Файловая подсистема Linux

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

У каждой ф.с. м.б. один superblock;

superblock 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;

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

Раздел „Файловая система в Стивенсе"

На экзамен можно принести распечатанные структуры;

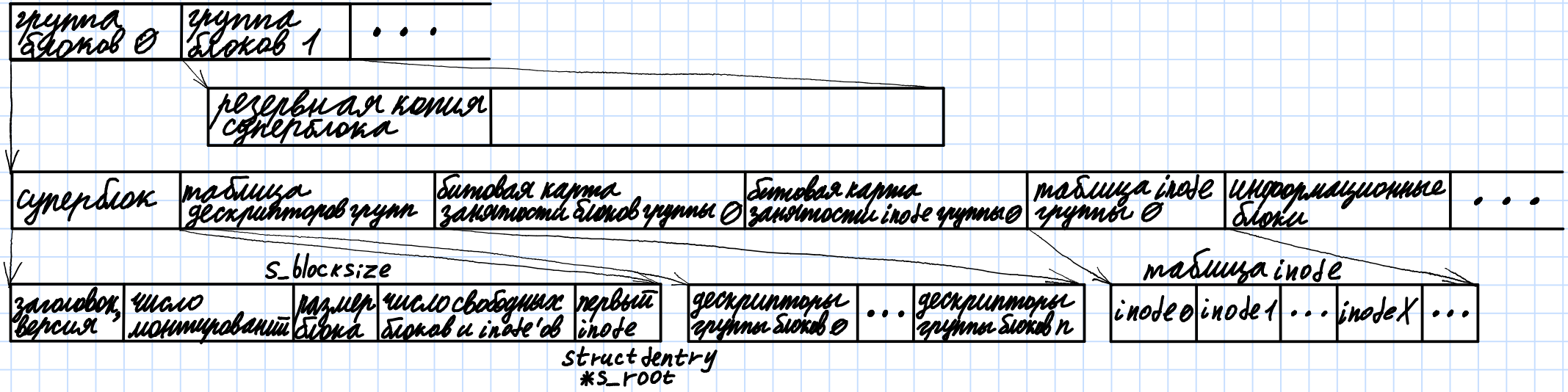
Если пользователь желает, чтобы ф.с. стала доступна (пользователь сможет получить доступ к файлам и каталогам данной ф.с.), она должна быть подмонтирована и для неё должен быть выделен раздел жёсткого диска;

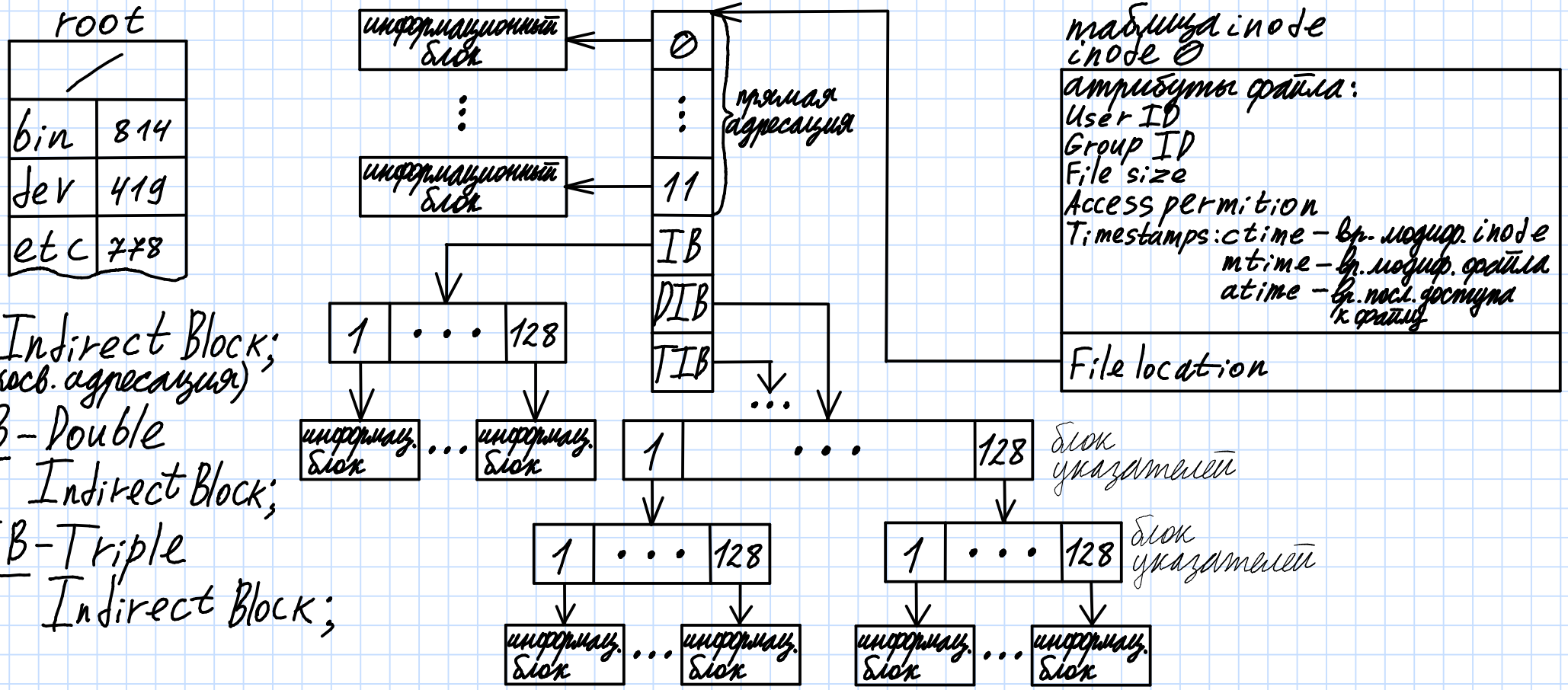
Раздел на диске – выделенное адресное пр-во;

1-я структура в разделе – super-block;

Обычно все приводимые картинки относятся к ф.с. ext2 – „родной" для Linux

Раздел жёсткого диска ext2





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 сбр. выч. системах файлы отн. большие, поэтому они не могут храниться в непр. адр. пр-ве, они хранятся вразброс в свобод-ных блоках;

Система должна иметь возможность обра-щаться к каждому блоку, в котором храни-ся информация. Для этого в inode имеются соотв. ссылки;

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

„Файл размера не имеет";

"(в конце файла стоит признак конца файла)

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

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

s_blocksize - (размер блока) имя поля в struct superblock;

Первый inode - inode корневого каталога ф. с.;

В группе блоков 1 находится резервная копия суперблока;

Информация о каталогах также 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.;

А есть открытые файлы - файлы, с которыми работают процессы.
Только 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., хране-

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

Пример: мы хотим посмотреть содержимое флешки. Флешка имеет свою ф.с., она м.б. подмонтирована к дереву каталогов, и её директории, поддиректории и файлы, которые мы сохранили на флешке, будут доступны. Потом мы достаём флешку. „Хорошая" система контролирует это и сделает демонтаж ф.с. за нас;

timestamps – временные отметки, указывающие 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) – структура, в которой каждый бит показывает, занят (1) блок или нет (0) (отведён ли он какому-то файлу);

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

це inode'ов (индексных дескрипторов);

Т. е. показывает, какие индексные дескрипторы заняты, а какие свободны;

Информация по superblock есть в мануале;

Про адресацию и <u>время доступа</u>:
Прямая адресация - быстрый доступ к блоку;

Время доступа - плата за возможность адресации очень больших файлов;

Деител:
Оптимальный алгоритм обращения к блоку данных на диске
2 типа движения:
- возвратно-поступательное;
- вращательное;
На основе их комбинации выстраивается очередь, чтобы сократить время доступа к физ. блоку;

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

```
struct super_block
{
    struct list_head        s_list;     // device (устройство, на котором находится ф.с.)
    kdev_t                  s_dev;
    unsigned long           s_blocksize; // размер блока в байтах
    unsigned char           s_dirt;     // флаг изменения суперблока
    struct file_system_type *s_type;    // в ядре есть структура, описывающая этот ф.с.
    struct super_operations *s_op;      // операции на суперблоке
    struct block_device     *b_dev;     // описывает устр-во, на котором находится ф.с. (соответствует драйверу блочного устр-ва)
    ...
    unsigned long           s_magic;    // магическое число (смонтированной) ф.с.
    struct dentry           *s_root;    // точка монтирования (каталога) ф.с.
    ...
    int                     s_count;    // число ссылок
    struct list_head        s_dirty;    // список изменённых inode'ов
    char                    s_id[32];   // имя ?
}
```

На любой структуре, описывающей объекты ядра, определены ф-ции для работы с объектом соотв. типа (struct file_operations, struct inode_operations, struct dentry_operations);

„В ядре всё определено, всё перечислено";

---

flashback: 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 super_block *);
}
```

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 ⟹ в struct super_operations есть ф-ция alloc_inode (принимает superblock, т.к. любой файл должен принадлежать конкретной ф.с., т.е. конкретному суперблоку; конкретный superblock обеспечивает доступ к конкретному файлу);

dirty_inode вызывается VFS, когда в (inode) индекс 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, которая находится в оперативной памяти;
Ф-ция dirty_inode позволяет отметить, что inode был изменён, и эту информацию надо скопировать 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 superblock;
Вызывается при монтировании ф.с.;
Возвращает указатель на новый superblock или NULL, если не удалось выделить superblock;

Пример обращения к фо-ции:
```
sb -> s_op -> write_super(sb);
```

Запись похожа на ООП, но ядро написано на Си (без ООП). Язык Си настолько мощный, что позволяет исп-ть такие записи, в которых поля структур ссылаются на другие структуры

struct inode

На диске (устр-ве долговременного хранения) должны храниться файлы, содержащие информацию о регулярных (обычных) файлах и файлах-директориях, чтобы эта информация могла быть представлена виде дерева каталогов;
⟹ inode содержат информацию о файлах-директориях, так и об обычных файлах (обих расположении на диске);

у struct inode есть номер (inode number) - индекс (смещение к inode в таблице inode'ов);
ls -i — увидеть inode;
 index

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	IUsed	IFree	IUse%	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: