



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»

ОТЧЁТ

По лабораторной работе №8

По курсу: «Операционные системы»

Тема: «Создание виртуальной файловой системы»

Студент: Керимов А. Ш.

Группа: ИУ7-64Б

Преподаватель: Рязанова Н. Ю.

Москва

2020

Виртуальная файловая система

Листинг 1: myfs.c

```
1 #include <linux/fs.h>
2 #include <linux/kernel.h>
3 #include <linux/module.h>
4 #include <linux/slab.h>
5
6 MODULE_LICENSE("GPL");
7 MODULE_AUTHOR("Kerimov A. IU7-64b");
8 MODULE_DESCRIPTION("my filesystem");
9
10 #define MYFS_MAGIC_NUMBER 0xDEADBEAF
11 #define SLABNAME "myfs_inode_cache"
12
13 struct myfs_inode {
14     umode_t      i_mode;
15     unsigned long i_ino;
16 };
17
18 static struct kmem_cache *myfs_inode_cachep = NULL;
19 static struct myfs_inode **myfs_inodes = NULL;
20 static int cache_inode_count = 128;
21 module_param(cache_inode_count, int, 0);
22 static int busy = 0;
23
24 static struct myfs_inode *cache_get_inode(void)
25 {
26     if (busy == cache_inode_count) {
27         return NULL;
28     }
29     return myfs_inodes[busy++] = kmem_cache_alloc(myfs_inode_cachep, GFP_KERNEL);
30 }
31
32 static struct inode *myfs_make_inode(struct super_block *sb, int mode)
33 {
34     struct myfs_inode *myfs_inodep = NULL;
35     struct inode *ret = new_inode(sb);
36     if (ret) {
37         inode_init_owner(ret, NULL, mode);
38         ret->i_size = PAGE_SIZE;
39         ret->i_atime = ret->i_mtime = ret->i_ctime = current_time(ret);
40         if ((myfs_inodep = cache_get_inode())) {
41             myfs_inodep->i_mode = ret->i_mode;
42             myfs_inodep->i_ino = ret->i_ino;
43         }
44         ret->i_private = myfs_inodep;
45     }
46     return ret;
47 }
48
49 static void myfs_put_super(struct super_block *sb)
50 {
51     printk(KERN_DEBUG "myfs: super block destroyed\n");
52 }
53
54 static const struct super_operations myfs_super_ops = {
55     .put_super = myfs_put_super,
56     .statfs = simple_statfs,
57     .drop_inode = generic_delete_inode,
58 };
59
60 static int myfs_fill_sb(struct super_block *sb, void *data, int silent)
61 {
62     struct inode *root = NULL;
63     sb->s_blocksize = PAGE_SIZE;
64     sb->s_blocksize_bits = PAGE_SHIFT;
65     sb->s_magic = MYFS_MAGIC_NUMBER;
66     sb->s_op = &myfs_super_ops;
67     if (!(root = myfs_make_inode(sb, S_IFDIR | 0755))) {
68         printk(KERN_ERR "myfs: inode allocation failed\n");
69         return -ENOMEM;
70     }
71 }
```

```

74     }
75     root->i_op = &simple_dir_inode_operations;
76     root->i_fop = &simple_dir_operations;
77
78     if (!(sb->s_root = d_make_root(root))) {
79         iput(root);
80         printk(KERN_ERR "myfs: root creation failed\n");
81         return -ENOMEM;
82     }
83
84     return 0;
85 }
86
87 static struct dentry *myfs_mount(struct file_system_type *type, int flags, __attribute__((unused))
88     const char *dev, void *data)
89 {
90     struct dentry *entry = mount_nodev(type, flags, data, myfs_fill_sb);
91     if (IS_ERR(entry)) {
92         printk(KERN_ERR "myfs: mounting failed\n");
93     } else {
94         printk(KERN_DEBUG "myfs: mounted\n");
95     }
96     return entry;
97 }
98
99 static struct file_system_type myfs_type = {
100     .owner    = THIS_MODULE,
101     .name     = "myfs",
102     .mount    = myfs_mount,
103     .kill_sb  = kill_anon_super,
104 };
105
106 static int __init myfs_init(void)
107 {
108     int ret = register_filesystem(&myfs_type);
109     if (ret) {
110         printk(KERN_ERR "myfs: cannot register filesystem\n");
111         return ret;
112     }
113
114     if (!(myfs_inodes = kmalloc(sizeof(struct myfs_inode *) * cache_inode_count, GFP_KERNEL))) {
115         printk(KERN_ERR "myfs: kmalloc error\n");
116         return -ENOMEM;
117     }
118
119     if (!(myfs_inode_cachep = kmem_cache_create(SLABNAME, sizeof(struct myfs_inode), 0,
120         SLAB_POISON, NULL))) {
121         kfree(myfs_inodes);
122         printk(KERN_ERR "myfs: kmem_cache_create error\n");
123         return -ENOMEM;
124     }
125
126     printk(KERN_DEBUG "myfs: module loaded\n");
127     return 0;
128 }
129
130 static void __exit myfs_exit(void)
131 {
132     int i;
133     for (i = 0; i < busy; ++i) {
134         kmem_cache_free(myfs_inode_cachep, myfs_inodes[i]);
135     }
136     kmem_cache_destroy(myfs_inode_cachep);
137     kfree(myfs_inodes);
138
139     if (unregister_filesystem(&myfs_type)) {
140         printk(KERN_ERR "myfs: cannot unregister filesystem\n");
141     }
142     printk(KERN_DEBUG "myfs: module unloaded\n");
143 }
144
145 module_init(myfs_init)
146 module_exit(myfs_exit)

```

```
user@lenovo: ~/bmstu/OS/sem06/lab08
user@lenovo:~/bmstu/OS/sem06/lab08$ sudo insmod myfs.ko
user@lenovo:~/bmstu/OS/sem06/lab08$ dmesg | tail -1
[11782.984925] myfs: module loaded
user@lenovo:~/bmstu/OS/sem06/lab08$ touch image
user@lenovo:~/bmstu/OS/sem06/lab08$ mkdir dir
user@lenovo:~/bmstu/OS/sem06/lab08$ sudo mount -o loop -t myfs image dir
user@lenovo:~/bmstu/OS/sem06/lab08$ dmesg | tail -1
[11831.718848] myfs: mounted
user@lenovo:~/bmstu/OS/sem06/lab08$ ls -al
total 148
drwxr-xr-x 6 user user 4096 мая 5 19:36 .
drwxr-xr-x 9 user user 4096 мая 3 21:43 ..
drwxr-xr-x 3 user user 4096 мая 3 20:41 cmake-build-debug
-rw-r--r-- 1 user user 206 мая 3 20:41 CMakeLists.txt
drwxr-xr-x 1 root root 4096 мая 5 19:36 dir
drwxr-xr-x 4 user user 4096 мая 3 21:06 doc
drwxr-xr-x 3 user user 4096 мая 5 19:31 .idea
-rw-rw-r-- 1 user user 0 мая 5 19:36 image
-rw-r--r-- 1 user user 191 апр 27 19:35 Makefile
-rw-rw-r-- 1 user user 40 мая 5 19:35 modules.order
-rw-rw-r-- 1 user user 0 мая 5 17:52 Module.symvers
-rw-r--r-- 1 user user 3515 мая 5 19:31 myfs.c
-rw-rw-r-- 1 user user 9488 мая 5 19:31 myfs.ko
-rw-rw-r-- 1 user user 270 мая 5 19:31 .myfs.ko.cmd
-rw-rw-r-- 1 user user 40 мая 5 19:31 myfs.mod
-rw-rw-r-- 1 user user 560 мая 5 19:31 myfs.mod.c
-rw-rw-r-- 1 user user 147 мая 5 19:31 .myfs.mod.cmd
-rw-rw-r-- 1 user user 2792 мая 5 19:31 myfs.mod.o
-rw-rw-r-- 1 user user 31040 мая 5 19:31 .myfs.mod.o.cmd
-rw-rw-r-- 1 user user 7624 мая 5 19:31 myfs.o
-rw-rw-r-- 1 user user 38011 мая 5 19:31 .myfs.o.cmd
user@lenovo:~/bmstu/OS/sem06/lab08$ sudo cat /proc/slabinfo | grep myfs
myfs_inode_cache 1 170 24 170 1 : tunables 0 0 0 : slabdata 1 1 0
user@lenovo:~/bmstu/OS/sem06/lab08$ sudo umount dir
user@lenovo:~/bmstu/OS/sem06/lab08$ dmesg | tail -1
[11967.811093] myfs: super block destroyed
user@lenovo:~/bmstu/OS/sem06/lab08$ sudo rmmod myfs
user@lenovo:~/bmstu/OS/sem06/lab08$ dmesg | tail -1
[11977.262221] myfs: module unloaded
user@lenovo:~/bmstu/OS/sem06/lab08$
```

Рис. 1: Демонстрация работы программы