



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»

**Лабораторная работа № 8**

Тема **Создание виртуальной файловой системы**

Студент Сушина А.Д.

Группа ИУ7-616

Оценка (баллы) \_\_\_\_\_

Преподаватель Рязанова Н.Ю.

Москва.  
2020 г

## Текст программы

На листинге 1 представлен текст программы slab.c

Листинг 1. slab.c

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/fs.h>
#include <linux/time.h>
#include <linux/slab.h>

#define MYFS_MAGIC_NUMBER 0x13131313;
#define SLABNAME "my_cache"
struct myfs_inode
{
    int i_mode;
    unsigned long i_ino;
};

int inode_number = 0;
static struct kmem_cache *cache;

static void myfs_put_super(struct super_block *sb)
{
    printk(KERN_DEBUG "myfs super block destroyed\n");
}

int free_alloc_inodes(struct inode *inode)
{
    kmem_cache_free(cache, inode->i_private);
    return 1;
}

static struct super_operations const myfs_super_ops = {
    .put_super = myfs_put_super,
    .statfs = simple_statfs,
    .drop_inode = free_alloc_inodes,
};

static struct inode *myfs_make_inode(struct super_block *sb, int mode)
{
    struct inode *ret = new_inode(sb);
    if (ret)
    {
        struct myfs_inode *my_inode = kmem_cache_alloc(cache, GFP_KERNEL);
        inode_init_owner(ret, NULL, mode);
        *my_inode = (struct myfs_inode){
            .i_mode = ret->i_mode,
            .i_ino = ret->i_ino
        };
        ret->i_size = PAGE_SIZE;
        ret->i_atime = ret->i_mtime = ret->i_ctime = current_time(ret);
        ret->i_private = my_inode;
    }
    return ret;
}
```

```

static int myfs_fill_sb(struct super_block *sb, void *data, int silent)
{
    struct inode *root = NULL;

    sb->s_blocksize = PAGE_SIZE;
    sb->s_blocksize_bits = PAGE_SHIFT;
    sb->s_magic = MYFS_MAGIC_NUMBER;
    sb->s_op = &myfs_super_ops;

    root = myfs_make_inode(sb, S_IFDIR|0755);
    if (!root)
    {
        printk(KERN_ERR "myfs inode allocation failed\n");
        return -ENOMEM;
    }
    root->i_op = &simple_dir_inode_operations;
    root->i_fop = &simple_dir_operations;

    sb->s_root = d_make_root(root);

    if (!sb->s_root)
    {
        printk(KERN_ERR "myfs root creation failed\n");
        iput(root);
        return -ENOMEM;
    }

    return 0;
}

static struct dentry* myfs_mount(struct file_system_type * type, int flags,
char const *dev, void *data)
{
    struct dentry *const entry = mount_nodev(type, flags, data, myfs_fill_sb);
    if (IS_ERR(entry))
        printk(KERN_ERR "myfs mounting failed!\n");
    else
        printk(KERN_DEBUG "myfs mounted");
    return entry;
}

static struct file_system_type myfs_type = {
    .owner = THIS_MODULE,
    .name = "myfs",
    .mount = myfs_mount,
    .kill_sb = kill_litter_super,
};

void co (void *p)
{
    *(int *)p = (int)p;
    inode_number++;
}

static int __init myfs_init(void)
{
    int ret = register_filesystem(&myfs_type);
    cache = kmem_cache_create(SLABNAME, sizeof(struct myfs_inode), 0, 0, co);
    if (ret != 0)

```

```

    {
        printk(KERN_ERR "myfs can't register filesystem\n");
        return ret;
    }
    printk(KERN_INFO "myfs filesystem registered");
    return 0;
}

static void __exit myfs_exit(void)
{
    int ret = unregister_filesystem(&myfs_type);
    if (ret != 0)
        printk(KERN_ERR "myfs can't unregister filesystem!\n");
    kmem_cache_destroy(cache);
    printk(KERN_INFO "myfs unregistered %d", inode_number);
}

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Sushina Anastasia");

module_init(myfs_init);
module_exit(myfs_exit);

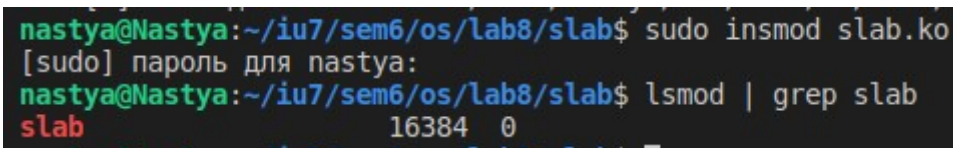
```

Если создается виртуальная файловая система, не связанная с каким-либо носителем, нет необходимости использовать функцию `mount_bdev()`. Вместо нее можно использовать функцию `mount_nodev()`.

### Демонстрация работы программы

Соберем и загрузим полученный модуль ядра командой `insmod`.

Рис 1. Загрузка модуля ядра



```

nastya@Nastya:~/iu7/sem6/os/lab8/slab$ sudo insmod slab.ko
[sudo] пароль для nastya:
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ lsmod | grep slab
slab                16384  0

```

Создадим образ диска командой:

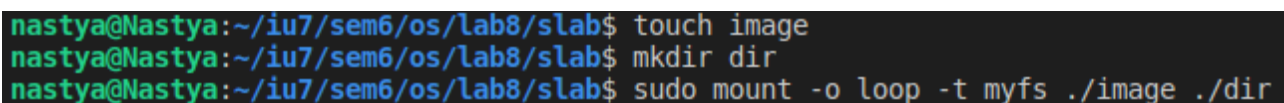
```
touch image
```

Кроме того, нужно создать каталог, который будет точкой монтирования (корнем) файловой системы:

```
mkdir dir
```

Теперь, используя этот образ, примонтируем файловую систему:

```
sudo mount -o loop -t myfs ./image ./dir
```



```

nastya@Nastya:~/iu7/sem6/os/lab8/slab$ touch image
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ mkdir dir
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ sudo mount -o loop -t myfs ./image ./dir

```

Рис 2. Монтирование файловой системы

Состояние Slab после монтирования:

```
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ sudo cat /proc/slabinfo | grep my_cache
my_cache 170 170 24 170 1 : tunables 0 0 0 : sl
abdata 1 1 0
```

Рис 3. Состояние slab после монтирования

Информация о смонтированной файловой системе:

```
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ cat /proc/filesystems | grep myfs
nodev myfs
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ mount | grep myfs
/dev/loop25 on /home/nastya/iu7/sem6/os/lab8/slab/dir type myfs (rw,relatime)
```

Рис 4. Состояние файловой системы после монтирования

Чтобы размонтировать файловую систему делаем так:

`sudo umount ./dir`

```
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ sudo umount ./dir
```

Рис 5. Размонтирование файловой системы

Проверяем системный лог.

Рис 6. 

```
nastya@Nastya:~/iu7/sem6/os/lab8/slab$ dmesg | grep myfs
[ 952.522833] myfs filesystem registered
[ 1116.502934] myfs mounted
[ 1329.796955] myfs super block destroyed
[ 1515.229620] myfs unregistered 170
```

Состояние системного лога