

# Файловая подсистема Linux

Ф.с. в Linux описывается суперблок (описывает подмонтированную ф.с.);

У каждой ф.с. и.б. один суперблок;

суперблок содержит информацию для доступа к файлам;

Файлы описываются inode (index node);

inode — дескриптор файла;

номер inode — индекс inode, по которому системой обеспечивается доступ;

Обычные файлы ~ регулярные;

inode должен содержать информацию об адресах блоков диска, в которых хранится информация, записываемая в файл;

⇒ суперблок должен содержать соотв. информацию о блоках на диске (свободен/занят), об inode'ах, иметь ссылку на соотв. таблицы inode'ов;

Раздел „Файловая система в стивенсе“

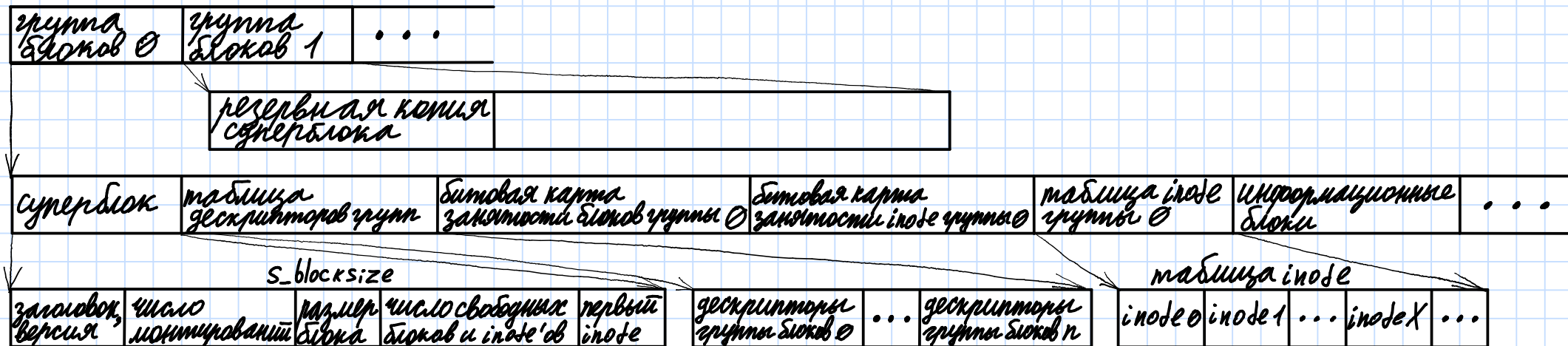
Каждый раздел можно приписать распечатанные структуры;

Если пользователь желает, чтобы ф.с. стала доступна (пользователь сможет получить доступ к файлам и каталогам данной ф.с.), она должна быть монтирована и для неё должен быть выделен раздел жесткого диска;

Раздел на диске - выделенное адресное пр-во;  
1-я структура в разделе - суперблок;

Обычно все приводимые картинки относятся к ф.с. `ext2` - "родной" для Linux

# Раздел жесткого диска в ext 2



struct dentry  
\*s\_root

root	
bin	814
dev	419
etc	778

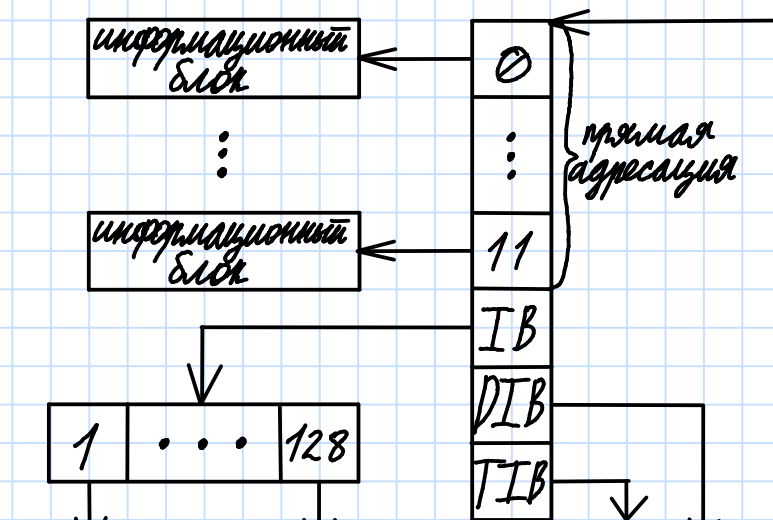


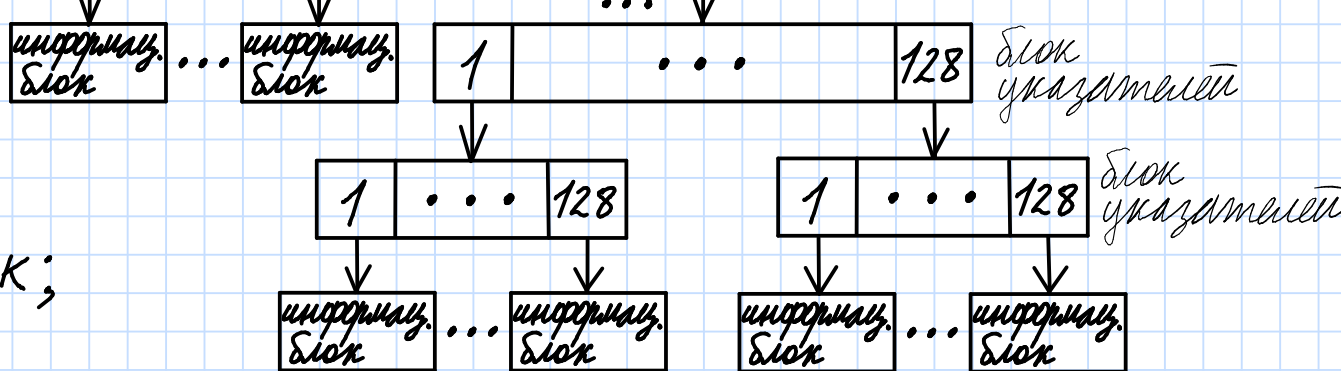
таблица inode  
inode 0

атрибуты файла:
User ID
Group ID
File size
Access permission
Timestamps: ctime - в. модиф. inode
mtime - в. модиф. файла
atime - в. посл. доступа к файлу
File location

IB - Indirect Block;  
(косв. адресация)

DIB - Double  
Indirect Block;

TIB - Triple  
Indirect Block;



Когда ф.с. монтируется, заполняются поля структуры `superblock`;

Размер блока (на диске) - непр. адр. пр-во;

В совр. выч. системах файлы отн. большие, поэтому они не могут храниться в непр. адр. пр-ве, они хранятся вразброс в свободных блоках;

Система должна иметь возможность обращаться к каждому блоку, в котором хранится информация. Для этого в `inode` имеются соотв. ссылки;

Чтобы ф.с. имела возможность хранить информацию о файлах (файлы имеют тенденцию увеличиваться в размере)

"Файл размера не имеет";  
(в конце файла стоит признак конца файла)

В этом состоит отличие файла от массива, размер которого задается 1 раз при создании

Чтобы система могла гибко выделять файлам новые блоки, необходимо иметь соотв. информацию;

`s_blocksize` - <sup>(размер блока)</sup> имя поля в `struct superblock`;

Первый `inode` - `inode` корневого каталога ф.с.;

В группе блоков 1 находится резервная копия суперблока;



Информация о каталогах так же хранится на диске (ее нужно хранить долго)

Иерархическая структура каталогов очень удобна для доступа к файлам;

Информация о каталогах должна храниться длительное время, т.к. используется для доступа к файлам

⇒ На диске хранятся файлы-каталоги и собственно файлы;

Доступ к любому эл-ту каталога осуществляется по его индексу (номеру inode);

Доступ к информации ф.с. осуществляется через superblock;

Если файл просто лежит на диске, то через дерево каталогов можно увидеть это;

Увидеть можно только подмонтированную ф.с.;

✶ есть открытые файлы - файлы, с которыми работают процессы.

Каждый процесс может открыть файл.

superblock содержит информацию, необходимую для монтирования и управления ф.с.;

В каждой ф.с. один superblock, и он располагается в начале раздела (у него есть копия для обеспечения надёжности работы с ф.с., тране-

ния файлов;  
superblock считывается в память ядра  
при монтировании ф.с. и находится  
там до её демонтирования (или завер-  
шения работы с системой);

Пример: мы хотим посмотреть содержи-  
мое флешки. Флешка имеет свою ф.с., она м.б.  
подмонтирована к дереву каталогов, и её дирек-  
тории, поддиректории и файлы, которые мы сохра-  
нили на флешке, будут доступны. Потом  
мы достаем флешку. "Хорошая" система контро-  
лирует это и сделает демонтаж ф.с. за нас;

timestamps - временные отметки, указы-  
вающие время модификации inode:

- ctime - время модификации inode;
- mtime - время модификации файла;
- atime - время последнего доступа к файлу;

В дисковом inode хранится информация  
о физ. расположении файла на диске;

Блок м.б. адресован ~ мы знаем его адрес;

Битовая карта блоков (block bit map) - струк-  
тура, в которой каждый бит показывает,  
занят блок или нет (отведён ли он какому-то  
файлу);

Битовая карта индексных дескрипторов (inode)  
Выполняет аналогичную роль по отн-ю к табли-

це inode'ов (индексных дескрипторов);  
П. е. показывает, какие индексные дескрипторы заняты, а какие свободны;  
Информация по superblock есть в мануале;  
Тю адресацию и время доступа:  
Прямая адресация - быстрый доступ к блоку;  
Время доступа - плата за возможность адресации очень больших файлов;

---

Действ:

Оптимальный алгоритм обращения к блоку данных на диске

2 типа движения:

- возвратно-последовательное;
- случайное;

На основе их комбинации выстраивается очередь, чтобы сократить время доступа к орг- блоку;

Сейчас это не обосновывается, потому что, видимо, был найден оптимальный алгоритм, лучше которого пока не придумали;

---

```
struct super_block
{
```

```
    struct list_head
```

```
    kdev_t
```

```
    unsigned long
```

```
    unsigned char
```

```
    struct file_system_type
```

```
    struct super_operations
```

```
    struct block_device
```

```
    ...
```

```
    unsigned long
```

```
    struct dentry
```

```
    ...
```

```
    int
```

```
    struct list_head
```

```
    char
```

```
}
```

```
    s_list;
```

```
    s_dev;
```

```
    s_blocksize; // размер блока в байтах
```

```
    s_dirt; // флаг изменения суперблока
```

```
    *s_type; // в ядре есть структура, описывающая тип ф.с.
```

```
    *s_op; // операции на суперблоке
```

```
    *b_dev; // описывает устр-во, на котором находится ф.с.
```

```
    // (соответствует драйверу блочного устр-ва)
```

```
    s_magic; // магический номер (идентификатор) ф.с.
```

```
    *s_root; // точка монтирования (каталога) ф.с.
```

```
    s_count; // число ссылок
```

```
    s_dirty; // список измененных inode'ов
```

```
    s_id[32]; // UUID
```

```
    ?
```

На любой структуре, описывающей объект ядра, определены ф-ции для работы с объектом соотв. нима (struct file\_operations, struct inode\_operations, struct dentry\_operations);

„В ядре всё определено, всё перечислено“;

Flash back: разработка UNIX началась с ф.с. Без ф.с. невозможно создание приложений, работающих в режиме пользователя (сложно разделить user mode и kernel mode)



<linux/fs.h>

```
struct super_operations
```

```
{  
    struct inode *(alloc_inode)(struct superblock *sb);  
    void (*destroy_inode)(struct inode *);
```

```
    ...  
    void (*dirty_inode)(struct inode *, int flags);
```

```
    int (*write_inode)(struct inode *, struct writeback_control *wbc);
```

```
    int (*drop_inode)(struct inode *);
```

```
    void (*put_super)(struct superblock *);  
}
```

Осн. информация, которую хранит superblock — информация, которая обеспечивает доступ к таблице inode'ов, и каждый дескриптор inode обеспечивает информацию для доступа к данным, хранящимся в файле  $\Rightarrow$  В struct super\_operations есть ф-ция alloc\_inode (принимает superblock, т.к. любой файл должен принадлежать конкретной ф.с., т.е. конкретной суперблоку; конкретный superblock обеспечивает доступ к конкретному файлу);

dirty\_inode вызывается VFS, когда в индексе <sup>(inode)</sup> вносятся изменения (ф-ция используется для изменения соотв. таблицы структуры);

Ядро хранит копию таблицы inode'ов в памяти ядра (т.к. доступ к диску — медленная операция), т.е. inode, к которым были обращения, кешируются

для ускорения доступа к файлам;  
Сначала изменения вносятся в таблицу, кото-  
рая находится в оперативной памяти;  
Ф-ция `dirty_inode` позволяет отметить, что `inode`  
был изменён, и эту информацию надо сохранить  
в таблицу, которая находится на диске;

`write_inode` предназначена для записи `inode`  
на диск и помечает `inode` как изменённый;

`put_super` вызывается VFS при размонтировании ф.с.;

```
static struct super_block *alloc_super(  
    struct file_system_type *type,  
    int flags,  
    struct user_namespace *user_ns  
)
```

Ф-ция `alloc_super` создаёт новый суперблок;  
Вызывается при монтировании ф.с.;

Возвращает указатель на новый суперблок или  
`NULL`, если не удалось выделить суперблок;

Пример обращения к ф-ции:

`sb → s_op → write_super(sb);`

Запись похотся на ООП, но ядро написано на Си  
(без ООП). Язык Си настолько мощный, что  
позволяет исп-ть такие записи, в которых  
одна структура ссылается на другие структуры

## struct inode

На диске (устойчивое долговременное хранение) должны храниться файлы, содержащие информацию о регулярных (обычных) файлах и файлах-директориях, чтобы эта информация могла быть представлена в виде дерева каталогов;

⇒ inode содержат информацию о файлах-директориях, так и об обычных файлах (об их расположении на диске);

у struct inode есть номер (inode number) - индекс (смещение к inode в таблице inode'ов);

ls -li - увидеть inode;

df - увидеть список файловых систем; при этом в списке мы увидим, сколько inode'ов содержит ф.с., сколько inode'ов используется, сколько свободных inode'ов, % исп-я inode'ов и точку монтирования;

Пример df -hi:

Filesystem	Inodes	I used	I free	I use%	Mounted on
------------	--------	--------	--------	--------	------------

Информацию из inode файла также можно получить в user mode, используя команду stat:

stat user.txt:

File: user.txt regular file  
Size: 78 Blocks: 8 IO Block: 4096  
Device:  
Access: