Дубликат

Ф. Платонова

И. Ольга

О. Сергеевна

Студент ИУ7-35Б
(группа)

2018 2022 уч. г.

Начальник отдела охраны

А. Максименко

Выдан «06» 12 2019 г.

1. Файловая система

Файловая система – часть операц. системы, которая отвечает за возможность длительного хранения данных и доступа к ним (создания, чтения, записи и т.д.)

Файловая система определяет
- формат сохран. данных
- способ физ. хранения данных
- связь физ. носителя данных и сис. вызовов для доступа к файлам

Файл – поименованная совокупность данных, хранящ. во вторичной памяти (энерго-независимые блочные устрой-ва).

Система Linux поддерживает 7 типов файлов:

1. регулярный
2. директория
3. именованный канал
4. сокет
5. символич. ссылка
6. спец. файл символьного устрой-ва
7. спец. файл блочного устрой-ва

Система не интерпретирует данные, только обеспечивает их хранение и доступ к ним.

2. Задачи ФС

Основная задача ФС – организация доступа к данным, хранящ. на диске.

1) именование файлов
2) обеспеч. программного интерфейса для работы с файлами приложен./пользоват.
3) отображение логич. представл. на физ. организ. хранения данных
4) обеспеч. длительного хранения файлов и доступа к ним
5) обеспеч. совместного доступа к файлам.

3. Иерархическая организация ФС

5 уровней:

1. Символьный уровень
   уровень именования файлов
   организация каталогов/подкаталогов

   Имя файла не явл. идентификатором для доступа к нему из разных директорий

2. Базовый уровень
уровень формирования дескриптора файла
В ФС файл идентифиц. номером inode
2 типа inode: 1) ядерный
содержит инф-цию для нахождения файла
2) дисковый
содержит инф-цию для адресации данных в файле
Для доступа: переход от симв. имени к номеру inode (метаданные).

3. Уровень проверки прав доступа
у каждого файла сущ. права доступа
устанавливаются для пользователя
гр. пользователей
остальных

4. Логический уровень
позволяет ~~создать~~ обеспечить доступ к данным в файле в формате, отличном от формата их факт. хранения
запрос на чтение данных —преобраз.→ запрос на физ. послед-ть байтов
Поддерживается 2 типа файлов:
- байториентированные (оперир. байтами)
- блокориентированные (оперир. блоками)

5. Физический уровень
уровень хранения данных и доступа к ним

3. Файловая подсистема Linux
Поддержка большого числа ФС
Наличие VFS (Linux) и VFS/vnode (Unix) позволяет ФС Unix/Linux поддерживать большое число разных ФС.

Структура, описывающая ФС

Основная структура, описывающая ФС в ядре - struct file_system_type. Определяет тип ФС. Объявлена в linux/fs.h. Для каждой ФС только одна структура file_system_type.

```
struct file_system_type
{
    const char *name;
    int fs_flags;
    #define FS_REQUIRES_DEV 1
    #define FS_BINARY_MOUNTDATA 2
    ...
    struct dentry *(*mount) (struct file_system_type *,
                             int, const char *, void *);
    void (*kill_sb) (struct super_block *);
    struct module *owner;
    struct file_system_type *next;
    struct hlist_head fs_supers;
    ...
}
```

name - название ФС
fs_flags - флаги (напр., FS_REQUIRES_DEV - для ФС, кот. могут быть смонтированы только с блоч. устрой-ва)
next - указатель на след. элемент в списке file_systems.

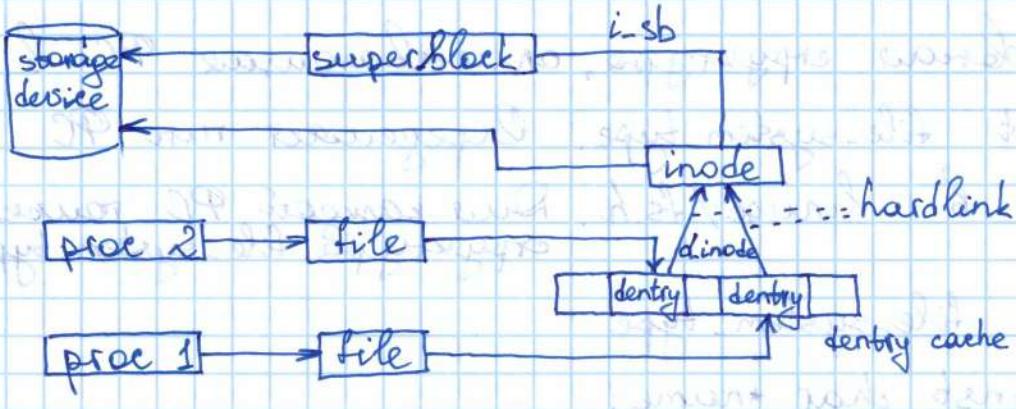
VFS: 4 основные структуры и связь между ними

Виртуальная файловая система - набор структур

Базовые структуры:

1) Супер-блок
   struct super_block
   позволяет создавать пользоват. тип ФС
2) Индексный узел
   struct inode
3) Элемент каталога
   struct dentry
4) Файл struct file

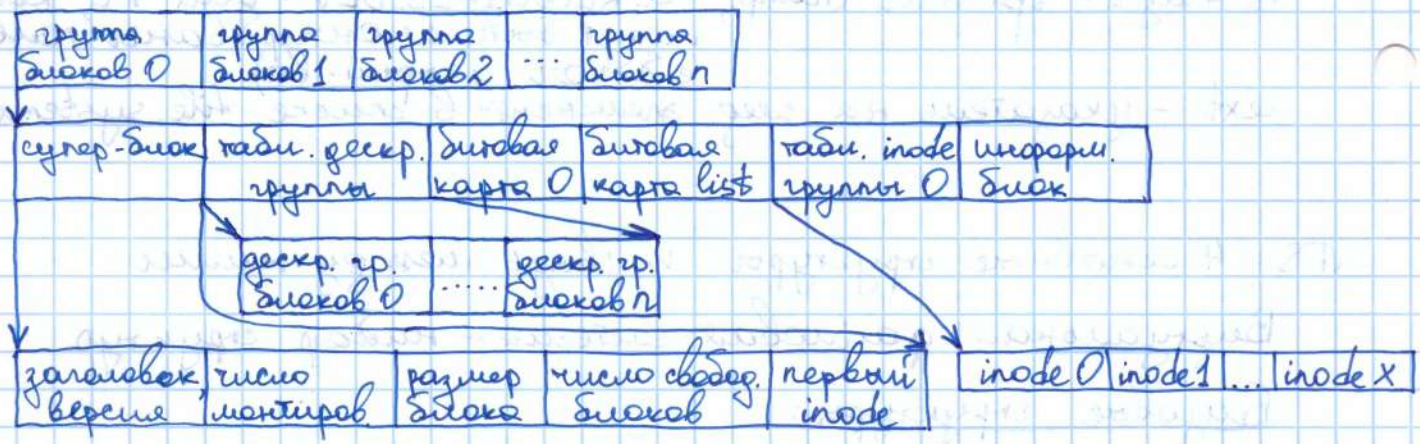
Структуры связаны между собой



1) Процесс пытается открыть файл.
Для каждого процесса создается ф.д. в таблице открытых файлов. В системе ∃1 таблица всех ф.д. открытых файлов
2) Доступ к файлу по его полному имени – для этого struct dentry
3) Идентиф. файла – номер inode
дескрипт – struct inode
4) Супер-блок содержит общую инф-цию об ФС, т.к. любой файл принадлежит опред. ФС

Раздел жесткого диска и суперблок

Суперблок содержит инф-цию о смонтированной ФС.



атрибуты файла
адрес блока 0 прямой
...
адрес блока 11 прямой
IB (косвенная ссылка)
DIB (двойная к. с.)
TIB (тройная к. с.)

Смонтиров. ФС содержит 1 суперблок

Инф-ция о ФС: инф-ция о файлах данной ФС

Для обращения к файлам необход:
- битовая карта i$list
- таблица inode-ов.

12 блоков прямой адр
косвенная адр
двойная -//-
тройная -//-

Структура struct seyper_operations

Описывает группу операций, определенные на суперблоке.

Определена в <linux/fs.h>

```
struct super_operations
{
    struct inode *(*allec_inode)(struct super_block *sb);
    void (*destroy_inode)(struct inode *);
    void (*dirty_inode)(struct inode *, int flags);
    int (*write_inode)(struct inode *, struct write_back_control
                                                    *wbc);
    int (*drop_inode)(struct inode *);
    void (*put_super)(struct super_block *);
    int (*sync_fs)(struct syne_block *sb, int wait);
    ...
    int (*stat_fs)(struct dentry, struct kstat_fs *);
    int (*remount_fs)(struct super_block, int *, char *);
    void (*umount_begin)(struct superblock);
}
```

alloc_inode - создает и инициализ. новый объект inode
dirty_inode - обновляет инф-цию в журналируемых ФС
write_inode - записывает указанный inode на диск
drop_inode - вызывается подсистемой ВФС когда исчезает последняя ссылка на inode
put_super - при размонтир. для освобожд. суперблока
sync_fs - синхрон. метаданные с данными на диске
stat_fs - возвращ. инф-цию о загрущ. ФС.
remount_fs - вызывается когда ФС монтир. с др. параметрами
umount_begin - прерывание монтирования

Монтирование ФС

Основная структура данных, описывающая ФС в ядре – struct file_system_type.

Прежде всего – инициализация этой структуры:

а)
```
static struct file_system_type myfs_type =
{
        .owner = THIS_MODULE,
        .name = "myfs",
        .mount = myfs_mount,
        .kill_sb = kill_block_super,
};
```

.owner – отслеживает счётчик ссылок на модуль, чтобы не выгрузить случайно
.mount – функция, вызываемая при монтировании системы
.kill_sb – при размонтировании

б)
```
static struct dentry *myfs_mount(struct file_system_type *type,
                                 int flags, char const *dev,
                                 void *data)
{
        struct dentry * const entry = mount_bdev(type, flags, dev, data,
                                                 myfs_fill_sb);
        ...
        return entry;
}
```

ф-ция монтирует устрой-во и возвращ. структуру dentry, описывающую корневой каталог ФС

mount_bdev выполняет основную работу
принимает в кач-ве аргумента myfs_fill_sb.

в)
```
static int myfs_fill_sb(struct super_block *sb, void *data,
                        int silent)
{
        sb->s_blocksize      = PAGE_SIZE;
        sb->s_blocksize_bits = PAGE_SHIFT;
        sb->s_magic          = MAGIC_NUMBER;
        sb->s_op             = &myfs_super_ops;

        root = myfs_make_inode(sb, S_IFDIR | 0755);
        ...

        struct inode *root = NULL;
        ...
        sb->s_root = d_make_root(root);
        ...

        return 0;
}
```

Заполняется структура super_block
Создается inode корневого каталога нашей ФС.

2)
```
static struct inode * myfs_make_inode (struct super_block *sb,
                                       int mode)
{
    struct inode *ret = new_inode(sb);
    if (ret) {
        inode_init_owner(ret, NULL, mode);
        ret->i_private = myfs_inode;
    }
    return ret;
}
```

myfs_inode - структура для собственного inode.

Ф-ция создает inode, аргумент mode задает разрешения на создаваемый файл

Команда mount

Команда монтирования ФС

mount -o loop -t myfs ./image ./dir

-o список параметров
loop указание на устрой-во (вместо реального диска)
-t myfs тип ФС : myfs

./image - устрой-во
./dir - каталог

Когда ФС монтируется, создается структура vfs_mount, которая представляет конкретный экземпляр ФС

```
struct vfs_mount
{
    struct dentry *mount_root;     //указатель на корневой каталог
    struct superblock *mnt_sb;     //указатель на суперблок
    int mnt_flags;                 //флаги монтирования
};
```