**Лабораторная работа № 13.**

**Разработка модуля ядра в Linux.**

Цель лабораторной работы:

* Научиться создавать и использовать модули ядра Linux

Несмотря на то, что ядро Linux является монолитным, оно позволяет выполнять динамическую вставку и удаление кода ядра в процессе работы. Загружаемый объект ядра называется *модулем*.

Модуль по своей сути примерно то же, что и обычная программа. Модуль так же имеет точку входа и выхода и находится в своем бинарном файле. Но модули имеют непосредственный доступ к структурам и функциям ядра. Для программ в пространстве пользователя этот доступ ограничен библиотечными интерфейсами компилятора.

**Задание 1.** Просмотрите список загруженных модулей командой *lsmod*, которая в свою очередь обращается за необходимыми сведениями к файлу */proc/modules*.

**Задание 2.** Реализуйте простейший код модуля ядра (классический "Hello, world").

2.1. Ознакомьтесь с кодом модуля ядра.

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

void sayHello()

{

printk(KERN\_INFO "Hello, world\n");

}

EXPORT\_SYMBOL(sayHello);

static int \_\_init hello\_init(void)

{

sayHello();

return 0;

}

static void \_\_exit hello\_exit(void)

{

printk(KERN\_INFO "Goodbye, world\n");

}

module\_init(hello\_init);

module\_exit(hello\_exit);

MODULE\_LICENSE("GPL");

Точка входа в модуль – это функция, объявленная как static int \_\_init. Функция обязана в случае успешной загрузки возвращать нулевое значение. Точка выхода – функция, объявленная как static void \_\_exit.

Основное назначение функции *printk* – дать ядру механизм регистрации событий и предупреждений. В модуле ядра нет доступа к стандартным потокам ввода-вывода, поэтому печать может осуществляться только в стандартный лог-файл с помощью функции *printk*. Результаты этого вывода можно посмотреть с помощью команды *dmesg*.

Любой модуль ядра должен подключать заголовочный файл *linux/module.h*. В файле *linux/kernel.h* содержатся определения макросов для функции *printk*, в том числе и *KERN\_INFO*. В заголовочном файле *linux/init.h* содержатся определения макросов *\_\_init* и *\_\_exit*.

C помощью директивы *EXPORT\_SYMBOL()* осуществляется экспорт функций ядра. Экспорт функций необходим, чтобы они были доступными для других модулей.

В данном случае функция *sayHello()* будет доступна для других модулей. Директива *MODULE\_LICENSE()* указывает на лицензию, под которой распространяется данный модуль. Некоторые функции могут быть доступны только для модулей, распространяемых под лицензией GPL (например, функции для работы с очередями процессов).

2.2. Создайте в домашнем каталоге подкаталог *hello*. Создайте в этом подкаталоге файл *hello.c*. Добавьте в него приведенный в предыдущем задании код.

2.3. Создайте в подкаталоге *hello* файл с именем *Makefile*. Добавьте в него следующие строчки:

obj-m += hello.o

KERNEL\_VERSION := $(shell uname -r)

KDIR ?= /lib/modules/$( KERNEL\_VERSION) )/build

all:

$(MAKE) -C $(KDIR) M=$(PWD) modules

clean:

$(MAKE) -C $(KDIR) M=$(PWD) clean

Цель *obj-m* содержит список модулей, которые нужно собрать. При этом нет необходимости перечислять файлы с исходным кодом *.c*.

2.4. Что содержится в переменной *KDIR* файла *Makefile*?

2.5. За что отвечает параметр *M* в файле *Makefile*?

2.6. В терминале перейдите в каталог *hello* и выполните команду *make*. Убедитесь в том, что сборка модуля завершилась успешно. Какие файлы появились в результате компиляции? Что в них содержится?

2.7. Выполните команду:

*modinfo hello.ko*

Какая информация появится в терминале после выполнения данной команды?

2.8. Загрузите собранный модуль:

*sudo insmod hello.ko*

Проверьте, появился ли он в списке модулей операционной системы.

2.9. Убедитесь, что модуль добавил приветствие при загрузке:

*dmesg | tail* -1

2.10. Выгрузите модуль с помощью команды *rmmod*. Проверьте, появилось ли новое сообщение от модуля.

**Задание 3**. Исследуйте объектный файл модуля.

3.1. Форматом модуля является обычный объектный ELF формат, но дополненный в таблице внешних имён некоторыми дополнительными именами, такими как: \_\_mod\_author5, \_\_mod\_license4, \_\_mod\_srcversion23, \_\_module\_depends, \_\_mod\_vermagic5, … которые определяются специальными макросами.

Выполните команду:

*file hello.ko*

3.2. Изучите структуру секций объектного файла модуля. Выполните команду:

*objdump -h hello.ko*

Секции обозначаются следующим образом:

* .text — код модуля (инструкции);
* .init.text — код инициализации модуля;
* .exit.text — код завершения модуля;
* .modinfo — текст макросов модуля;
* .data — инициализированные данные;
* .bss — не инициализированные данные (Block Started Symbol).

3.3. Изучите список имён модуля, которые могут иметь локальную или глобальную видимость, и могут экспортироваться модулем. Найдите имена функций, описанных в коде модуля; рядом с ними указывается имя секции, в которой находятся эти имена. Выполните команду:

*objdump -t hello.ko*

3.4. Ещё один альтернативный инструмент детального анализа объектной структуры модуля (он даёт несколько иные срезы информации), хорошо видна сфера видимости имён (колонка Bind : LOCAL — имя не экспортируется за пределы кода модуля). Выполните команду:

*readelf -s hello.ko*

3.5. Убедитесь в том, что имя модуля закодировано в его объектном коде и не зависит от имени файла. Выполните команды:

*cp hello.ko hello-1.ko*

*sudo insmod hello-1.ko*

3.6. Выведите список модулей ядра. Выполните команду:

*lsmod | grep hello*

Под каким именем загружен модуль? Какой командой можно удалить модуль из ядра?

**Задание 4**. Изучите параметры модуля.

4.1. Изучите исходный код модуля *./parms/mod\_params.c.*

Модулю при его загрузке могут быть переданы значения параметров — здесь наблюдается полная аналогия (по смыслу, но не по формату) с передачей параметров пользовательскому процессу из командной строки через массив *argv[]*.  
Значения параметрам могут быть установлены во время загрузки модуля через *insmod* или *modprobe*; последняя команда также может прочитать значение параметров из своего файла конфигурации *(/etc/modprobe.conf*) для загрузки модулей.

Обработка входных параметров модуля обеспечивается макросами (описаны в *<linux/moduleparam.h>*):

module\_param(name, type, perm)

module\_param\_string( name, string, len, perm )

module\_param\_array( name, type, nump, perm )

Для параметров модуля в макросе *module\_param()* могут быть указаны следующие типы:

* bool, invbool – булева величина (true или false) – связанная переменная должна быть типа int. Тип invbool инвертирует значение, так что значение true приходит как false и наоборот.
* charp – значение указателя на char – выделяется память для строки, заданной пользователем (не нужно предварительно распределять место для строки), и указатель устанавливается соответствующим образом.
* int, long, short, uint, ulong, ushort – базовые целые величины разной размерности; версии, начинающиеся с u, являются беззнаковыми величинами.

В качестве входного параметра может быть определён и массив выше перечисленных типов (макрос module\_param\_array()).

Обратите внимание на следующие особенности, которые могут показаться непривычными программисту на языке С, нарушающие стереотипы этого языка, и поначалу именно в этом порождающие ошибки программирования в собственных модулях:

отсутствие резервирования памяти для символьного параметра sparam1;  
и динамический размер параметра-массива aparam. (динамически изменяющийся после загрузки модуля);

при этом этот динамический размер не может превысить статически зарезервированную максимальную размерность массива (такая попытка вызывает ошибку).

4.2. Выполните загрузку модуля с разными комбинациями параметров, например:

*sudo insmod mod\_params.ko iparam=3 nparam=4 bparam=1 sparam=str1 cparam=str2 aparam=5,4,3*

4.3. Изучите вывод модуля:

*dmesg | tail -n8*

Обратите внимание на то, что массив *aparam* получил и новую размерность *arnum*, и его элементам присвоены новые значения.

**Задание 5**. Изучите экспортирование символов.

5.1. Изучите исходный код модулей в каталоге ./export-data/.

5.2. Выполните сборку модулей.

5.3. Определите правильный порядок загрузки и выгрузки модулей.

5.4. Исследуйте свойства символов md1\_data и md1\_proc.