## LAB5 实验报告

## 一 小组成员及贡献排序

汪值 林元灿 陈景昊 吕若皙 木扎拜·塔依尔

## 二 实验要求:

- (1) 查看当前系统所使用的文件系统的类型及版本号;
- (2) 阅读 Ext3(或 Ext4)文件系统, 特别是索引节点相关的源代码。
- (3) 为内核添加一个新的系统调用 filesys, 其从调用者接收一个磁盘文件的全局路径名, 打印该文件占用的所有磁盘块;
- (4) 重新编译、安装内核;
- (5) 编写用户测试程序, 测试 filesys 系统调用

## 三 实验过程

(1) 查看当前系统所使用的文件系统的类型及版本号 文件系统类型

```
🔞 🗐 🗊 wangzhi@u16: ~
wangzhi@u16:~$ df -T -h
                                   Used Avail Use% Mounted on
Filesystem
                            Size
                 Type
udev
                 devtmpfs
                            725M
                                      0
                                         725M
                                                  0% /dev
                                          143M
tmpfs
                 tmpfs
                            150M
                                   7.0M
                                                  5% /run
                                                 12% /
/dev/sda1
                 ext4
                             38G
                                   4.0G
                                           32G
                                                 1% /dev/shm
1% /run/lock
0% /sys/fs/cgroup
tmpfs
                 tmpfs
                            746M
                                   316K
                                          746M
                            5.0M
                                          5.0M
tmpfs
                                   4.0K
                 tmpfs
tmpfs
                            746M
                 tmpfs
                                          746M
                                      0
                                                  1% /run/user/1000
tmpfs
                 tmpfs
                            150M
                                    52K
                                          150M
wangzhi@u16:~$
```

查看版本号

(2) 阅读 Ext3(或 Ext4)文件系统, 特别是索引节点相关的源代码 Ext4 文件系统类型源码:

/fs/ext4/ext4.h

/fs/ext4/inode.c

Ext4 中的 inode 大小为 256 字节

Ext4 超级块

```
struct ext4_super_block {
/*00*/ __le32 s_inodes_count;
                                          /* Inodes count */
    __le32 s_blocks_count_lo; /* Blocks count */
__le32 s_r_blocks_count_lo; /* Reserved blocks count */
__le32 s_free_blocks_count_lo; /* Free blocks count */
/*10*/ __le32 s_free_inodes_count; /* Free inodes count */
    __le32 s_first_data_block; /* First Data Block */
__le32 s_log_block_size; /* Block size */
__le32 s_log_cluster_size; /* Allocation cluster size */
块描述符 ext4 group desc 结构体
 * Structure of a blocks group descriptor
struct ext4_group_desc
      _le32 bg_block_bitmap_lo; /* Blocks bitmap block */
    __le32 bg_inode_bitmap_lo; /* Inodes bitmap block */
__le32 bg_inode_table_lo; /* Inodes table block */
__le16 bg_free_blocks_count_lo;/* Free blocks count */
     __le16 bg_free_inodes_count_lo;/* Free inodes count */
     __le16 bg_used_dirs_count_lo; /* Directories count */
__le16 bg_flags; /* EXT4_BG_flags (INODE_UNINIT, etc) */
      le32 bg exclude bitmap lo: /* Exclude bitmap for snapshots */
索引节点 ext4_inode
 681: struct ext4_inode {
           682:
 684:
Ext4_inode 结构体中 i_block 数组中存放的即为文件所在的磁盘块号
           __le32 i_block[EXT4_N_BLOCKS];/* Pointers to blocks */
__le32 i_generation; /* File version (for NFS) */
__le32 i_file_acl_lo; /* File ACL */
__le32 i_size_high;
__le32 i_obso_faddr; /* Obsoleted fragment address */
                                         /* Obsoleted fragment address */
EXT4_N_BLOCKS 的大小为 15
  362: /*
  363: * Constants relative to the data blocks
  365: #define EXT4_NDIR_BLOCKS
  366: #define EXT4_IND_BLOCK
                                                EXT4_NDIR_BLOCKS
  367: #define EXT4_DIND_BLOCK
                                                (EXT4_IND_BLOCK + 1)
  368: #define EXT4_TIND_BLOCK
                                                (EXT4_DIND_BLOCK + 1)
                                                (EXT4_TIND_BLOCK + 1)
  369: #define EXT4_N_BLOCKS
ext4_inode 中的 i_block[]转换到 vfs 的 inode 里的 i_data
4712: struct inode *ext4_iget(struct super_block *sb, unsigned long ino)
4713: {
```

```
4832:
           inode->i_generation = le32_to_cpu(raw_inode->i_generation);
4833:
           ei->i_block_group = iloc.block_group;
4834
           ei->i_last_alloc_group = ~0;
4835:
           * NOTE! The in-memory inode i_{data} array is in little-endian order
4836:
           * even on big-endian machines: we do NOT byteswap the block numbers!
4837:
4838:
4839:
          for (block = 0; block < EXT4_N_BLOCKS; block++)
    ei->i_data[block] = raw_inode->i_block[block];
4840:
4841:
           INIT_LIST_HEAD(&ei->i_orphan);
4842:
```

Block 大小为 512B

- (3) 为内核添加一个新的系统调用 filesys, 其从调用者接收一个磁盘文件的全局路径名, 打印该文件占用的所有磁盘块;
  - 1) 在系统调用表中添加系统调用
    - a) 打开 syscalls/syscall\_64.tbl 在末尾添加对应的系统调用号

```
      327
      64
      preadv2
      sys_preadv2

      328
      64
      pwritev2
      sys_pwritev2

      329
      64
      filesys
      sys_filesys
```

2) 在头文件中声明函数

}

- a) 打开 include/linux/syscalls.h, 在末尾添加函数声明
- b) asmlinkage int sys\_filesys (const char \_\_user \*str);

```
asmlinkage int sys_filesys (const char __user *str);
```

3) 添加系统调用函数 位置/kernel/sys.c

```
a) 实现函数 sys_filesys
asmlinkage int sys_filesys (const char __user *str){
         struct inode *inode=NULL;
         struct file *file=NULL;
         struct address space i data;
         struct radix tree root page tree;
         file=filp_open(str,O_DIRECTORY,0);
         if(IS_ERR(file)){
                   file=filp_open(str,O_RDONLY,0444);
                   if(IS ERR(file)){
                        printk("The path %s is error!\n",str);
                        return 0;
                   }
                   inode = file_inode(file);
                   //inode=file->f dentry->d inode;
                   printk("file name is: %s\n", str);
printk("inode number : %ld\n",inode->i_ino);
printk("file size(byte) : %d\n",(int)inode->i_size);
                   printk("block count : %lu\n",inode->i_blocks);
                   return 0;
         }
```

- 4) 重新编译安装内核
- 5) 添加测试文件, 运行测试文件

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

#include <sys/syscall.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(){

    const char filename[] = "test.zip";
    syscall(329, filename);

    return 0;
}
```

6) 查看结果

```
wangzhi@u16:~/module$ sudo dmesg -c
[ 5643.329080] file name is: test.zip
[ 5643.329083] inode number : 797208
[ 5643.329084] file size(byte) : 5258
[ 5643.329085] block count : 16
wangzhi@u16:~/module$
```