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

По предмету: «Операционные системы»

Тема: Создание виртуальной файловой системы.

Преподаватель: Рязанова Н. Ю.
Студент: Мирзоян С.А.,
Группа: ИУ7-65Б

Москва, 2020 г.

Листинг

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/pagemap.h>
#include <linux/fs.h>
#include <asm/atomic.h>
#include <asm/uaccess.h>
 #include <linux/slab.h>

    #define VFS_MAGIC_NUMBER 0x13131313
    #define SLABNAME "vfs_cache"

    MODULE_LICENSE("GPL");
    MODULE_AUTHOR("Sergey Mirzoyan");
    MODULE_DESCRIPTION("VFS");


    static int size = 7;
    module_param(size, int, 0);
    static int number = 31;
    module_param(number, int, 0);

    static void* *line = NULL;

    static void co(void* p)
    {
        *(int*)p = (int)p;
    }
    struct kmem_cache *cache = NULL;

    static struct vfs_inode
    {
        int i_mode;
        unsigned long i_ino;
    } vfs_inode;

    // Создание inode
```

```
38.     static struct inode * vfs_make_inode(struct super_block *sb, int
    mode)
39.     {
40.         struct inode *ret = new_inode(sb);
41.         if (ret)
42.         {
43.             inode_init_owner(ret, NULL, mode);
44.             ret->i_size = PAGE_SIZE;
45.             ret->i_atime = ret->i_mtime = ret->i_ctime =
    current_time(ret);
46.             ret->i_private = &vfs_inode;
47.         }
48.         return ret;
49.     }
50.
51.     static void vfs_put_super(struct super_block *sb)
52.     {
53.         printk(KERN_DEBUG "VFS super block destroyed\n");
54.     }
55.
56.     // Операции структуры суперблок
57.     static struct super_operations const vfs_super_ops = {
58.         .put_super = vfs_put_super,
59.         .statfs = simple_statfs,
60.         .drop_inode = generic_delete_inode,
61.         };
62.
63.     // Функция инициализации суперблока
64.     // Создание корневого каталога ФС
65.     static int vfs_fill_sb(struct super_block *sb, void *data, int
    silent)
66.     {
67.         struct inode* root = NULL;
68.
69.         sb->s_blocksize = PAGE_SIZE;
70.         sb->s_blocksize_bits = PAGE_SHIFT;
71.         sb->s_magic = VFS_MAGIC_NUMBER;
72.         sb->s_op = &vfs_super_ops;
73.
74.         root = vfs_make_inode(sb, S_IFDIR | 0755);
```

```
75.         if (!root)
76.         {
77.             printk (KERN_ERR "VFS inode allocation failed\n");
78.             return -ENOMEM;
79.         }
80.
81.         root->i_op = &simple_dir_inode_operations;
82.         root->i_fop = &simple_dir_operations;
83.
84.         sb->s_root = d_make_root(root);
85.         if (!sb->s_root)
86.         {
87.             printk(KERN_ERR "VFS root creation failed\n");
88.             iput(root);
89.             return -ENOMEM;
90.         }
91.         return 0;
92.     }
93.
94.      // Монтирование ФС
95.     static struct dentry* vfs_mount(struct file_system_type *type, int
    flags, const char *dev, void *data)
96.     {
97.         struct dentry* const entry = mount_nodev(type, flags, data,
    vfs_fill_sb);
98.
99.         if (IS_ERR(entry))
100.        printk(KERN_ERR "VFS mounting failed\n");
101.        else
102.        printk(KERN_DEBUG "VFS mounted\n");
103.
104.        return entry;
105.    }
106.
107.    static struct file_system_type vfs_type = {
108.        .owner = THIS_MODULE, // Счетчик ссылок на модуль
109.        .name = "Sergey_vfs", // Название ФС
110.        .mount = vfs_mount, // Функция, вызываемая при монтировании ФС
111.        .kill sb = kill litter super, // Функция, вызываемая при
    размонтировании ФС
```

```
112.    };
113.
114.    // Инициализация модуля
115.    static int __init vfs_module_init(void)
116.    {
117.        int i;
118.        if(size < 0)
119.        {
120.            printk(KERN_ERR "VFS invalid sizeof objects\n");
121.            return -EINVAL;
122.        }
123.
124.        line = kmalloc(sizeof(void*) *number, GFP_KERNEL);
125.        if(!line)
126.        {
127.            printk(KERN_ERR "VFS kmalloc error\n");
128.            kfree(line);
129.            return -ENOMEM;
130.        }
131.
132.        for(i = 0; i < number; i++)
133.            line[i] = NULL;
134.
135.         // Создание кэша slab
136.         cache = kmem_cache_create(SLABNAME, size, 0,
    SLAB_HWCACHE_ALIGN, co);
137.
138.         if(!cache)
139.         {
140.             printk(KERN_ERR "VFS cannot create cache\n");
141.             // Уничтожение slab
142.             kmem_cache_destroy(cache);
143.             return -ENOMEM;
144.         }
145.
146.        for(i = 0; i < number; i++)
147.        {
148.            if(NULL == (line[i] = kmem_cache_alloc(cache,
    GFP KERNEL)))
149.            {
```

```
150.                 printk(KERN_ERR "VFS cannot alloc cache\n");
151.                 for(i = 0; i < number; i++ )
152.                     kmem_cache_free(cache, line[i]);
153.                 return -ENOMEM;
154.             }
155.         }
156.
157.         printk(KERN_INFO "VFS allocate %d objects into slab: %s\n",
    number, SLABNAME);
158.         printk(KERN_INFO "VFS object size %d bytes, full size %ld
    bytes\n", size, (long)size *number);
159.
160.         // Регистрация файловой системы
161.         int ret = register_filesystem(&vfs_type);
162.         if (ret != 0)
163.         {
164.             printk(KERN_ERR "VFS cannot register filesystem\n");
165.             return ret;
166.         }
167.
168.         printk(KERN_DEBUG "VFS loaded\n");
169.         return 0;
170.     }
171.
172.     static void __exit vfs_module_exit(void)
173.     {
174.         int i;
175.         for(i = 0; i < number; i++)
176.         {
177.             kmem_cache_free(cache, line[i]);
178.         }
179.
180.         kmem_cache_destroy(cache);
181.         kfree(line);
182.
183.         if (unregister_filesystem(&vfs_type) != 0)
184.         {
185.             printk(KERN_ERR "VFS cannot unregister filesystem!\n");
186.         }
187.
```

```
188.         printk(KERN_DEBUG "VFS unloaded!\n");
189.     }
190.
191.     module_init(vfs_module_init);
192.     module_exit(vfs_module_exit);
```

Результат работы программы

Загрузка модуля.

```
sergey@sergey-VirtualBox:~$ sudo insmod VFS.ko
[sudo] пароль для sergey:
sergey@sergey-VirtualBox:~$ lsmod | grep VFS
VFS                    16384  0
sergey@sergey-VirtualBox:~$
```

touch file – Создается образ диска
mkdir mydir – Создается корень файловой системы
sudo mount –o loop –t myfs ./file ./mydir – монтирование файловой системы
sudo umount ./mydir – размонтирование файловой системы

```
sergey@sergey-VirtualBox:~$ touch file
sergey@sergey-VirtualBox:~$ mkdir mydir
sergey@sergey-VirtualBox:~$ sudo mount -o loop -t myfs ./file ./mydir
sergey@sergey-VirtualBox:~$ sudo umount ./mydir
```

ls –l

```
drwxr-xr-x 2 sergey sergey    4096 мая 18 19:54  mydir
-rw-r--r-- 1 sergey sergey       0 мая 20 23:10  my_image
```

Сообщения о регистрации, монтировании и уничтожении ФС.

```
sergey@sergey-VirtualBox:~$ dmesg | tail -n3
[  102.056509] MYFS filesystem registered
[  221.041712] MYFS mounted
[  276.990639] MYFS super block destroyed
```

Состояние Slab (после монтирования)

```
sergey@sergey-VirtualBox:~$ sudo cat /proc/slabinfo | grep my_cache
my_cache           170    170     24  170    1 : tunables    0    0    0 : slabdata      1      1      0
```

Созданная файловая система : df - отчёт об использовании дискового пространства, -a - Включает в список файловых систем те, которые имеют размер в 0 блоков, -T – тип

Файл.система	Тип	1K-блоков	Использовано	Доступно	Использовано%	Смонтировано в
sysfs	sysfs	0	0	0	-	/sys
proc	proc	0	0	0	-	/proc
udev	devtmpfs	1987640	0	1987640	0%	/dev
devpts	devpts	0	0	0	-	/dev/pts
tmpfs	tmpfs	403024	1256	401768	1%	/run
/dev/sda1	ext4	20509264	9194848	10249560	48%	/
securityfs	securityfs	0	0	0	-	/sys/kernel/security
tmpfs	tmpfs	2015100	0	2015100	0%	/dev/shm
tmpfs	tmpfs	5120	4	5116	1%	/run/lock
tmpfs	tmpfs	2015100	0	2015100	0%	/sys/fs/cgroup
cgroup2	cgroup2	0	0	0	-	/sys/fs/cgroup/unified
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/systemd
pstore	pstore	0	0	0	-	/sys/fs/pstore
bpf	bpf	0	0	0	-	/sys/fs/bpf
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/cpuset
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/pids
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/devices
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/net_cls,net_prio
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/cpu,cpuacct
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/perf_event
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/rdma
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/blkio
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/hugetlb
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/memory
cgroup	cgroup	0	0	0	-	/sys/fs/cgroup/freezer
hugetlbfs	hugetlbfs	0	0	0	-	/dev/hugepages
systemd-1	-	-	-	-	-	/proc/sys/fs/binfmt_misc
debugfs	debugfs	0	0	0	-	/sys/kernel/debug
mqueue	mqueue	0	0	0	-	/dev/mqueue
configfs	configfs	0	0	0	-	/sys/kernel/config
fusectl	fusectl	0	0	0	-	/sys/fs/fuse/connections
/dev/loop0	squashfs	1024	1024	0	100%	/snap/gnome-logs/100
/dev/loop2	squashfs	248320	248320	0	100%	/snap/gnome-3-34-1804/27
/dev/loop3	squashfs	1024	1024	0	100%	/snap/gnome-logs/93
/dev/loop1	squashfs	261760	261760	0	100%	/snap/gnome-3-34-1804/33
/dev/loop4	squashfs	96256	96256	0	100%	/snap/core/9066
/dev/loop5	squashfs	384	384	0	100%	/snap/gnome-characters/539
/dev/loop6	squashfs	2560	2560	0	100%	/snap/gnome-calculator/748
/dev/loop7	squashfs	96128	96128	0	100%	/snap/core/8935
/dev/loop9	squashfs	63616	63616	0	100%	/snap/gtk-common-themes/1506
/dev/loop10	squashfs	56320	56320	0	100%	/snap/core18/1705
/dev/loop8	squashfs	164096	164096	0	100%	/snap/gnome-3-28-1804/116
/dev/loop11	squashfs	2560	2560	0	100%	/snap/gnome-calculator/730
/dev/loop13	squashfs	56192	56192	0	100%	/snap/gtk-common-themes/1502
/dev/loop12	squashfs	153600	153600	0	100%	/snap/gnome-3-28-1804/71
/dev/loop14	squashfs	15104	15104	0	100%	/snap/gnome-characters/495
/dev/loop15	squashfs	56320	56320	0	100%	/snap/core18/1754
в_убунту	vboxsf	248751100	201650776	47100324	82%	/media/sf__
tmpfs	tmpfs	403020	36	402984	1%	/run/user/1000
gvfsd-fuse	fuse.gvfsd-fuse	0	0	0	-	/run/user/1000/gvfs
/dev/fuse	fuse	0	0	0	-	/run/user/1000/doc
/dev/sr0	iso9660	83904	83904	0	100%	/media/sergey/VBox_GAs_6.0.4
binfmt_misc	binfmt_misc	0	0	0	-	/proc/sys/fs/binfmt_misc
/dev/loop16	myfs	0	0	0	-	/home/sergey/mydir