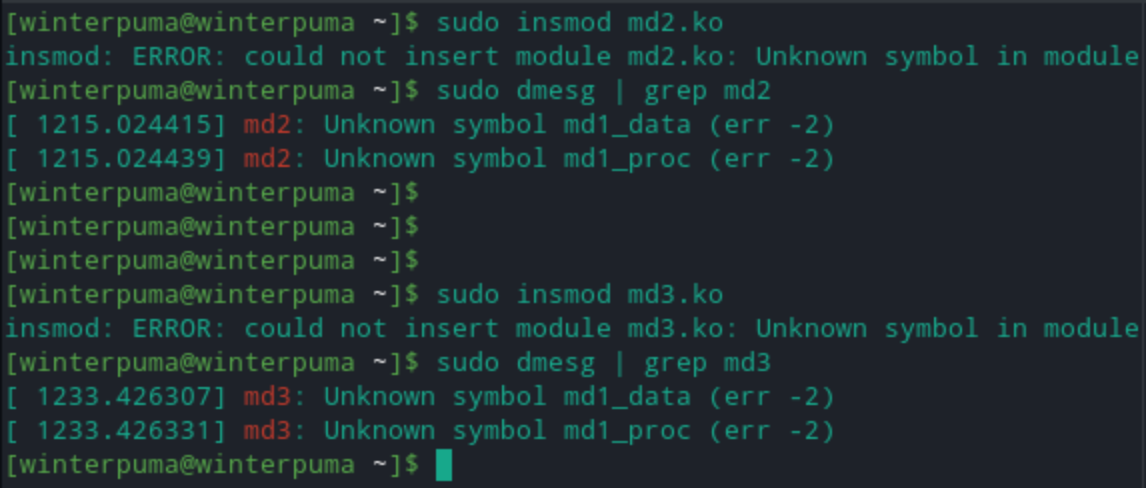
**}**

module\_init**(**md\_init**);**

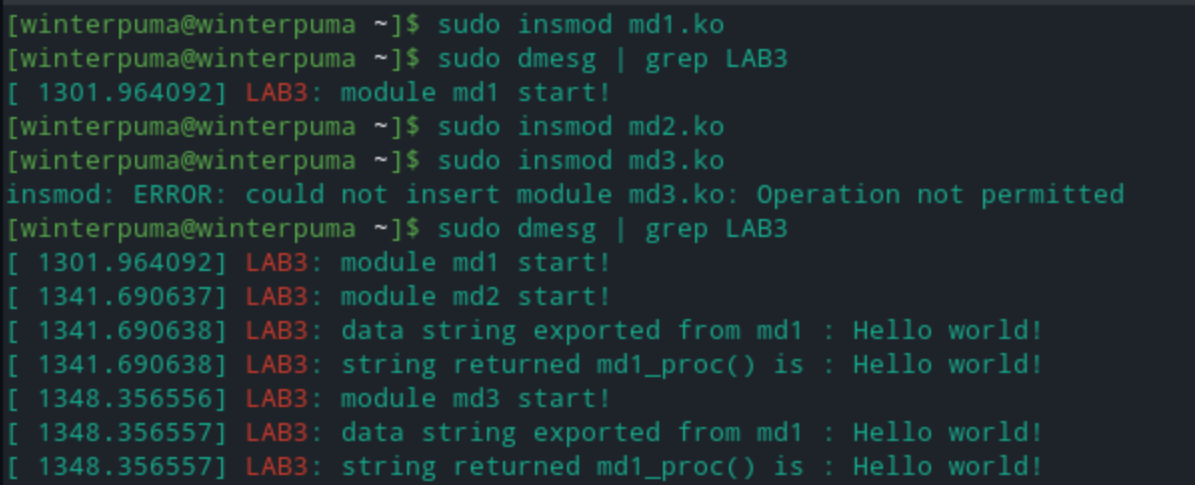
module\_exit**(**md\_exit**);**

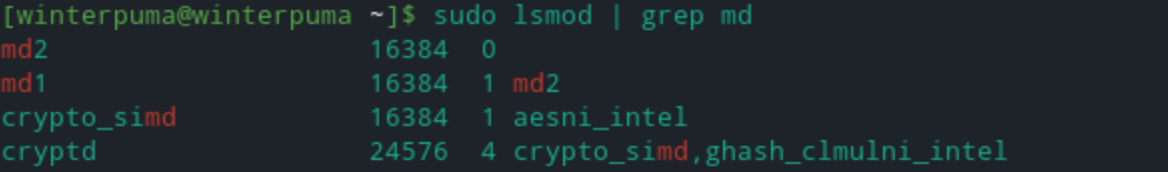
**Результат работы программы**

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

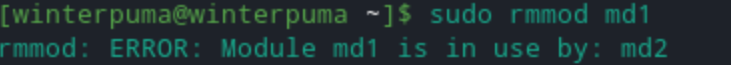


При загрузке модулей в правильном порядке md1 и md2 будут успешно загружены, а md3 не будет загружен так как функция инициализации md3 вернула ненулевое значение (но код инициализирующей функции будет выполнен, что можно наблюдать в буфере сообщений ядра).



Модуль md1 используется модулем md2:

При попытке выгрузки md2 до md1 возникает следующая ошибка:



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

