**Создание файловой системы в Linux**

Семинар (ИУ7, Рязанова Н.Ю.)

**Коротко о VFS**

Виртуальная файловая система (Virtual File System, известная также как Virtual Fileystem Switch) - это подсистема в ядре Linux, который обеспечивает интерфейс файловой системы для программ пользовательского пространства. Он также предоставляет абстракцию в ядре, которая позволяет сосуществовать различным реализациям файловой системы.

VFS определяет интерфейс, который должны поддерживать конкретные файловые системы, чтобы работать в Linux.

В Linux всегда существует корневая файловая система типа rootfs (“/”), а в ее поддиректории /home может быть смонтирована ФС такая, как ext(2,3,4), nfs, и т.д.

VFS определяет четыре базовые абстракции:

* ***dentry***

VFS реализует системные вызовы open(2), stat(2), chmod(2) и аналогичные. Переданный им аргумент pathname используется VFS для поиска в кэше записей каталога или в кэше dentry, который называется dcache. Это обеспечивает очень быстрый механизм поиска для перевода пути (имени файла) в конкретный dentry. Dentries находятся в RAM и никогда не сохраняются на диск: они существуют только для увеличения производительности.

Поскольку как правило ограниченный объем ОЗУ не может вместить все dentry одновременно, некоторые dentry в кеше отсутствуют. Чтобы преобразовать путь к файлу в dentry, VFS, возможно, придется в процессе обращения создать dentries, а затем загрузить inode. Это делается путем поиска в иноде (inode).

*/\**

*\* dentry->d\_lock spinlock nesting subclasses:*

*\**

*\* 0: normal*

*\* 1: nested*

*\*/*

enum [**dentry\_d\_lock\_class**](https://elixir.bootlin.com/linux/latest/ident/dentry_d_lock_class)

{

[**DENTRY\_D\_LOCK\_NORMAL**](https://elixir.bootlin.com/linux/latest/ident/DENTRY_D_LOCK_NORMAL), */\* implicitly used by plain spin\_lock() APIs. \*/*

[**DENTRY\_D\_LOCK\_NESTED**](https://elixir.bootlin.com/linux/latest/ident/DENTRY_D_LOCK_NESTED)

};

struct [**dentry\_operations**](https://elixir.bootlin.com/linux/latest/ident/dentry_operations) {

int (\***[d\_revalidate](https://elixir.bootlin.com/linux/latest/ident/d_revalidate)**)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, unsigned int);

int (\*d\_weak\_revalidate)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, unsigned int);

int (\***[d\_hash](https://elixir.bootlin.com/linux/latest/ident/d_hash)**)(const struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, struct [**qstr**](https://elixir.bootlin.com/linux/latest/ident/qstr) \*);

int (\*d\_compare)(const struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, unsigned int, const char \*, const struct [**qstr**](https://elixir.bootlin.com/linux/latest/ident/qstr) \*);

int (\***[d\_delete](https://elixir.bootlin.com/linux/latest/ident/d_delete)**)(const struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\*d\_init)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

void (\*d\_release)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

void (\*d\_prune)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

void (\*d\_iput)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*);

char \*(\*d\_dname)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, char \*, int);

struct [**vfsmount**](https://elixir.bootlin.com/linux/latest/ident/vfsmount) \*(\*d\_automount)(struct [**path**](https://elixir.bootlin.com/linux/latest/ident/path) \*);

int (\*d\_manage)(const struct [**path**](https://elixir.bootlin.com/linux/latest/ident/path) \*, [**bool**](https://elixir.bootlin.com/linux/latest/ident/bool));

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*(\***[d\_real](https://elixir.bootlin.com/linux/latest/ident/d_real)**)(struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, const struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*);

} [**\_\_\_\_cacheline\_aligned**](https://elixir.bootlin.com/linux/latest/ident/____cacheline_aligned);

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) {

*/\* RCU lookup touched fields \*/*

unsigned int d\_flags; */\* protected by d\_lock \*/*

[**seqcount\_t**](https://elixir.bootlin.com/linux/latest/ident/seqcount_t) d\_seq; */\* per dentry seqlock \*/*

struct [**hlist\_bl\_node**](https://elixir.bootlin.com/linux/latest/ident/hlist_bl_node) [**d\_hash**](https://elixir.bootlin.com/linux/latest/ident/d_hash); */\* lookup hash list \*/*

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*d\_parent; */\* parent directory \*/*

struct [**qstr**](https://elixir.bootlin.com/linux/latest/ident/qstr) d\_name;

struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \***[d\_inode](https://elixir.bootlin.com/linux/latest/ident/d_inode)**; */\* Where the name belongs to - NULL is \* negative \*/*

unsigned char d\_iname[[**DNAME\_INLINE\_LEN**](https://elixir.bootlin.com/linux/latest/ident/DNAME_INLINE_LEN)]; */\* small names \*/*

*/\* Ref lookup also touches following \*/*

struct [**lockref**](https://elixir.bootlin.com/linux/latest/ident/lockref) d\_lockref; */\* per-dentry lock and refcount \*/*

const struct [**dentry\_operations**](https://elixir.bootlin.com/linux/latest/ident/dentry_operations) \*d\_op;

struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*d\_sb; */\* The root of the dentry tree \*/*

unsigned long d\_time; */\* used by d\_revalidate \*/*

void \*d\_fsdata; */\* fs-specific data \*/*

union {

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) d\_lru; */\* LRU list \*/*

[**wait\_queue\_head\_t**](https://elixir.bootlin.com/linux/latest/ident/wait_queue_head_t) \***[d\_wait](https://elixir.bootlin.com/linux/latest/ident/d_wait)**; */\* in-lookup ones only \*/*

};

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) d\_child; */\* child of parent list \*/*

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) d\_subdirs; */\* our children \*/*

*/\**

*\* d\_alias and d\_rcu can share memory*

*\*/*

union {

struct [**hlist\_node**](https://elixir.bootlin.com/linux/latest/ident/hlist_node) d\_alias; */\* inode alias list \*/*

struct [**hlist\_bl\_node**](https://elixir.bootlin.com/linux/latest/ident/hlist_bl_node) d\_in\_lookup\_hash; */\* only for in-lookup ones \*/*

struct [**rcu\_head**](https://elixir.bootlin.com/linux/latest/ident/rcu_head) d\_rcu;

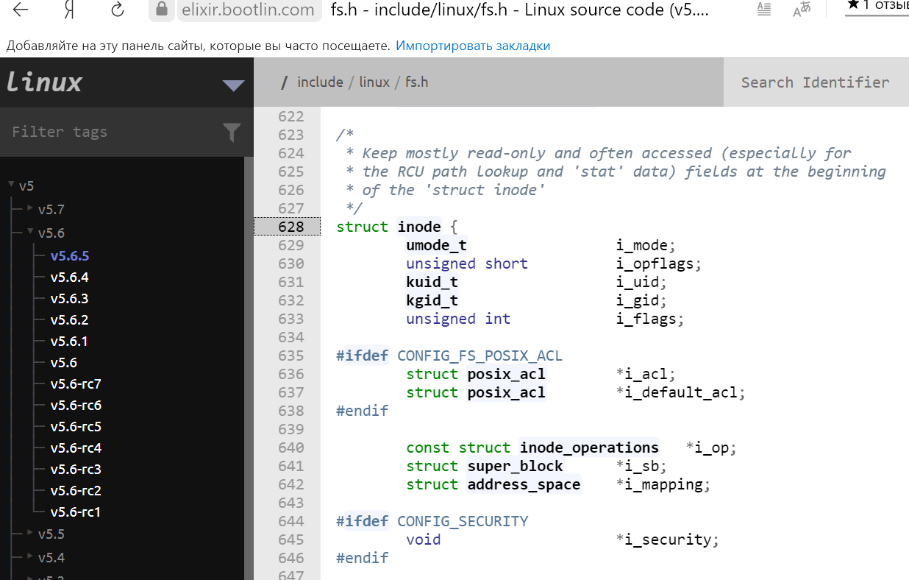
} d\_u;

} [**\_\_randomize\_layout**](https://elixir.bootlin.com/linux/latest/ident/__randomize_layout);

Структура dentry содержит указатель на суперблок и это корневой каталог дерева директорий.

* ***inode***

Отдельный dentry обычно имеет указатель на inode. inode - это объекты файловой системы, такие как обычные файлы, каталоги, FIFO и другие сущности. Они находятся либо на диске (для файловых систем блочных устройств), либо в памяти (для псевдофайловых систем). Иноды, которые находятся на диске, при необходимости копируются в память, а изменения в иноде записываются обратно на диск. На один инод может указывать несколько denties (например, это делают жесткие ссылки).



На inode определены следующие операции:

struct [**inode\_operations**](https://elixir.bootlin.com/linux/latest/ident/inode_operations) {

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \* (\*[**lookup**](https://elixir.bootlin.com/linux/latest/ident/lookup)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, unsigned int);

const char \* (\***[get\_link](https://elixir.bootlin.com/linux/latest/ident/get_link)**) (struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**delayed\_call**](https://elixir.bootlin.com/linux/latest/ident/delayed_call) \*);

int (\*[**permission**](https://elixir.bootlin.com/linux/latest/ident/permission)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, int);

struct [**posix\_acl**](https://elixir.bootlin.com/linux/latest/ident/posix_acl) \* (\***[get\_acl](https://elixir.bootlin.com/linux/latest/ident/get_acl)**)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, int);

int (\***[readlink](https://elixir.bootlin.com/linux/latest/ident/readlink)**) (struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, char [**\_\_user**](https://elixir.bootlin.com/linux/latest/ident/__user) \*,int);

int (\*[**create**](https://elixir.bootlin.com/linux/latest/ident/create)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, [**umode\_t**](https://elixir.bootlin.com/linux/latest/ident/umode_t), [**bool**](https://elixir.bootlin.com/linux/latest/ident/bool));

int (\*[**link**](https://elixir.bootlin.com/linux/latest/ident/link)) (struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*,struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\*[**unlink**](https://elixir.bootlin.com/linux/latest/ident/unlink)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\***[symlink](https://elixir.bootlin.com/linux/latest/ident/symlink)**) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*,const char \*);

int (\***[mkdir](https://elixir.bootlin.com/linux/latest/ident/mkdir)**) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*,**[umode\_t](https://elixir.bootlin.com/linux/latest/ident/umode_t)**);

int (\***[rmdir](https://elixir.bootlin.com/linux/latest/ident/rmdir)**) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\***[mknod](https://elixir.bootlin.com/linux/latest/ident/mknod)**) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*,struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*,**[umode\_t](https://elixir.bootlin.com/linux/latest/ident/umode_t)**,[**dev\_t**](https://elixir.bootlin.com/linux/latest/ident/dev_t));

int (\*[**rename**](https://elixir.bootlin.com/linux/latest/ident/rename)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*,

struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, unsigned int);

int (\*setattr) (struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, struct [**iattr**](https://elixir.bootlin.com/linux/latest/ident/iattr) \*);

int (\*getattr) (const struct [**path**](https://elixir.bootlin.com/linux/latest/ident/path) \*, struct [**kstat**](https://elixir.bootlin.com/linux/latest/ident/kstat) \*, [**u32**](https://elixir.bootlin.com/linux/latest/ident/u32), unsigned int);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\***[listxattr](https://elixir.bootlin.com/linux/latest/ident/listxattr)**) (struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, char \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t));

int (\***[fiemap](https://elixir.bootlin.com/linux/latest/ident/fiemap)**)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**fiemap\_extent\_info**](https://elixir.bootlin.com/linux/latest/ident/fiemap_extent_info) \*, [**u64**](https://elixir.bootlin.com/linux/latest/ident/u64) start,

[**u64**](https://elixir.bootlin.com/linux/latest/ident/u64) len);

int (\***[update\_time](https://elixir.bootlin.com/linux/latest/ident/update_time)**)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**timespec64**](https://elixir.bootlin.com/linux/latest/ident/timespec64) \*, int);

int (\***[atomic\_open](https://elixir.bootlin.com/linux/latest/ident/atomic_open)**)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*,

struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, unsigned [**open\_flag**](https://elixir.bootlin.com/linux/latest/ident/open_flag),

[**umode\_t**](https://elixir.bootlin.com/linux/latest/ident/umode_t) create\_mode);

int (\*tmpfile) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, [**umode\_t**](https://elixir.bootlin.com/linux/latest/ident/umode_t));

int (\*set\_acl)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**posix\_acl**](https://elixir.bootlin.com/linux/latest/ident/posix_acl) \*, int);

} [**\_\_\_\_cacheline\_aligned**](https://elixir.bootlin.com/linux/latest/ident/____cacheline_aligned);

Для поиска inode требуется, чтобы VFS вызывала метод lookup() родительского каталога inode. Этот метод устанавливается конкретной реализацией файловой системы, в которой «живет» inode. Как только VFS находит требуемый dentry (и, следовательно, inode), можно открывать файл системным вызовом open(2) или получать информацию о файле функцией stat(2), которая. просматривает данные inode и передает часть их в пространство пользователя.

* ***file***

Для открытия файла требуется еще одна операция: выделение файловой структуры struct file - дескриптор файлов в системной таблице открытых файлов.

struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) {

union {

struct [**llist\_node**](https://elixir.bootlin.com/linux/latest/ident/llist_node) fu\_llist;

struct [**rcu\_head**](https://elixir.bootlin.com/linux/latest/ident/rcu_head) fu\_rcuhead;

} f\_u;

struct [**path**](https://elixir.bootlin.com/linux/latest/ident/path) f\_path;

struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*f\_inode; */\* cached value \*/*

const struct [**file\_operations**](https://elixir.bootlin.com/linux/latest/ident/file_operations) \*f\_op;

*/\**

*\* Protects f\_ep\_links, f\_flags.*

*\* Must not be taken from IRQ context.*

*\*/*

[**spinlock\_t**](https://elixir.bootlin.com/linux/latest/ident/spinlock_t) f\_lock;

enum [**rw\_hint**](https://elixir.bootlin.com/linux/latest/ident/rw_hint) f\_write\_hint;

[**atomic\_long\_t**](https://elixir.bootlin.com/linux/latest/ident/atomic_long_t) f\_count;

unsigned int f\_flags;

[**fmode\_t**](https://elixir.bootlin.com/linux/latest/ident/fmode_t) f\_mode;

struct [**mutex**](https://elixir.bootlin.com/linux/latest/ident/mutex) f\_pos\_lock;

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) f\_pos;

struct [**fown\_struct**](https://elixir.bootlin.com/linux/latest/ident/fown_struct) f\_owner;

const struct [**cred**](https://elixir.bootlin.com/linux/latest/ident/cred) \*f\_cred;

struct [**file\_ra\_state**](https://elixir.bootlin.com/linux/latest/ident/file_ra_state) f\_ra;

[**u64**](https://elixir.bootlin.com/linux/latest/ident/u64) f\_version;

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_SECURITY

void \*f\_security;

#endif

*/\* needed for tty driver, and maybe others \*/*

void \***[private\_data](https://elixir.bootlin.com/linux/latest/ident/private_data)**;

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_EPOLL

*/\* Used by fs/eventpoll.c to link all the hooks to this file \*/*

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) f\_ep\_links;

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) f\_tfile\_llink;

#endif */\* #ifdef CONFIG\_EPOLL \*/*

struct [**address\_space**](https://elixir.bootlin.com/linux/latest/ident/address_space) \*f\_mapping;

[**errseq\_t**](https://elixir.bootlin.com/linux/latest/ident/errseq_t) f\_wb\_err;

} [**\_\_randomize\_layout**](https://elixir.bootlin.com/linux/latest/ident/__randomize_layout)

Структура struct file инициализируется указателем на dentry и вектором файловых операций, которые берутся из данных, которые адресуются с помощью inode. Затем вызывается метод файла open(), чтобы конкретная реализация файловой системы могла выполнить свою работу. Структура файла помещается в таблицу дескрипторов файлов, открытых процессом.

Чтение, запись и закрытие файлов (и другие различные операции VFS) выполняются с помощью дескриптора файла пространства пользователя, чтобы получить соответствующую структуру файла, а затем вызывая требуемый метод структуры файла, чтобы сделать все, что требуется. Пока файл открыт, он поддерживает использование dentry, что, в свою очередь, означает, что индекс VFS все еще используется.

Структура struct file содержит указатель на вектор операций, определенных на файле:

struct [**file\_operations**](https://elixir.bootlin.com/linux/latest/ident/file_operations) {

struct [**module**](https://elixir.bootlin.com/linux/latest/ident/module) \*[**owner**](https://elixir.bootlin.com/linux/latest/ident/owner);

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) (\***[llseek](https://elixir.bootlin.com/linux/latest/ident/llseek)**) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), int);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*[**read**](https://elixir.bootlin.com/linux/latest/ident/read)) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, char [**\_\_user**](https://elixir.bootlin.com/linux/latest/ident/__user) \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) \*);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*[**write**](https://elixir.bootlin.com/linux/latest/ident/write)) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, const char [**\_\_user**](https://elixir.bootlin.com/linux/latest/ident/__user) \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) \*);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*read\_iter) (struct [**kiocb**](https://elixir.bootlin.com/linux/latest/ident/kiocb) \*, struct [**iov\_iter**](https://elixir.bootlin.com/linux/latest/ident/iov_iter) \*);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*write\_iter) (struct [**kiocb**](https://elixir.bootlin.com/linux/latest/ident/kiocb) \*, struct [**iov\_iter**](https://elixir.bootlin.com/linux/latest/ident/iov_iter) \*);

int (\*iopoll)(struct [**kiocb**](https://elixir.bootlin.com/linux/latest/ident/kiocb) \***[kiocb](https://elixir.bootlin.com/linux/latest/ident/kiocb)**, [**bool**](https://elixir.bootlin.com/linux/latest/ident/bool) [**spin**](https://elixir.bootlin.com/linux/latest/ident/spin));

int (\*iterate) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, struct [**dir\_context**](https://elixir.bootlin.com/linux/latest/ident/dir_context) \*);

int (\*iterate\_shared) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, struct [**dir\_context**](https://elixir.bootlin.com/linux/latest/ident/dir_context) \*);

[**\_\_poll\_t**](https://elixir.bootlin.com/linux/latest/ident/__poll_t) (\*[**poll**](https://elixir.bootlin.com/linux/latest/ident/poll)) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, struct [**poll\_table\_struct**](https://elixir.bootlin.com/linux/latest/ident/poll_table_struct) \*);

long (\*unlocked\_ioctl) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, unsigned int, unsigned long);

long (\*compat\_ioctl) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, unsigned int, unsigned long);

int (\***[mmap](https://elixir.bootlin.com/linux/latest/ident/mmap)**) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, struct [**vm\_area\_struct**](https://elixir.bootlin.com/linux/latest/ident/vm_area_struct) \*);

unsigned long mmap\_supported\_flags;

int (\*[**open**](https://elixir.bootlin.com/linux/latest/ident/open)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*);

int (\*[**flush**](https://elixir.bootlin.com/linux/latest/ident/flush)) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**fl\_owner\_t**](https://elixir.bootlin.com/linux/latest/ident/fl_owner_t) id);

int (\*[**release**](https://elixir.bootlin.com/linux/latest/ident/release)) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*);

int (\***[fsync](https://elixir.bootlin.com/linux/latest/ident/fsync)**) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), int datasync);

int (\***[fasync](https://elixir.bootlin.com/linux/latest/ident/fasync)**) (int, struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, int);

int (\*lock) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, int, struct [**file\_lock**](https://elixir.bootlin.com/linux/latest/ident/file_lock) \*);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*sendpage) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, struct page \*, int, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) \*, int);

unsigned long (\***[get\_unmapped\_area](https://elixir.bootlin.com/linux/latest/ident/get_unmapped_area)**)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, unsigned long, unsigned long, unsigned long, unsigned long);

int (\***[check\_flags](https://elixir.bootlin.com/linux/latest/ident/check_flags)**)(int);

int (\*[**flock**](https://elixir.bootlin.com/linux/latest/ident/flock)) (struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, int, struct [**file\_lock**](https://elixir.bootlin.com/linux/latest/ident/file_lock) \*);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*splice\_write)(struct [**pipe\_inode\_info**](https://elixir.bootlin.com/linux/latest/ident/pipe_inode_info) \*, struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), unsigned int);

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*splice\_read)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) \*, struct [**pipe\_inode\_info**](https://elixir.bootlin.com/linux/latest/ident/pipe_inode_info) \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), unsigned int);

int (\***[setlease](https://elixir.bootlin.com/linux/latest/ident/setlease)**)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, long, struct [**file\_lock**](https://elixir.bootlin.com/linux/latest/ident/file_lock) \*\*, void \*\*);

long (\***[fallocate](https://elixir.bootlin.com/linux/latest/ident/fallocate)**)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*[**file**](https://elixir.bootlin.com/linux/latest/ident/file), int mode, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) offset,

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) len);

void (\***[show\_fdinfo](https://elixir.bootlin.com/linux/latest/ident/show_fdinfo)**)(struct [**seq\_file**](https://elixir.bootlin.com/linux/latest/ident/seq_file) \*m, struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*f);

#ifndef CONFIG\_MMU

unsigned (\*mmap\_capabilities)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*);

#endif

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\***[copy\_file\_range](https://elixir.bootlin.com/linux/latest/ident/copy_file_range)**)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*,

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), unsigned int);

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) (\*remap\_file\_range)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*file\_in, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) pos\_in,

struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*file\_out, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) pos\_out,

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) len, unsigned int remap\_flags);

int (\*fadvise)(struct [**file**](https://elixir.bootlin.com/linux/latest/ident/file) \*, [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t), int);

} [**\_\_randomize\_layout**](https://elixir.bootlin.com/linux/latest/ident/__randomize_layout);

* ***superblock и монтирование***

Регулярные (обычные) файлы принадлежат конкретным файловым системам. К ним же принадлежат и файлы, хранящие denry. Чтобы получить доступ к файловой системе в Linux, сначала нужно ее смонтировать.

Дерево каталогов и файлов ОС Linux формируется из ветвей, которые соответствуют различным физическим носителям. Часто говорят, что оно формируется из отдельных файловых систем. Файловая система Linux это – одно большое дерево с корневой директорией “/”. При монтировании фактически монтируется файловая система какого-либо устройства.

Монтирование корневой файловой системы (/) является частью процесса инициализации ОС. Все остальные файловые системы невозможно использовать до тех пор, пока они не будут смонтированы в определенных точках монтирования. Файловые системы, перечисленные в /etc/fstab монтируются в процессе загрузки системы. Файл / etc / fstab содержит список записей в следующей форме:

[File System] [Mount Point] [File System Type] [Options] [Dump] [Pass]

Точкой монтирования является обычная директория дерева каталогов. Монтирование это – система действий, в результате которой файловая система устройства становиться доступной.

Чтобы отсоединить смонтированную файловую систему, используется команда umount, за которой следует либо каталог, в котором она была смонтирована (точка монтирования), либо имя устройства:

$ umount DIRECTORY[[1]](#footnote-1)

$ umount DEVICE\_NAME

Монтировать файловую систему просто означает сделать конкретную файловую систему доступной в определенный момент в дереве каталогов Linux. При монтировании файловой системы не имеет значения, является ли файловая система разделом жесткого диска, компакт-диска, дискеты или запоминающего устройства USB. Для монтирования нужно знать имя устройства, связанного с конкретным устройством хранения, и каталог, к которому файловая система монтируется.

Прежде чем можно будет смонтировать файловую систему в определенный каталог, нужно войти в систему как root (некоторые файловые системы могут монтироваться обычным пользователем). Каталог, в который монтируется файловая система, должен существовать. Кроме того, в некоторых ситуациях в систему нужно входить как пользователь root, чтобы создать конкретный каталог монтирования. Если каталог существует, и любой пользователь может подключить это конкретное устройство, нет необходимости входить в систему как пользователь root. При монтировании конкретной файловой системы или устройства необходимо знать файл специального устройства, связанный с ним. Файл устройства это - специальный файл в операционных системах Unix / Linux, который позволяет программам и пользователю напрямую связываться с различными разделами и устройствами на компьютере. Эти файлы устройств находятся в папке /dev.

Для монтирования файловой системы в заданном месте (точке монтирования) в Linux используется команда mount в следующей форме:

mount [OPTION...] DEVICE\_NAME DIRECTORY

После подключения файловой системы точка монтирования становится корневым каталогом смонтированной файловой системы.

В качестве первого примера, рассмотрим реальный пример доступа файлам Windows с дискеты в Linux. Чтобы подключить устройство к определенной папке, эта папка должна существовать. Многие дистрибутивы Linux заранее будут содержать папку /mnt или даже папку /mnt /floppy, которая используется для монтирования различных устройств. Если папка, в которую вы хотите смонтировать устройство, существует, то все готово. Если нет, вам нужно создать его так:

**$ mkdir / mnt / floppy**

Эта команда создаст каталог с именем / mnt / floppy. Следующим шагом будет монтирование файловой системы в эту папку или точку монтирования.

**$ mount -t msdos /dev/fd0 /mnt/floppy**

В результате будет смонтирована файловая система msdos, на что указывает опция -t (тип). Устройство распознается точкой / mnt / floppy. Теперь можно получить доступ к дискам в формате MS-DOS, как и к любому другому каталогу.

Чтобы смонтировать CD-ROM:

**$ mount -t iso9660 / dev / cdrom / mnt / cdrom**

Опять же, это тот же метод, что и выше, для монтирования CD-ROM.

Различные файловые системы также могут быть смонтированы одинаково:

**$ mount -t vfat / dev / hda1 / win**

Любые файловые системы, которые не смонтированы, можно увидеть с помощью команды df.

*Примечание. Параметр* ***-t*** *используется для указания операционной системе конкретный тип файловой системы, к которому требуется подключить устройство. Если опция -t в команде не указана, mount будет пытаться определить правильный тип файловой системы, с которой должно монтироваться устройство.*

Самое простое определение суперблока, что суперблок это - метаданные файловой системы. Подобно тому, как inode хранит метаданные файлов, superblock хранит метаданные файловой системы. Суперблок это - структура, представляющая смонтированную файловую систему. Он включает в себя необходимую информацию для управления файловой системой во время работы. Он включает имя файловой системы (например, ext2), размер файловой системы и ее состояние, ссылку на блочное устройство и информацию метаданных (например, свободные списки и т. д.).

struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) {

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) s\_list; */\* Keep this first. Список всех суперблоков в системе \*/*

[**dev\_t**](https://elixir.bootlin.com/linux/latest/ident/dev_t) s\_dev; */\* search index; \_not\_ kdev\_t \*/*

unsigned char s\_blocksize\_bits;

unsigned long s\_blocksize;

[**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t) s\_maxbytes; */\* Max file size \*/*

struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \*s\_type;

const struct [**super\_operations**](https://elixir.bootlin.com/linux/latest/ident/super_operations) \*s\_op; /\*вектор операций, определенных на суперблоке\*/

const struct [**dquot\_operations**](https://elixir.bootlin.com/linux/latest/ident/dquot_operations) \*dq\_op;

const struct [**quotactl\_ops**](https://elixir.bootlin.com/linux/latest/ident/quotactl_ops) \*s\_qcop;

const struct [**export\_operations**](https://elixir.bootlin.com/linux/latest/ident/export_operations) \*s\_export\_op;

unsigned long s\_flags; /\*флаги монтирования\*/

unsigned long s\_iflags; */\* internal SB\_I\_\* flags \*/*

unsigned long s\_magic; /\*магическое число\*/

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*s\_root; /\*корневой каталог файловой системы – точка монтирования\*/

struct [**rw\_semaphore**](https://elixir.bootlin.com/linux/latest/ident/rw_semaphore) s\_umount; /\*семафор размонтирования\*/

int s\_count; /\*счетчик ссылок на суперблок\*/

[**atomic\_t**](https://elixir.bootlin.com/linux/latest/ident/atomic_t) s\_active; /\*счетчик активных ссылок\*/

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_SECURITY

void \*s\_security;

#endif

const struct [**xattr\_handler**](https://elixir.bootlin.com/linux/latest/ident/xattr_handler) \*\*s\_xattr;

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_FS\_ENCRYPTION

const struct [**fscrypt\_operations**](https://elixir.bootlin.com/linux/latest/ident/fscrypt_operations) \*s\_cop;

struct [**key**](https://elixir.bootlin.com/linux/latest/ident/key) \*s\_master\_keys; */\* master crypto keys in use \*/*

#endif

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_FS\_VERITY

const struct [**fsverity\_operations**](https://elixir.bootlin.com/linux/latest/ident/fsverity_operations) \*s\_vop;

#endif

struct [**hlist\_bl\_head**](https://elixir.bootlin.com/linux/latest/ident/hlist_bl_head) s\_roots; */\* alternate root dentries for NFS \*/*

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) s\_mounts; */\* list of mounts; \_not\_ for fs use \*/*

struct [**block\_device**](https://elixir.bootlin.com/linux/latest/ident/block_device) \*s\_bdev;

struct [**backing\_dev\_info**](https://elixir.bootlin.com/linux/latest/ident/backing_dev_info) \*s\_bdi;

struct [**mtd\_info**](https://elixir.bootlin.com/linux/latest/ident/mtd_info) \*s\_mtd;

struct [**hlist\_node**](https://elixir.bootlin.com/linux/latest/ident/hlist_node) s\_instances;

unsigned int s\_quota\_types; */\* Bitmask of supported quota types \*/*

struct [**quota\_info**](https://elixir.bootlin.com/linux/latest/ident/quota_info) s\_dquot; */\* Diskquota specific options \*/*

struct [**sb\_writers**](https://elixir.bootlin.com/linux/latest/ident/sb_writers) s\_writers;

*/\**

*\* Keep s\_fs\_info, s\_time\_gran, s\_fsnotify\_mask, and*

*\* s\_fsnotify\_marks together for cache efficiency. They are frequently*

*\* accessed and rarely modified.*

*\*/*

void \*s\_fs\_info; */\* Filesystem private info \*/*

*/\* Granularity of c/m/atime in ns (cannot be worse than a second) \*/*

[**u32**](https://elixir.bootlin.com/linux/latest/ident/u32) s\_time\_gran;

*/\* Time limits for c/m/atime in seconds \*/*

[**time64\_t**](https://elixir.bootlin.com/linux/latest/ident/time64_t) s\_time\_min;

[**time64\_t**](https://elixir.bootlin.com/linux/latest/ident/time64_t) s\_time\_max;

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_FSNOTIFY

[**\_\_u32**](https://elixir.bootlin.com/linux/latest/ident/__u32) s\_fsnotify\_mask;

struct [**fsnotify\_mark\_connector**](https://elixir.bootlin.com/linux/latest/ident/fsnotify_mark_connector) [**\_\_rcu**](https://elixir.bootlin.com/linux/latest/ident/__rcu) \*s\_fsnotify\_marks;

#endif

char s\_id[32]; */\* Informational name - текстовое имя\*/*

[**uuid\_t**](https://elixir.bootlin.com/linux/latest/ident/uuid_t) s\_uuid; */\* UUID \*/*

unsigned int s\_max\_links;

[**fmode\_t**](https://elixir.bootlin.com/linux/latest/ident/fmode_t) s\_mode;

*/\**

*\* The next field is for VFS \*only\*. No filesystems have any business*

*\* even looking at it. You had been warned.*

*\*/*

struct [**mutex**](https://elixir.bootlin.com/linux/latest/ident/mutex) s\_vfs\_rename\_mutex; */\* Kludge \*/*

*/\**

*\* Filesystem subtype. If non-empty the filesystem type field*

*\* in /proc/mounts will be "type.subtype"*

*\*/*

const char \*s\_subtype;

const struct [**dentry\_operations**](https://elixir.bootlin.com/linux/latest/ident/dentry_operations) \*s\_d\_op; */\* default d\_op for dentries \*/*

*/\**

*\* Saved pool identifier for cleancache (-1 means none)*

*\*/*

int cleancache\_poolid;

struct [**shrinker**](https://elixir.bootlin.com/linux/latest/ident/shrinker) s\_shrink; */\* per-sb shrinker handle \*/*

*/\* Number of inodes with nlink == 0 but still referenced \*/*

[**atomic\_long\_t**](https://elixir.bootlin.com/linux/latest/ident/atomic_long_t) s\_remove\_count;

*/\* Pending fsnotify inode refs \*/*

[**atomic\_long\_t**](https://elixir.bootlin.com/linux/latest/ident/atomic_long_t) s\_fsnotify\_inode\_refs;

*/\* Being remounted read-only \*/*

int s\_readonly\_remount;

*/\* AIO completions deferred from interrupt context \*/*

struct [**workqueue\_struct**](https://elixir.bootlin.com/linux/latest/ident/workqueue_struct) \*s\_dio\_done\_wq;

struct [**hlist\_head**](https://elixir.bootlin.com/linux/latest/ident/hlist_head) s\_pins;

*/\**

*\* Owning user namespace and default context in which to*

*\* interpret filesystem uids, gids, quotas, device nodes,*

*\* xattrs and security labels.*

*\*/*

struct [**user\_namespace**](https://elixir.bootlin.com/linux/latest/ident/user_namespace) \*s\_user\_ns;

*/\**

*\* The list\_lru structure is essentially just a pointer to a table*

*\* of per-node lru lists, each of which has its own spinlock.*

*\* There is no need to put them into separate cachelines.*

*\*/*

struct [**list\_lru**](https://elixir.bootlin.com/linux/latest/ident/list_lru) s\_dentry\_lru;

struct [**list\_lru**](https://elixir.bootlin.com/linux/latest/ident/list_lru) s\_inode\_lru;

struct [**rcu\_head**](https://elixir.bootlin.com/linux/latest/ident/rcu_head) rcu;

struct [**work\_struct**](https://elixir.bootlin.com/linux/latest/ident/work_struct) destroy\_work;

struct [**mutex**](https://elixir.bootlin.com/linux/latest/ident/mutex) s\_sync\_lock; */\* sync serialisation lock \*/*

*/\**

*\* Indicates how deep in a filesystem stack this SB is*

*\*/*

int s\_stack\_depth;

*/\* s\_inode\_list\_lock protects s\_inodes \*/*

[**spinlock\_t**](https://elixir.bootlin.com/linux/latest/ident/spinlock_t) s\_inode\_list\_lock [**\_\_\_\_cacheline\_aligned\_in\_smp**](https://elixir.bootlin.com/linux/latest/ident/____cacheline_aligned_in_smp);

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) s\_inodes; */\* all inodes – список всех inode \*/*

[**spinlock\_t**](https://elixir.bootlin.com/linux/latest/ident/spinlock_t) s\_inode\_wblist\_lock;

struct [**list\_head**](https://elixir.bootlin.com/linux/latest/ident/list_head) s\_inodes\_wb; */\* writeback inodes \*/*

} [**\_\_randomize\_layout**](https://elixir.bootlin.com/linux/latest/ident/__randomize_layout);

Суперблок обычно хранится во вторичной памяти, но может быть создан и в оперативной памяти.

Одним из важных элементов суперблока является определение операций суперблока. Эта структура определяет набор функций для управления inode в файловой системе.

struct [**super\_operations**](https://elixir.bootlin.com/linux/latest/ident/super_operations) {

struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*(\***[alloc\_inode](https://elixir.bootlin.com/linux/latest/ident/alloc_inode)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*sb); /\*создает и инициализирует новый объект inode, связанного с суперблоком\*/

void (\***[destroy\_inode](https://elixir.bootlin.com/linux/latest/ident/destroy_inode)**)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*); /\*уничтожает inode\*/

void (\***[free\_inode](https://elixir.bootlin.com/linux/latest/ident/free_inode)**)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*);

void (\*dirty\_inode) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, int flags);

int (\***[write\_inode](https://elixir.bootlin.com/linux/latest/ident/write_inode)**) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*, struct [**writeback\_control**](https://elixir.bootlin.com/linux/latest/ident/writeback_control) \*wbc);

int (\*drop\_inode) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*);

void (\*evict\_inode) (struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*);

void (\***[put\_super](https://elixir.bootlin.com/linux/latest/ident/put_super)**) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*);

int (\*sync\_fs)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*sb, int [**wait**](https://elixir.bootlin.com/linux/latest/ident/wait));

int (\***[freeze\_super](https://elixir.bootlin.com/linux/latest/ident/freeze_super)**) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*);

int (\*freeze\_fs) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*);

int (\***[thaw\_super](https://elixir.bootlin.com/linux/latest/ident/thaw_super)**) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*);

int (\*unfreeze\_fs) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*);

int (\***[statfs](https://elixir.bootlin.com/linux/latest/ident/statfs)**) (struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*, struct [**kstatfs**](https://elixir.bootlin.com/linux/latest/ident/kstatfs) \*);

int (\*remount\_fs) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, int \*, char \*);

void (\*umount\_begin) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*);

int (\*show\_options)(struct [**seq\_file**](https://elixir.bootlin.com/linux/latest/ident/seq_file) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\*show\_devname)(struct [**seq\_file**](https://elixir.bootlin.com/linux/latest/ident/seq_file) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\*show\_path)(struct [**seq\_file**](https://elixir.bootlin.com/linux/latest/ident/seq_file) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

int (\***[show\_stats](https://elixir.bootlin.com/linux/latest/ident/show_stats)**)(struct [**seq\_file**](https://elixir.bootlin.com/linux/latest/ident/seq_file) \*, struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*);

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_QUOTA

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*quota\_read)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, int, char \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t));

[**ssize\_t**](https://elixir.bootlin.com/linux/latest/ident/ssize_t) (\*quota\_write)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, int, const char \*, [**size\_t**](https://elixir.bootlin.com/linux/latest/ident/size_t), [**loff\_t**](https://elixir.bootlin.com/linux/latest/ident/loff_t));

struct [**dquot**](https://elixir.bootlin.com/linux/latest/ident/dquot) \*\*(\*get\_dquots)(struct [**inode**](https://elixir.bootlin.com/linux/latest/ident/inode) \*);

#endif

int (\***[bdev\_try\_to\_free\_page](https://elixir.bootlin.com/linux/latest/ident/bdev_try_to_free_page)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block)\*, struct page\*, [**gfp\_t**](https://elixir.bootlin.com/linux/latest/ident/gfp_t));

long (\*nr\_cached\_objects)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*,

struct [**shrink\_control**](https://elixir.bootlin.com/linux/latest/ident/shrink_control) \*);

long (\*free\_cached\_objects)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*,

struct [**shrink\_control**](https://elixir.bootlin.com/linux/latest/ident/shrink_control) \*);

};

Например, inode можно выделить с помощью alloc\_inode() или удалить с помощью destroy\_inode(). Вы можете читать и записывать inode с помощью read\_inode() и write\_inode() или синхронизировать файловую систему с sync\_fs. Структуру super\_operations можно найти в /linux/include/linux/fs.h. Каждая файловая система предоставляет свои собственные методы inode, которые реализуют операции и предоставляют общую абстракцию для уровня VFS.

**Описание файловых систем**

Linux поддерживает большое количество файловых систем. Для того, чтобы это было возможно в VFS определена структура, определяющая тип файловой системы - struct file\_system\_type:

struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) {

const char \*name;

int fs\_flags;

#define [**FS\_REQUIRES\_DEV**](https://elixir.bootlin.com/linux/latest/ident/FS_REQUIRES_DEV) 1 /\*требуется блочное устройство\*/

#define [**FS\_BINARY\_MOUNTDATA**](https://elixir.bootlin.com/linux/latest/ident/FS_BINARY_MOUNTDATA) 2 /\* с версии 2.6.5 монтируемые данные являются бинарными\*/

#define [**FS\_HAS\_SUBTYPE**](https://elixir.bootlin.com/linux/latest/ident/FS_HAS_SUBTYPE) 4

#define [**FS\_USERNS\_MOUNT**](https://elixir.bootlin.com/linux/latest/ident/FS_USERNS_MOUNT) 8 */\* Can be mounted by userns root \*/*

#define [**FS\_DISALLOW\_NOTIFY\_PERM**](https://elixir.bootlin.com/linux/latest/ident/FS_DISALLOW_NOTIFY_PERM) 16 */\* Disable fanotify permission events \*/*

#define [**FS\_RENAME\_DOES\_D\_MOVE**](https://elixir.bootlin.com/linux/latest/ident/FS_RENAME_DOES_D_MOVE) 32768 */\* FS will handle d\_move() during rename() internally. \*/*

int (\*init\_fs\_context)(struct [**fs\_context**](https://elixir.bootlin.com/linux/latest/ident/fs_context) \*);

const struct [**fs\_parameter\_spec**](https://elixir.bootlin.com/linux/latest/ident/fs_parameter_spec) \*[**parameters**](https://elixir.bootlin.com/linux/latest/ident/parameters);

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*(\*[**mount**](https://elixir.bootlin.com/linux/latest/ident/mount)) (struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \*, int, const char \*, void \*);

void (\*kill\_sb) (struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*); /\*прекращение доступа к суперблоку\*/

struct [**module**](https://elixir.bootlin.com/linux/latest/ident/module) \*[**owner**](https://elixir.bootlin.com/linux/latest/ident/owner); /\*счетчик ссылок на файловую систему\*/

struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \* next;

struct [**hlist\_head**](https://elixir.bootlin.com/linux/latest/ident/hlist_head) fs\_supers;

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) s\_lock\_key;

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) s\_umount\_key;

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) s\_vfs\_rename\_key;

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) s\_writers\_key[[**SB\_FREEZE\_LEVELS**](https://elixir.bootlin.com/linux/latest/ident/SB_FREEZE_LEVELS)];

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) i\_lock\_key;

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) i\_mutex\_key;

struct [**lock\_class\_key**](https://elixir.bootlin.com/linux/latest/ident/lock_class_key) i\_mutex\_dir\_key;

};

Для каждого типа файловой системы существует только одна структура file\_system\_file, независимо от того, сколько таких файловых систем смонтировано и смонтирован ли хотя бы один экземпляр файловой системы.

Функция mount() служит для монтирования файловой системы, определенной конкретной структурой file\_system\_type, и заполнения супер\_блока соответствующими данными. Структура struct super\_block содержит указатель на struct file\_system\_type.

Когда файловая система монтируется, создается структура struct vfsmount, которая представляет конкретный экземпляр файловой системы, или, другими словами, точка монтирования.

Определены следующие флаги монтирования:

#define [**MNT\_NOSUID**](https://elixir.bootlin.com/linux/latest/ident/MNT_NOSUID) 0x01 /\*запрещает использование флагов setuid и setgid\*/

#define [**MNT\_NODEV**](https://elixir.bootlin.com/linux/latest/ident/MNT_NODEV) 0x02 /\*запрещает доступ к файлам устройств\*/

#define [**MNT\_NOEXEC**](https://elixir.bootlin.com/linux/latest/ident/MNT_NOEXEC) 0x04 /\*запрещает выполнение программ\*/

#define [**MNT\_NOATIME**](https://elixir.bootlin.com/linux/latest/ident/MNT_NOATIME) 0x08

#define [**MNT\_NODIRATIME**](https://elixir.bootlin.com/linux/latest/ident/MNT_NODIRATIME) 0x10

#define [**MNT\_RELATIME**](https://elixir.bootlin.com/linux/latest/ident/MNT_RELATIME) 0x20

#define [**MNT\_READONLY**](https://elixir.bootlin.com/linux/latest/ident/MNT_READONLY) 0x40 */\* does the user want this to be r/o? \*/*

struct [**vfsmount**](https://elixir.bootlin.com/linux/latest/ident/vfsmount) {

struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \*mnt\_root; */\* root of the mounted tree \*/*

struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*mnt\_sb; */\* pointer to superblock \*/*

int mnt\_flags; /\*флаги монтирования\*/

} [**\_\_randomize\_layout**](https://elixir.bootlin.com/linux/latest/ident/__randomize_layout);

Структура содержит указатель на точку монтирования и указатель на супер\_блок, смонтированной файловой системы.

Функции монтирования:

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_BLOCK

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_bdev](https://elixir.bootlin.com/linux/latest/ident/mount_bdev)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, const char \***[dev\_name](https://elixir.bootlin.com/linux/latest/ident/dev_name)**, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int));

#else

static inline struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_bdev](https://elixir.bootlin.com/linux/latest/ident/mount_bdev)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, const char \***[dev\_name](https://elixir.bootlin.com/linux/latest/ident/dev_name)**, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int))

{

[**return**](https://elixir.bootlin.com/linux/latest/ident/return) [**ERR\_PTR**](https://elixir.bootlin.com/linux/latest/ident/ERR_PTR)(-[**ENODEV**](https://elixir.bootlin.com/linux/latest/ident/ENODEV));

}

#endif

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_single](https://elixir.bootlin.com/linux/latest/ident/mount_single)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int));

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_nodev](https://elixir.bootlin.com/linux/latest/ident/mount_nodev)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int));

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_subtree](https://elixir.bootlin.com/linux/latest/ident/mount_subtree)**(struct [**vfsmount**](https://elixir.bootlin.com/linux/latest/ident/vfsmount) \***[mnt](https://elixir.bootlin.com/linux/latest/ident/mnt)**, const char \*[**path**](https://elixir.bootlin.com/linux/latest/ident/path));

**Регистрация и монтирование файловой системы**

Для связывания файловой системы с VFS необходимо определить некий минимальный набор функций и структур данных:

* struct file\_system\_type является глобальным "определителем" файловой системы и содержит имя ФС, а также функции инициализации и уничтожения суперблока.
* struct super\_operations содержит набор функций работы с глобальными данными файловой системы. Здесь могут использоваться "заглушки", предоставляемые libfs.
* struct file\_operations определяет набор функций для работы с файлами. Для файлов счетчиков мы реализуем только три из них: open, read и write; для каталогов же используем "заглушки" libfs.

Код файловой системы может быть реализован или в виде загружаемого модуля ядра, или статически связан с ядром.

Для создания новой файловой системы надо заполнить поля структуры struct file\_system\_file и зарегистрировать файловую систему VFS с помощью следующих функции API:

**#include <linux / fs.h>**

**extern int register\_filesystem (struct file\_system\_type \*);**

**extern int unregister\_filesystem (struct file\_system\_type \*);**

Переданный функци **register\_filesystem() тип** struct file\_system\_type описывает файловую систему. Например:

**struct file\_system\_file my\_fs\_type = {**

**. owner = THIS\_MODULE,**

**. name = “my\_fs”,**

**. mount = my\_mount,**

**. kill\_sb = my\_kill\_super\_block,**

**. fs\_flags = FS\_REQUIRED\_DEV,**

**};**

Этот тип передается соответствующей функции монтирования, например:

**static dentry\* my\_mount(struct file\_system\_type \*type, int flags, char const \*dev, void \*data)**

**{**

**struct dentry\* const entry = mount\_bdev(type, flags, dev, data, my\_fill\_sb);**

**if(IS\_ERR(entry))**

**printk(KERN\_ERR “my\_fs mounting failed \n”);**

**else printk(KERN\_DEBUG “my\_fs mounted\n”);**

**return entry;**

**}**

Когда делается запрос на монтирование файловой системы в каталог в определенном пространстве имен, VFS вызывает соответствующий метод mount() для конкретной файловой системы.

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_BLOCK

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_bdev](https://elixir.bootlin.com/linux/latest/ident/mount_bdev)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, const char \***[dev\_name](https://elixir.bootlin.com/linux/latest/ident/dev_name)**, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int));

#else

static inline struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_bdev](https://elixir.bootlin.com/linux/latest/ident/mount_bdev)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, const char \***[dev\_name](https://elixir.bootlin.com/linux/latest/ident/dev_name)**, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int))

{

[**return**](https://elixir.bootlin.com/linux/latest/ident/return) [**ERR\_PTR**](https://elixir.bootlin.com/linux/latest/ident/ERR_PTR)(-[**ENODEV**](https://elixir.bootlin.com/linux/latest/ident/ENODEV));

}

#endif

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_single](https://elixir.bootlin.com/linux/latest/ident/mount_single)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int));

extern struct [**dentry**](https://elixir.bootlin.com/linux/latest/ident/dentry) \***[mount\_nodev](https://elixir.bootlin.com/linux/latest/ident/mount_nodev)**(struct [**file\_system\_type**](https://elixir.bootlin.com/linux/latest/ident/file_system_type) \***[fs\_type](https://elixir.bootlin.com/linux/latest/ident/fs_type)**,

int flags, void \*data,

int (\***[fill\_super](https://elixir.bootlin.com/linux/latest/ident/fill_super)**)(struct [**super\_block**](https://elixir.bootlin.com/linux/latest/ident/super_block) \*, void \*, int));

Передаваемая функции mount() функция fill\_super() заполняет поля структуры struct super\_block. Все перечисленные функции mount() возвращают указатель на struct dentry.

Все зарегистрированные файловые системы в ядре можно увидеть в поддиректории:

file / proc / filesystems

Как была сказано ранее файловая система разрабатывается в виде загружаемого модуля ядра. Например:

**static int \_\_init my\_fs\_init(void)**

**{**

**int ret = register\_filesystem(&my\_fs\_type);**

**if(ret!=0) {**

**printk(KERN\_ERR “my\_fs module cannot register file system\n”);**

**return ret;**

**}**

**printk(KERN\_DEBUG “my\_fs module loaded\n”);**

**}**

**static void \_\_exit my\_fs\_exit(void)**

**{**

**int ret = unregister\_filesystem(&my\_fs\_type);**

**if(ret!=0) {**

**printk(KERN\_ERR “my\_fs module cannot unregister file system\n”);**

**}**

**printk(KERN\_DEBUG “my\_fs module unloaded\n”);**

**}**

Таким образом можно определить алгоритм работы создаваемой ФС:

1. Точкой входа в модуль является функция my\_fs\_init, выполняющая регистрацию ФС вызовом register\_filesystem(&my\_fs\_type).
2. При монтировании ФС вызывается функция my\_fs\_type->mount(), указывающая на my\_mount(). Эта функция вызывает libfs-helper **mount\_bdev** (), передавая ей в качестве параметра указатель на my\_fill\_sb. **mount\_bdev**() размещает структуру struc super\_block, и инициализирует некоторыми значениями ее поля, вызывая в завершение my\_fill\_sb().
3. my\_fill\_sb() устанавливает некоторые значащие поля суперблока, а также создает и инициализируется логическую структуру файловой системы - файл "counter" в корневом каталоге, и файл "subcounter" в каталоге "subdir", размещая и вставляя в соответствующие VFS-кэши все необходимые структуры [1].

Если создается виртуальная файловая система, не связанная с каким-либо носителем, нет необходимости использовать функцию **mount\_bdev().** Вместо нее можно использовать функцию mount\_nodev(). Вместо собственной функции kill\_super() можно использовать функцию kill\_little\_super() -- это generic-функция, предоставляемая VFS, она просто освобождает все внутренние структуры при размонтировании ФС; т.о. авторы простых виртуальных файловых систем не должны заботится об этом аспекте (конечно, необходимо дерегистрировать ФС в момент выгрузки модуля, это будет реализовано в exit-функции модуля).

**Список использованных источников**

1. Пешеходов А.П. Создание виртуальных файловых систем в Linux (fs linux virtual) Документ составлен на основе статьи Jonathan Corbet "Creating virtual filesystems with libfs" (<http://lwn.net/Articles/57369/>). https://www.opennet.ru/base/dev/virtual\_fs.txt.html
2. VFS. Виртуальная файловая система. Часть 1 https://annimon.com/

1. Используйте опцию -f (--force), чтобы выполнить безусловное unmount. Этот параметр обычно используется для размонтирования недоступной системы NFS.

   umount -f DIRECTORY [↑](#footnote-ref-1)