МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій

Кафедра комп'ютерної інженерії та інформаційних систем

ЛАБОРАТОРНА РОБОТА №7

з дисципліни «Теорія і технології проектування спеціалізованих операційних систем»

ЛРКІ.180112.22.01.17 ПЗ

Галузь знань \_\_\_\_\_\_\_\_\_\_12 – Інформаційні технології\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність \_\_\_\_\_\_\_\_\_123 –Комп’ютерна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Виконав:

студент 1 курсу, групи КІ2м-22-1 Біньковський Я.В.

(Підпис)

Перевірив: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лисенко С.М.

(Підпис)

Хмельницький – 2023 р

Тема:

Розроблення драйверів пристроїв як модулів ядра для ОС Linux. Створення драйвера як модуля ядра. Збирання ядра разом з модулем драйвера. Завдання:

Простий рівень.

Вивчити наведені приклади взаємодії з драйверами пристроїв. Реалізувати наведений в метод. вказівках приклад драйвера як модуль ядра. Зібрати ядро ОС разом з модулем драйвера. Оформити звіт з виконаної роботи.

Реалізація

Створення модуля ядра вимагає побудови за допомогою makefile. Makefile вказує системі збірки ядра, які компоненти необхідні для збирання модуля.

Якщо модуль складається з одного вихідного файлу, потрібен лише один рядок у makefile:

obj-m := source\_file\_name.o

Якщо є кілька вихідних файлів, потрібні два рядки у makefile:

obj-m := module\_name.o

module\_name-objs := source\_1.o source\_2.o … source\_n.o

Для збірки модуля потрібно виконати команду make -C KERNEL\_MODULE\_BUILD\_SYSTEM\_FOLDER M=pwd modules, де KERNEL\_MODULE\_BUILD\_SYSTEM\_FOLDER - це шлях до системи збірки модулів.

Для очищення папки збірки використовується команда make -C KERNEL\_MODULES\_BUILD\_SYSTEM\_FOLDER M=pwd clean.

Система збірки модулів зазвичай знаходиться в /lib/modules/uname -r/build.

Для завантаження модуля в ядро використовується команда insmod ./<module\_name>.ko, а для вивантаження - rmmod ./<module\_name>.ko.

Для завантаження та використання модуля потрібно виконати команду make load з папки вихідного файлу.

Після завантаження, ім'я драйвера буде відображено у файлі /proc/modules, а пристрій, пов'язаний з модулем, буде відображено у файлі /proc/devices.

Щоб створити файл пристрою, потрібно виконати команду mknod /dev/simple-driver c 250 0.

Для перевірки роботи драйвера можна використовувати команду cat /dev/simple-driver.

Код реалізації драйвера:

main.c

#include "device\_file.h"

#include <linux/init.h> /\* module\_init, module\_exit \*/

#include <linux/module.h> /\* version info, MODULE\_LICENSE, MODULE\_AUTHOR,

printk() \*/

MODULE\_DESCRIPTION("Simple Linux driver");

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("Apriorit, Inc");

static int simple\_driver\_init(void)

{

int result = 0;

printk( KERN\_NOTICE "Simple-driver: Initialization started\n" );

result = register\_device();

return result;

}

static void simple\_driver\_exit(void)

{

printk( KERN\_NOTICE "Simple-driver: Exiting\n" );

unregister\_device();

}

module\_init(simple\_driver\_init);

module\_exit(simple\_driver\_exit);

device\_file.h

#ifndef DEVICE\_FILE\_H\_

#define DEVICE\_FILE\_H\_

#include <linux/compiler.h> /\* \_\_must\_check \*/

\_\_must\_check int register\_device(void); /\* 0 if Ok\*/

void unregister\_device(void);

#endif //DEVICE\_FILE\_H\_

device\_file.c

#include "device\_file.h"

#include <linux/fs.h> /\* file stuff \*/

#include <linux/kernel.h> /\* printk() \*/

#include <linux/errno.h> /\* error codes \*/

#include <linux/module.h> /\* THIS\_MODULE \*/

#include <linux/cdev.h> /\* char device stuff \*/

#include <linux/uaccess.h> /\* copy\_to\_user() \*/

static const char g\_s\_Hello\_World\_string[] = "Hello world from kernel

mode!\n\0";

static const ssize\_t g\_s\_Hello\_World\_size = sizeof(g\_s\_Hello\_World\_string);

static ssize\_t device\_file\_read(

struct file \*file\_ptr

, char \_\_user \*user\_buffer

, size\_t count

, loff\_t \*possition)

{

printk( KERN\_NOTICE "Simple-driver: Device file is read at offset = %i, read

bytes count = %u\n"

, (int)\*possition

, (unsigned int)count );

if( \*possition >= g\_s\_Hello\_World\_size )

return 0;

if( \*possition + count > g\_s\_Hello\_World\_size )

count = g\_s\_Hello\_World\_size - \*possition;

if( copy\_to\_user(user\_buffer, g\_s\_Hello\_World\_string + \*possition, count) !=

0 )

return -DEFAULT;

\*possition += count;

return count;

}

static struct file\_operations simple\_driver\_fops =

{

.owner = THIS\_MODULE,

.read = device\_file\_read,

};

static int device\_file\_major\_number = 0;

static const char device\_name[] = "Simple-driver";

int register\_device(void)

{

int result = 0;

printk( KERN\_NOTICE "Simple-driver: register\_device() is called.\n" );

result = register\_chrdev( 0, device\_name, &simple\_driver\_fops );

if( result < 0 )

{

printk( KERN\_WARNING "Simple-driver: can\'t register character device

with errorcode = %i\n", result );

return result;

}

device\_file\_major\_number = result;

printk( KERN\_NOTICE "Simple-driver: registered character device with major

number = %i and minor numbers 0...255\n"

, device\_file\_major\_number );

return 0;

}

void unregister\_device(void)

{

printk( KERN\_NOTICE "Simple-driver: unregister\_device() is called\n" );

if(device\_file\_major\_number != 0)

{

unregister\_chrdev(device\_file\_major\_number, device\_name);

}

}

makefile

TARGET\_MODULE:=simple-module

# If we running by kernel building system

ifneq ($(KERNELRELEASE),)

$(TARGET\_MODULE)-objs := main.o device\_file.o

obj-m := $(TARGET\_MODULE).o

# If we are running without kernel build system

else

BUILDSYSTEM\_DIR:=/lib/modules/$(shell uname -r)/build

PWD:=$(shell pwd)

all :

# run kernel build system to make module

$(MAKE) -C $(BUILDSYSTEM\_DIR) M=$(PWD) modules

clean:

# run kernel build system to cleanup in current directory

$(MAKE) -C $(BUILDSYSTEM\_DIR) M=$(PWD) clean

load:

insmod ./$(TARGET\_MODULE).ko

unload:

rmmod ./$(TARGET\_MODULE).ko

endif