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	Операционые 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*) * number, 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     for (i = 0; i < number; i++)
136     {
137         line[i] = NULL;
138     }
139
140     cache = kmem_cache_create(SLABNAME, size, 0, SLAB_HWCACHE_ALIGN, co);
141
142     if (cache == NULL)
143     {
144         printk(KERN_ERR "[wfs] kmem_cache_create error\n" );
145         kmem_cache_destroy(cache);
146         kfree(line);
147         return -ENOMEM;
148     }
149
150     for (i = 0; i < number; i++)
151     {
152         if (NULL == (line[i] = kmem_cache_alloc(cache, GFP_KERNEL))) {
153             printk(KERN_ERR "[wfs] kmem_cache_alloc error\n");
154
155             for (i = 0; i < number; i++)
156             {
157                 kmem_cache_free(cache, line[i]);
158             }
159
160             kmem_cache_destroy(cache);
161             kfree(line);
162             return -ENOMEM;
163         }
164     }
165
166     printk(KERN_INFO "[wfs] allocate %d objects into slab: %s\n",
167         number, SLABNAME);
```

```
168     printk(KERN_INFO "[wfs] object size %d bytes, full size %ld bytes\n",
169         size, (long)size * number);
170     printk(KERN_INFO "[wfs] constructor called %d times\n", sco);
171 
172     ret = register_filesystem(&wfs_type);
173 
174     if (ret!= 0)
175     {
176         printk(KERN_ERR "[wfs] module cannot register filesystem!\n");
177         return ret;
178     }
179 
180     printk(KERN_DEBUG "[wfs] module loaded!\n");
181     return 0;
182 }
183 
184 static void __exit wfs_exit(void)
185 {
186     int i;
187     int ret;
188 
189     for (i = 0; i < number; i++)
190         kmem_cache_free(cache, line[i]);
191 
192     kmem_cache_destroy(cache);
193     kfree(line);
194 
195     ret = unregister_filesystem(&wfs_type);
196 
197     if (ret!= 0)
198         printk(KERN_ERR "[wfs] module cannot unregister filesystem!\n");
199 
200     printk(KERN_DEBUG "[wfs] module unloaded!\n");
201 }
202 
203 module_init(wfs_init);
204 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 указател
   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
» sudo rmmod wfs

~/Repositories/operating-systems/sem_02/lab_08/src
» 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. 3: Выгрузка модуля

```
~/Repositories/operating-systems/sem_02/lab_08/src
» sudo insmod wfs.ko size=64 number=128

~/Repositories/operating-systems/sem_02/lab_08/src
» 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
» sudo cat /proc/slabinfo| grep wfs
wfs_cache            128    128    128   32    1 : tunables    0
```

Рис. 4: Загрузка модуля с заданием параметров