



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 8

Дисциплина	Операционные системы.
Тема	Создание виртуальной файловой системы.
Студент	Степанов А. О.
Группа	ИУ7-63Б
Оценка (баллы)	
Преподаватель	Рязанова Н.Ю.

Москва, 2020 г.

Листинг 1: Текст программы

```
1 #include <linux/module.h>
2 #include <linux/kernel.h>
3 #include <linux/init.h>
4 #include <linux/fs.h>
5 #include <linux/time.h>
6 #include <linux/slab.h>
7 #include <linux/version.h>
8
9 #define wfs_MAGIC_NUMBER 0x13131313
10 #define SLABNAME "wfs_cache"
11
12 MODULE_LICENSE("GPL");
13 MODULE_AUTHOR("Alexander Stepanov");
14 MODULE_DESCRIPTION("BMSTU_operating_systems_VFS");
15
16 static int size = 7;
17 module_param(size, int, 0);
18 static int number = 31;
19 module_param(number, int, 0);
20 static int sco = 0;
21
22 static void* *line = NULL;
23
24 static void co(void* p)
25 {
26     *(int*)p = (int)p;
27     sco++;
28 }
29
30 struct kmem_cache *cache = NULL;
31
32 static struct wfs_inode
33 {
34     int i_mode;
35     unsigned long i_ino;
36 } wfs_inode;
37
38 static struct inode *wfs_make_inode(struct super_block *sb, int mode)
39 {
40     struct inode *ret = new_inode(sb);
41
```

```

42     if (ret)
43     {
44         inode_init_owner(ret , NULL, mode);
45         ret->i_size = PAGE_SIZE;
46         ret->i_atime = ret->i_mtime = ret->i_ctime = current_time(ret);
47         ret->i_private = &wfs_inode;
48     }
49
50     return ret;
51 }
52
53 static void wfs_put_super(struct super_block * sb)
54 {
55     printk(KERN_DEBUG "[wfs]_super_block_destroyed!\n");
56 }
57
58 static struct super_operations const wfs_super_ops = {
59     .put_super = wfs_put_super,
60     .statfs = simple_statfs,
61     .drop_inode = generic_delete_inode,
62 };
63
64 static int wfs_fill_sb(struct super_block *sb, void *data, int silent)
65 {
66     struct inode* root = NULL;
67
68     sb->s_blocksize = PAGE_SIZE;
69     sb->s_blocksize_bits = PAGE_SHIFT;
70     sb->s_magic = wfs_MAGIC_NUMBER;
71     sb->s_op = &wfs_super_ops;
72
73     root = wfs_make_inode(sb, S_IFDIR | 0755);
74     if (!root)
75     {
76         printk (KERN_ERR "[wfs]_inode_allocation_failed!\n");
77         return -ENOMEM;
78     }
79
80     root->i_op = &simple_dir_inode_operations;
81     root->i_fop = &simple_dir_operations;
82
83     sb->s_root = d_make_root(root);

```

```

84     if (!sb->s_root)
85     {
86         printk(KERN_ERR "[wfs]_root_creation_failed!\n");
87         iput(root);
88         return -ENOMEM;
89     }
90
91     return 0;
92 }
93
94 static struct dentry* wfs_mount(
95     struct file_system_type *type, int flags, char const *dev, void *data
96 )
97 {
98     struct dentry* const entry = mount_nodev(type, flags, data, wfs_fill_sb);
99
100    if (IS_ERR(entry))
101        printk(KERN_ERR "[wfs]_mounting_failed!\n");
102    else
103        printk(KERN_DEBUG "[wfs]_mounted!\n");
104
105    return entry;
106 }
107
108 static struct file_system_type wfs_type = {
109     .owner = THIS_MODULE,
110     .name = "wfs",
111     .mount = wfs_mount,
112     .kill_sb = kill_litter_super,
113 };
114
115 static int __init wfs_init(void)
116 {
117     int i;
118     int ret;
119
120     if (size < 0)
121     {
122         printk(KERN_ERR "[wfs]_invalid_argument\n");
123         return -EINVAL;
124     }
125

```

```

126     line = kmalloc(sizeof(void*), GFP_KERNEL);
127
128     if (line == NULL)
129     {
130         printk(KERN_ERR "[wfs]_kmalloc_error\n");
131         kfree(line);
132         return -ENOMEM;
133     }
134
135     cache = kmem_cache_create(SLABNAME, size, 0, SLAB_HWCACHE_ALIGN, co);
136
137     if (cache == NULL)
138     {
139         printk(KERN_ERR "[wfs]_kmem_cache_create_error\n");
140         kmem_cache_destroy(cache);
141         kfree(line);
142         return -ENOMEM;
143     }
144
145     *line = kmem_cache_alloc(cache, GFP_KERNEL)
146     if (*line == NULL)
147     {
148         printk(KERN_ERR "[wfs]_kmem_cache_alloc_error\n");
149
150         kmem_cache_free(cache, *line);
151         kmem_cache_destroy(cache);
152
153         kfree(*line);
154
155         return -ENOMEM;
156     }
157
158     printk(KERN_INFO "[wfs]_allocate_%d_objects_into_slab:%s\n",
159           1, SLABNAME);
160     printk(KERN_INFO "[wfs]_object_size_%d_bytes, _full_size_%ld_bytes\n",
161           size, size);
162     printk(KERN_INFO "[wfs]_constructor_called_%d_times\n", sco);
163
164     ret = register_filesystem(&wfs_type);
165
166     if (ret != 0)
167     {

```

```

168         printk(KERN_ERR "[wfs]_module_cannot_register_filesystem!\n");
169         return ret;
170     }
171
172     printk(KERN_DEBUG "[wfs]_module_loaded!\n");
173     return 0;
174 }
175
176 static void __exit wfs_exit(void)
177 {
178     int i;
179     int ret;
180
181     for (i = 0; i < number; i++)
182         kmem_cache_free(cache, line[i]);
183
184     kmem_cache_destroy(cache);
185     kfree(line);
186
187     ret = unregister_filesystem(&wfs_type);
188
189     if (ret != 0)
190         printk(KERN_ERR "[wfs]_module_cannot_unregister_filesystem!\n");
191
192     printk(KERN_DEBUG "[wfs]_module_unloaded!\n");
193 }
194
195 module_init(wfs_init);
196 module_exit(wfs_exit);

```

```

[~] ~/Repositories/operating-systems/sem_02/lab_08/src  feature/lab_08_sem_02
[~] make
make -C /lib/modules/5.6.10-3-MANJARO/build M=/home/alex/Repositories/operating-systems/sem_02/lab_08/src mo
make[1]: вход в каталог «/usr/lib/modules/5.6.10-3-MANJARO/build»
CC [M] /home/alex/Repositories/operating-systems/sem_02/lab_08/src/wfs.o
/home/alex/Repositories/operating-systems/sem_02/lab_08/src/wfs.c: В функции «co»:
/home/alex/Repositories/operating-systems/sem_02/lab_08/src/wfs.c:26:16: предупреждение: приведение указателя
26 |      *(int*)p = (int)p;
    |      ^
MODPOST 1 modules
CC [M] /home/alex/Repositories/operating-systems/sem_02/lab_08/src/wfs.mod.o
LD [M] /home/alex/Repositories/operating-systems/sem_02/lab_08/src/wfs.ko
make[1]: выход из каталога «/usr/lib/modules/5.6.10-3-MANJARO/build»

[~] ~/Repositories/operating-systems/sem_02/lab_08/src  feature/lab_08_sem_02
[~] sudo insmod wfs.ko

[~] ~/Repositories/operating-systems/sem_02/lab_08/src  feature/lab_08_sem_02
[~] sudo lsmod | grep wfs
wfs                16384  0

[~] ~/Repositories/operating-systems/sem_02/lab_08/src  feature/lab_08_sem_02
[~] sudo dmesg | grep "[wfs\]"
[17308.793430] [wfs] allocate 31 objects into slab: wfs_cache
[17308.793432] [wfs] object size 7 bytes, full size 217 bytes
[17308.793433] [wfs] constructor called 256 times
[17308.793437] [wfs] module loaded!

[~] ~/Repositories/operating-systems/sem_02/lab_08/src  feature/lab_08_sem_02
[~] sudo cat /proc/slabinfo | grep wfs
wfs_cache          256    256    16  256    1 : tunables    0    0    0 : slabdata    1    1    0

```

Рис. 1: Загрузка модуля

```

[~] ~/Repositories/operating-systems/sem_02/lab_08/src/test
[~] touch image

[~] ~/Repositories/operating-systems/sem_02/lab_08/src/test
[~] md dir

[~] ~/Repositories/operating-systems/sem_02/lab_08/src/test
[~] sudo mount -o loop -t wfs ./image ./dir

[~] ~/Repositories/operating-systems/sem_02/lab_08/src/test
[~] sudo dmesg | grep "[wfs\]"
[17308.793430] [wfs] allocate 31 objects into slab: wfs_cache
[17308.793432] [wfs] object size 7 bytes, full size 217 bytes
[17308.793433] [wfs] constructor called 256 times
[17308.793437] [wfs] module loaded!
[17434.946328] [wfs] mounted!

[~] ~/Repositories/operating-systems/sem_02/lab_08/src/test
[~] sudo umount ./dir

[~] ~/Repositories/operating-systems/sem_02/lab_08/src/test
[~] sudo dmesg | grep "[wfs\]"
[17308.793430] [wfs] allocate 31 objects into slab: wfs_cache
[17308.793432] [wfs] object size 7 bytes, full size 217 bytes
[17308.793433] [wfs] constructor called 256 times
[17308.793437] [wfs] module loaded!
[17434.946328] [wfs] mounted!
[17455.783357] [wfs] super block destroyed!

```

Рис. 2: Монтирование файловой системы

```
~/Repositories/operating-systems/sem_02/lab_08/src/test  🐱  📁  release/lab_08_sem_02
└─ touch image

~/Repositories/operating-systems/sem_02/lab_08/src/test  🐱  📁  release/lab_08_sem_02
└─ md dir

~/Repositories/operating-systems/sem_02/lab_08/src/test  🐱  📁  release/lab_08_sem_02
└─ sudo mount -o loop -t wfs ./image ./dir

~/Repositories/operating-systems/sem_02/lab_08/src/test  🐱  📁  release/lab_08_sem_02
└─ ll
Permissions Size User Group Date Modified Name      Permissions Size User Group Date Modified Name
drwxr-xr-x   - root root  12 мая  8:51  dir      .rw-r--r--   0 alex alex  12 мая  8:50  image
```

Рис. 3: Файловая система в дереве каталогов

```
~/Repositories/operating-systems/sem_02/lab_08/src  🐱  📁  release/lab_08_sem_02
└─ sudo rmmod wfs

~/Repositories/operating-systems/sem_02/lab_08/src  🐱  📁  release/lab_08_sem_02
└─ sudo dmesg | grep "\[wfs\]"
[17308.793430] [wfs] allocate 31 objects into slab: wfs_cache
[17308.793432] [wfs] object size 7 bytes, full size 217 bytes
[17308.793433] [wfs] constructor called 256 times
[17308.793437] [wfs] module loaded!
[17434.946328] [wfs] mounted!
[17455.783357] [wfs] super block destroyed!
[17491.496252] [wfs] module unloaded!
```

Рис. 4: Выгрузка модуля

```
~/Repositories/operating-systems/sem_02/lab_08/src  🐱  📁  release/lab_08_sem_02
└─ sudo insmod wfs.ko size=64 number=128

~/Repositories/operating-systems/sem_02/lab_08/src  🐱  📁  release/lab_08_sem_02
└─ sudo dmesg | grep "\[wfs\]" | tail -4
[17580.157306] [wfs] allocate 128 objects into slab: wfs_cache
[17580.157308] [wfs] object size 64 bytes, full size 8192 bytes
[17580.157308] [wfs] constructor called 128 times
[17580.157313] [wfs] module loaded!

~/Repositories/operating-systems/sem_02/lab_08/src  🐱  📁  release/lab_08_sem_02
└─ sudo cat /proc/slabinfo | grep wfs
wfs_cache      128      128      128      32      1 : tunables      0
```

Рис. 5: Загрузка модуля с заданием параметров