|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

|  |  |
| --- | --- |
| **по лабораторной работе №** |  |

**Дисциплина:  *Операционные системы***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ***ИУ7И-66Б*** |  |  | **Нгуен Ф. С.** |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | **Рязанова Н. Ю.** |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2021*

**код программы**

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/vmalloc.h>

#include <linux/proc\_fs.h>

#include <asm/uaccess.h>

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

MODULE\_AUTHOR**(**"Nguyensanghso@gmail.com"**);**

#define OK 0

#define FORTUNE\_DIRNAME "Fortdir"

#define FORTUNE\_FILENAME "fortfile"

#define FORTUNE\_SYMLINK "fortune\_symlink"

#define FORTUNE\_PATH FORTUNE\_DIRNAME "/" FORTUNE\_FILENAME

#define MAX\_COOKIE\_BUF\_SIZE PAGE\_SIZE

#define KERN\_LOG\_MSG() { printk(KERN\_INFO "FORTUNE\_MODULE: %s called.\n", \_\_func\_\_); }

#define KERN\_ERR\_MSG(err) { printk(KERN\_ERR "FORTUNE\_MODULE: %s.\n", err); }

#define KERN\_INFO\_MSG(msg) { printk(KERN\_INFO "FORTUNE\_MODULE: %s.\n", msg); }

static struct proc\_dir\_entry **\***fortune\_dir**,** **\***fortune\_file**,** **\***fortune\_symlink **=** **NULL;**

static char **\***cookie\_buffer **=** **NULL;**

static int read\_index **=** 0**;**

static int write\_index **=** 0**;**

char tmp\_buffer**[**MAX\_COOKIE\_BUF\_SIZE**];**

static int fortune\_open(struct inode \*sp\_inode, struct file \*sp\_file)

**{**

KERN\_LOG\_MSG**();**

**return** OK**;**

**}**

static int fortune\_release(struct inode \*sp\_node, struct file \*sp\_file)

**{**

KERN\_LOG\_MSG**();**

**return** OK**;**

**}**

static ssize\_t fortune\_write(struct file \*file, const char \_\_user \*buf, size\_t len, loff\_t \*ppos)

**{**

KERN\_LOG\_MSG**();**

**if** **(**len **>** MAX\_COOKIE\_BUF\_SIZE **-** write\_index **+** 1**)**

**{**

KERN\_ERR\_MSG**(**"Buffer overflow"**);**

**return** **-**ENOSPC**;**

**}**

**if** **(**copy\_from\_user**(&**cookie\_buffer**[**write\_index**],** buf**,** len**)** **!=** 0**)**

**{**

KERN\_ERR\_MSG**(**"copy\_from\_user function get a error"**);**

**return** **-**EFAULT**;**

**}**

write\_index **+=** len**;**

cookie\_buffer**[**write\_index **-** 1**]** **=** '\0'**;**

**return** len**;**

**}**

**static ssize\_t fortune\_read(struct file \*file, char \_\_user \*buf, size\_t len, loff\_t \*f\_pos)**

**{**

KERN\_LOG\_MSG**();**

**if** **(\***f\_pos **>** 0 **||** write\_index **==** 0**)**

**{**

**return** 0**;**

**}**

**if** **(**read\_index **>=** write\_index**)**

**{**

read\_index **=** 0**;**

**}**

int read\_len **=** snprintf**(**tmp\_buffer**,** MAX\_COOKIE\_BUF\_SIZE**,** "%s\n"**,** **&**cookie\_buffer**[**read\_index**]);**

**if** **(**copy\_to\_user**(**buf**,** tmp\_buffer**,** read\_len**)** **!=** 0**)**

**{**

KERN\_ERR\_MSG**(**"copy\_to\_user function get a error"**)**

**return** **-**EFAULT**;**

**}**

read\_index **+=** read\_len**;**

**\***f\_pos **+=** read\_len**;**

**return** read\_len**;**

**}**

static const struct proc\_ops fops =

**{**

proc\_read**:** fortune\_read**,**

proc\_write**:** fortune\_write**,**

proc\_open**:** fortune\_open**,**

proc\_release**:** fortune\_release**,**

**};**

static void cleanup\_fortune(void)

**{**

KERN\_LOG\_MSG**();**

**if** **(**fortune\_symlink **!=** **NULL)**

**{**

remove\_proc\_entry**(**FORTUNE\_SYMLINK**,** **NULL);**

**}**

**if** **(**fortune\_file **!=** **NULL)**

**{**

remove\_proc\_entry**(**FORTUNE\_FILENAME**,** fortune\_dir**);**

**}**

**if** **(**fortune\_dir **!=** **NULL)**

**{**

remove\_proc\_entry**(**FORTUNE\_DIRNAME**,** **NULL);**

**}**

vfree**(**cookie\_buffer**);**

**}**

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

**{**

KERN\_LOG\_MSG**();**

**if** **((**cookie\_buffer **=** vzalloc**(**MAX\_COOKIE\_BUF\_SIZE**))** **==** **NULL)**

**{**

KERN\_ERR\_MSG**(**"Allocate memory error."**);**

**return** **-**ENOMEM**;**

**}**

**if** **((**fortune\_dir **=** proc\_mkdir**(**FORTUNE\_DIRNAME**,** **NULL))** **==** **NULL)**

**{**

KERN\_ERR\_MSG**(**"Error during create directory in proc"**);**

cleanup\_fortune**();**

**return** **-**ENOMEM**;**

**}**

**if** **((**fortune\_file **=** proc\_create**(**FORTUNE\_FILENAME**,** 0666**,** fortune\_dir**,** **&**fops**))** **==** **NULL)**

**{**

KERN\_ERR\_MSG**(**"Error during create file in proc"**);**

cleanup\_fortune**();**

**return** **-**ENOMEM**;**

**}**

**if** **((**fortune\_symlink **=** proc\_symlink**(**FORTUNE\_SYMLINK**,** **NULL,** FORTUNE\_PATH**))** **==** **NULL)**

**{**

KERN\_ERR\_MSG**(**"Error during create symlink in proc"**);**

cleanup\_fortune**();**

**return** **-**ENOMEM**;**

**}**

KERN\_INFO\_MSG**(**"Module has benn successfully loaded.\n"**);**

**return** OK**;**

**}**

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

**{**

KERN\_LOG\_MSG**();**

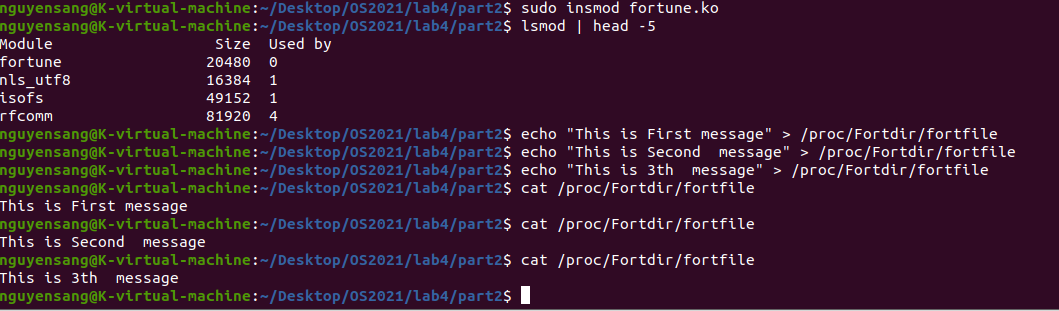
cleanup\_fortune**();**

KERN\_INFO\_MSG**(**"Module has been successfully removed"**);**

**}**

module\_init**(**fortune\_init**);**

module\_exit**(**fortune\_exit**);**

****

****