

Рязанова, лекция.

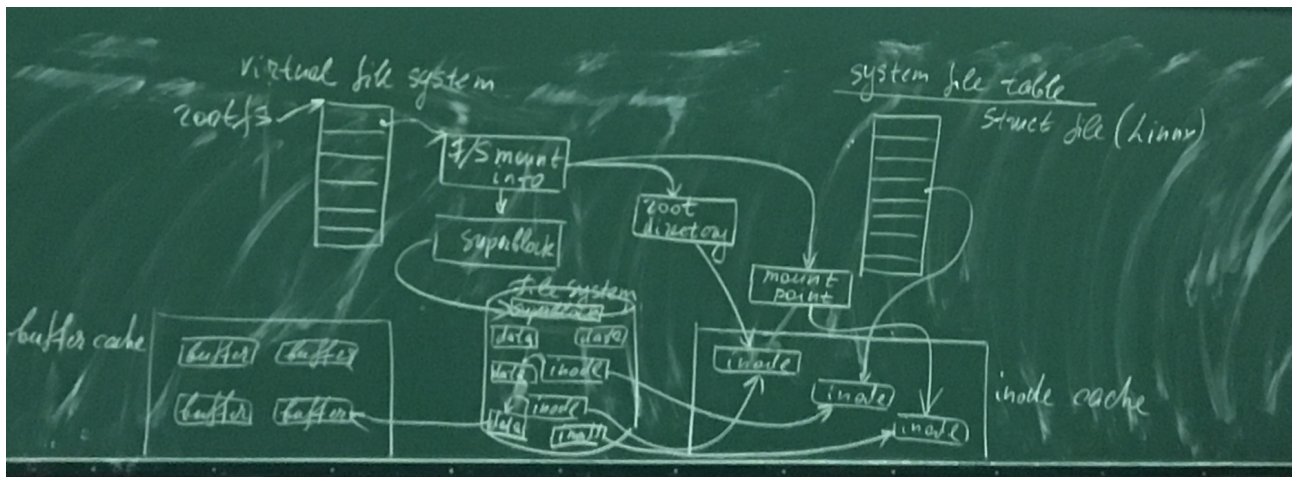
Продолжение. Файловая система UNIX/LINUX.

Поле `struct super_operations` - структура, которая определяет действия, которые мы можем выполнять над суперблоками.

Имеется также ссылка на `file_system_type`, она определяет тип файловой системы, в файловой системе может быть только один тип конкретной файловой системы, в VFS может существовать только одна структура `file_system_type`, но систем, которые могут иметь такой тип может быть сколько угодно.

Очевидно, что если не предпринять определенных действий в системе, а каждый раз обращаться к диску, то время получения инфы в файлах и о файлах будет значительным, потому что диск это внешнее устройство, медленно-действующее с точки зрения быстродействия системы.

Информация о файлах кэшируется, иначе доступ к файлам будет слишком долгий.



Все в системе есть файл, следовательно у всего есть inode.

`Struct file` описывает открытые файлы. Файл существует в двух эпостасях - файл, который лежит на диске, и мы можем посмотреть, что он есть.

30 марта, сб

В системе существует одна таблица всех открытых файлов - `struct file table`. Открыть файл значит обратиться к файлу на диске, эта структура позволяет организовать работу с открытым файлом. **А сам файл описывается `inode`'ом. `Inode` обеспечивает доступ к данным, которые находятся на физическом носителе.**

Чтобы обеспечить производительность обращения к файлам, для того чтобы процесс обращения к файлам не занимал слишком много времени, вся информация кэшируется. В значительно упрощенном виде происходит все кэширование.

Файловые структуры должны быть в ядре резидентно (постоянно). В `inode` кэше сохраняем соответствующие `inode`, данные и тд. Кроме того, эти данные сохраняются в различных таблицах.

`File_system_type`:

(Роберт Лав - про разработку ядра, в сети есть инфа из этой книжки 3е издание 2012, структуры, которые приводятся в этой книге устаревшие.)

`get_sb()` - ее название теперь `mount()`. Она вызывается когда выполняется монтирование фс, которое происходит из строки. При монтировании файловой системы создается суперблок. Мы определяем поля, определяем `super_operations`, это все должно быть описано, но все это начинает выполняться, когда мы выполняем монтирование, т е на самом деле это точка входа. Мы пишем модуль ядра, в котором описываем тип фс, описываем свою функцию `mount`, но срабатывает она когда мы монтируем эту фс. Инициализируются соответствующие поля `struct super_block`. Например, инициализируются обращения к операциям на суперблоке. В результате будут заполнены соответствующие поля, связанные с конкретными физическими файлами. И уже можно начинать работу с файлами, доступ к которым обеспечивает данная фс.

Struct inode. В UNIX/LINUX все доступно как файл. ВСЕ НАХОЙ ФАЙЛ.

Такие устройства как жесткий диск, как оптические диски, флешки системой рассматриваются как файлы. В директории `def` мы видим перечень устройств, но для системы это файл.

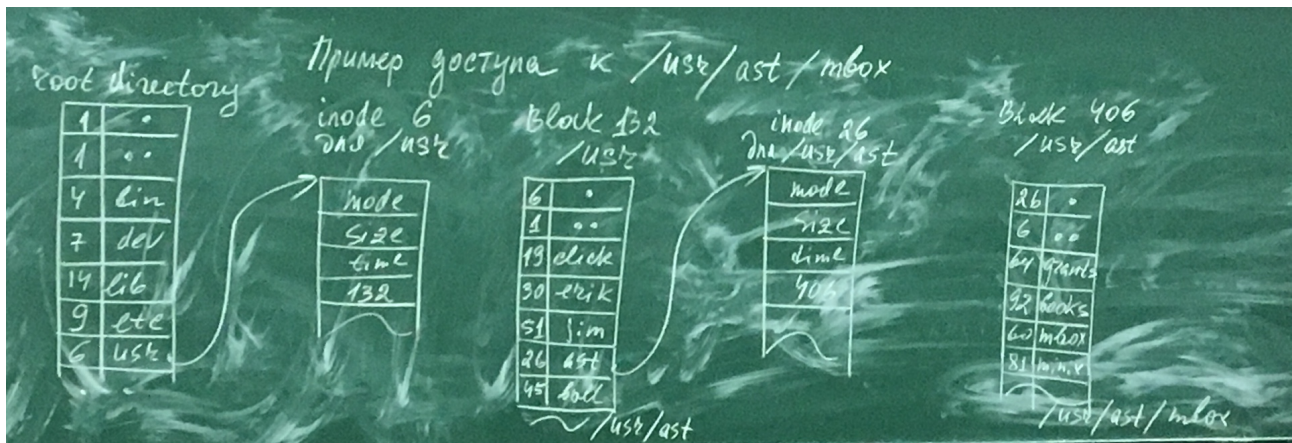
Еще раз повторим, что UNIX/LINUX поддерживают символьные имена в виде символьных строк. Такое именование файла удобно для пользователя. Имя файла в UNIX/LINUX в их родных файловых

30 марта, сб

подсистемах (EXT2) не является идентификатором. На самом деле идентификатором файла является номер inode, который принято называть метаданными.

Система, для того, чтобы получить доступ к файлу ищет его номер inode в таблице, которая называется таблицей inode'ов.

Давайте те же, дети мои, обратим очи на сию *УНАНУЮ* схему:



Тут показывается доступ, начиная с корневого каталога.

В системе существует структура `dentry`, которая создается, когда выполняется обращение к файлам/директориям. Очевидно, что каждый раз просматривая путь, мы фактически спускаемся по пути начиная с корневого каталога.

Структура inode:

```
struct inode {
    umode_t i_mode;
    unsigned short i_opflags;
    kuid_t i_uid;
    kgid_t i_gid;
    unsigned int i_flags;
    const struct inode_operations *i_op;
    struct super_block *i_sb;
    struct address_space *i_mapping;
};

/* Slot data, not accessed from path walking */
unsigned long i_ino;

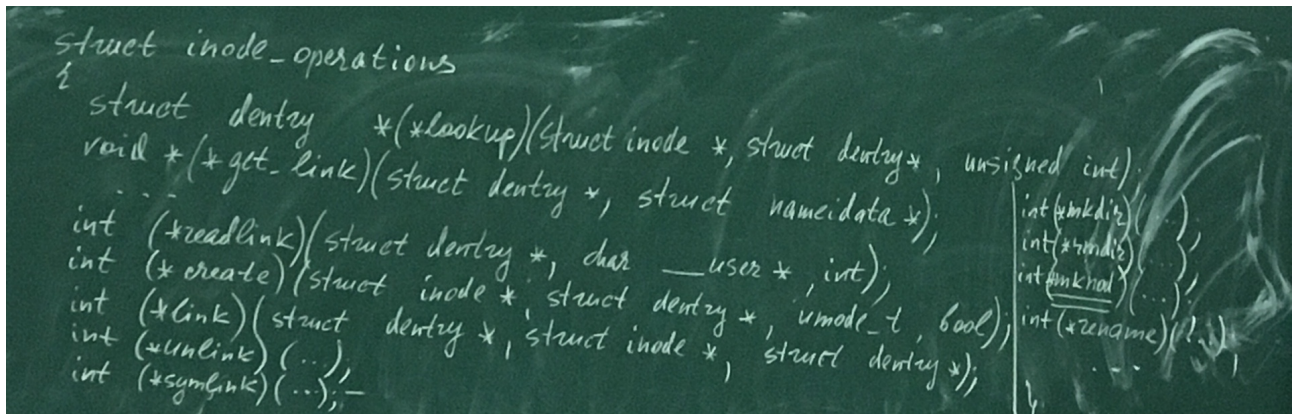
loff_t i_size;
struct timespec i_atime;
struct timespec i_mtime;
struct timespec i_ctime;
struct hlist_node i_hash;
const struct file_operations *i_op;
```

30 марта, сб

Мы видим, что в этой структуре присутствует указатель на struct inode_operations

В этой же структуре находится информация об устройствах.

struct inode_operations:



```
struct inode_operations
{
    struct dentry *(*lookup)(struct inode *, struct dentry *, unsigned int);
    void *(*get_link)(struct dentry *, struct nameidata *);
    int (*readlink)(struct dentry *, char __user *, int);
    int (*create)(struct inode *, struct dentry *, umode_t, bool);
    int (*link)(struct dentry *, struct inode *, struct dentry *);
    int (*unlink)(...);
    int (*symlink)(...);
    int (*mkdir)(...);
    int (*mknod)(...);
    int (*rmdir)(...);
    int (*rename)(...);
}
```

link - это операция, связанная с inode'ом, в частности можно рассмотреть, что делается в системе, когда вызывается link.

Любой системный вызов связан с последовательным вызовом других функций.