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»

Дисциплина: «Операционные системы»

Лабораторная работа №8

Тема работы: «Создание виртуальной файловой системы»

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватель: Рязанова Н. Ю.

Москва, 2020 г.

Создать файловую систему в виде загружаемого модуля ядра, а также slab-кэш для хранения inode созданной файловой системы.

Листинг кода программы

Ниже приведена программа, реализующая поставленную задачу.

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

static const unsigned long MYFS_MAGIC_NUMBER = 0x13131313;

struct kmem_cache *myfs_cache = NULL;

struct myfs_inode
{
        int i_mode;
        unsigned long i_ino;
};

static void myfs_put_super(struct super_block *sb)
{
    printk(KERN_INFO "MYFS super block destroyed!\n");
}

static int myfs_drop_inode(struct inode *inode)
{
        kmem_cache_free(myfs_cache, inode->i_private);
        return 1;
}

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

```
static struct inode *myfs_make_inode(struct super_block *sb, int mode)
{
    struct inode *ret = new_inode(sb);

    if (ret) {
        inode_init_owner(ret, NULL, mode);
        ret->i_size = PAGE_SIZE;
        ret->i_atime = ret->i_mtime = ret->i_ctime = current_time(ret);
        struct myfs_inode *myfs_inode = (struct myfs_inode *)
        kmem_cache_alloc(myfs_cache, GFP_KERNEL);
        myfs_inode->i_mode = ret->i_mode;
        myfs_inode->i_ino = ret->i_ino;
        ret->i_private = myfs_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");
        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_bdev(type, flags, dev, data,
    myfs_fill_sb);
    if (IS_ERR(entry))
        printk(KERN_ERR "MYFS mounting failed!\n");
    else
        printk(KERN_DEBUG "MYFS mounted!\n");
    return entry;
}

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

static int __init myfs_init(void)
{
        int ret = register_filesystem(&myfs_type);
        if (ret != 0)
        {
            printk(KERN_ERR "MYFS_MODULE cannot register filesystem!\n");
            return ret;
        }

        myfs_cache = kmem_cache_create("myfs_inode_cache",
        sizeof(struct myfs_inode), 0, SLAB_POISON, NULL);
        if (!myfs_cache)
        {
                printk(KERN_ERR "kmem_cache_create error\n");
                kmem_cache_destroy(myfs_cache);
                return -ENOMEM;
        }
```

```
        printk(KERN_DEBUG "MYFS inode cache created!\n");
            printk(KERN_DEBUG "MYFS_MODULE loaded\n");
            return 0;
}

static void __exit myfs_exit(void)
{
        int ret = unregister_filesystem(&myfs_type);

        if (ret != 0)
        {
                printk(KERN_ERR
                "MYFS_MODULE cannot unregister filesystem!\n");
        }
        else
        {
                kmem_cache_destroy(myfs_cache);
                printk(KERN_DEBUG "MYFS inode cache destroyed!\n");
                printk(KERN_DEBUG "MYFS module unloaded!\n");
        }
}

module_init(myfs_init);
module_exit(myfs_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Levushkin");
```

Сценарий сборки программы

```
ifneq ($(KERNELRELEASE),)
        obj-m := myfs.o

else
        CURRENT = $(shell uname -r)
        KDIR = /lib/modules/$(CURRENT)/build
        PWD = $(shell pwd)

default:
        $(MAKE) -C $(KDIR) M=$(PWD) modules
        sudo make clean
```

```
clean:
	rm -rf .tmp_versions
	rm .myfs.*
	#rm *.ko
	rm *.o
	rm *.mod.c
	rm *.symvers
	rm *.order

endif
```

Демонстрация работы программы

Ниже представлены загрузка модуля, создание slab-кэша для inode, созданный slab-кэш, создание образа диска, корневой директории и монтирование файловой системы:

```
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo insmod myfs.ko
ilalevuskin@ubuntu:~/Desktop/8_os$ dmesg | grep MYFS | tail -2
[   77.018025] MYFS inode cache created!
[   77.018027] MYFS_MODULE loaded
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo cat /proc/slabinfo | grep myfs
myfs_inode_cache       0      0     24  170    1 : tunables    0    0    0 : slabdata      0      0      0
ilalevuskin@ubuntu:~/Desktop/8_os$ touch image
ilalevuskin@ubuntu:~/Desktop/8_os$ mkdir ./myfs_root
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo mount -o loop -t myfs ./image ./myfs_root
ilalevuskin@ubuntu:~/Desktop/8_os$ dmesg | grep MYFS | tail -3
[   77.018025] MYFS inode cache created!
[   77.018027] MYFS_MODULE loaded
[  138.434052] MYFS mounted!
```

Был создан slab-кэш, содержащий 0 активных объектов, с размером каждого объекта, равным 24 байта.

Ниже представлено состояние slab-кэша после монтирования файловой системы:

```
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo cat /proc/slabinfo | grep myfs
myfs_inode_cache       1    170     24  170    1 : tunables    0    0    0 : slabdata      1      1      0
```

slab-кэш содержит 1 элемент, поскольку при монтировании файловой системы создается inode для корневой директории.

Размонтирование файловой системы и состояние slab-кэша после нее:

```
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo umount ./myfs_root
ilalevuskin@ubuntu:~/Desktop/8_os$ dmesg | grep MYFS | tail -4
[   77.018025] MYFS inode cache created!
[   77.018027] MYFS_MODULE loaded
[  138.434052] MYFS mounted!
[  174.195558] MYFS super block destroyed!
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo cat /proc/slabinfo | grep myfs
myfs_inode_cache       0    170     24  170    1 : tunables    0    0    0 : slabdata      1      1      0
```

Удаление slab-кэша и выгрузка модуля:

```
ilalevuskin@ubuntu:~/Desktop/8_os$ sudo rmmod myfs
ilalevuskin@ubuntu:~/Desktop/8_os$ dmesg | grep MYFS | tail -6
[   77.018025] MYFS inode cache created!
[   77.018027] MYFS_MODULE loaded
[  138.434052] MYFS mounted!
[  174.195558] MYFS super block destroyed!
[  199.576038] MYFS inode cache destroyed!
[  199.576039] MYFS module unloaded!
```