**Федеральное государственное автономное
образовательное учреждение высшего образования**

«Национальный исследовательский университет ИТМО»

факультет программной инженерии и компьютерной техники

Дисциплина: Операционные системы

Лабораторная работа №2

Выполнил:
Перевозчиков Иван Сергеевич
группа P33301

г. Санкт-Петербург
2022

Вариант:

Интерфейс передачи: debugfs

Целевые структуры: inode, vfsmount

User.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define BUFFER_SIZE 1024

int main(int argc, char *argv[]) {
    FILE *file = fopen("/sys/kernel/debug/kmod_dir/kmod_file", "r+");
    if (file != NULL)  {
        char *path[BUFFER_SIZE];
        if (sscanf(argv[1], "%s", (char *) path)) {
            char *buffer[BUFFER_SIZE];
            fprintf(file, "path: %s", (char *) path);
            while (true) {
                char *result = fgets((char *) buffer, BUFFER_SIZE, file);
                if (feo          f(file))
                    break;
                printf("%s", result);
            }
        } else {
            printf("Wrong arguments. Try again.");
        }
        fclose(file);
    } else {
        printf("File is not found.");
    }

    return 0;
}
```

kmod.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/debugfs.h>
#include <linux/fs.h>
#include <linux/namei.h>

#define BUFFER_SIZE 1024

MODULE_LICENSE("GPL");
MODULE_AUTHOR("IP13");
MODULE_DESCRIPTION("Lab");

static struct dentry *kmod_dir;
static struct dentry *kmod_file;
static struct inode *inode;
static struct vfsmount *vfsmount;


static struct inode *get_inode(char *path_name) {
    static struct path path;
    kern_path(path_name, LOOKUP_FOLLOW, &path);
    return path.dentry->d_inode;
}


static struct vfsmount *get_vfsmount(char *path_name) {
    static struct path path;
    kern_path(path_name, LOOKUP_FOLLOW, &path);
    return path.mnt;
}


static void print_inode(struct seq_file *file, struct inode *inode) {
    if (inode == NULL) {
        seq_printf(file, "Inode was not found\n");
    } else {
        seq_printf(file, "Inode number: %lu\n", inode->i_ino);
        seq_printf(file, "Inode permissions: %hu\n", inode->i_mode);
        seq_printf(file, "Inode number of hard links: %u\n", inode->i_nlink);
    }
```

```
}


static void print_vfsmount(struct seq_file *file, struct vfsmount *vfsmount) {
    if (vfsmount == NULL) {
        seq_printf(file, "Vfsmount was not found\n");
    } else {
        seq_printf(file, "Vfs flags: %i\n", vfsmount->mnt_flags);
        seq_printf(file, "Vfs' size of super_block: %lu\n", vfsmount->mnt_sb->s_blocksize);
        seq_printf(file, "Vfs's dentry flags: %lu\n", vfsmount->mnt_root->d_flags);
    }
}


static int print_to_user(struct seq_file *file, void *data) {
    print_inode(file, inode);
    print_vfsmount(file, vfsmount);
    return 0;
}


static ssize_t kmod_write(struct file *file, const char __user *buffer, size_t length, loff_t
*ptr_offset) {
    printk(KERN_INFO "kmod_write activated.\n");

    char user_data[BUFFER_SIZE];
    copy_from_user(user_data, buffer, length);

    char path[BUFFER_SIZE];
    sscanf(user_data, "path: %s", &path);

    inode = get_inode(path);
    vfsmount = get_vfsmount(path);

    single_open(file, print_to_user, NULL);

    return strlen(user_data);
}


static struct file_operations fops = {
        .read = seq_read,
        .write = kmod_write,
};


static int __init mod_init(void) {
    printk(KERN_INFO "kmod loaded.\n");
```

```
    kmod_dir = debugfs_create_dir("kmod_dir", NULL);
    kmod_file = debugfs_create_file("kmod_file", 0777, kmod_dir, NULL, &fops);
    return 0;
}


static void __exit mod_exit(void) {
    debugfs_remove_recursive(kmod_dir);
    printk(KERN_INFO "kmod removed\n");
}


module_init(mod_init);
module_exit(mod_exit);
```

Makefile

```
kmod=kmod
user=user
path=/home/ivan/Documents/OS1/Makefile
obj-m += $(kmod).o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
	make user
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
	make rm_user
insmod:
	sudo insmod $(kmod).ko
rmmod:
	sudo rmmod $(kmod).ko
user:
	gcc -o $(user) $(user).c
rm_user:
	rm $(user)
runmod:
	sudo ./$(user) $(path)
```

Вывод программы:
Inode number: 393242
Inode permissions: 33204
Inode number of hard links: 1
Vfs flags: 4128
Vfs' size of super_block: 4096
Vfs's dentry flags: 2162688