Файловая система должна иметь опр тип, тип опред в юникс/линукс в…

struct\_file\_system\_type  
{  
 const char \*name;  
 int fs\_flags; //fs – File System  
 #define FS\_REQUIRES\_DEV 1;  
 #define FS\_BINARY\_MOUNTDATA 2; //монтируемые данные явл бинарными  
 #define FS\_HAS\_SUBTYPE 4;  
 #define FS\_USERNS\_MOUNT 8; //монтир. сист. Может быть смонтирована корневым, т.е. юзер создал директорию и к ней подмонтировался  
 #define FS\_REVAL\_DOT 16384;  
 …

файловые системы все разные, но могут быть несколько одной структуры

struct dentry \*(\*mount) (struct file\_system\_type \*, int, const char \*, void \*);

заменила собой функцию …(*позже скажет*), должна вернуть точку монтирования, dentry описывает компонент пути. Передаем тип файловой системы…

det\_sb() // создать файл.сист. это создать супер блок

Функция маунт вызывается, когда в командной строке набираем команду маунт, то есть когда монтируем файловую систему. Для того, чтобы смонтировать систему, надо написать функцию маунт, так как мы ей должны передать файловую систему, ее тип. Соответсвенно следующая точка входа это kill\_sb

Void (\*kill\_sb) (struct super\_block \*);

Уничтожить файл.сист. = уничтожить супер блок. Заходит на этой точке входа при функции unmount.

Struct modile \*owner;

Struct fill\_system\_tipe \*next;

Struct hlist\_head fs\_supers;

Файловая система должна позволять создавать файлы, директории, линки, позволять писать и читать файлы, и это все – точки входа.

s- имеет отношение к суперблоку, i- к inode

Struct lock\_class\_key s\_lock\_key;  
-----//-------------------- s\_unmount\_key;  
-----//-------------------- s\_vfs\_rename\_key;

Критические действия – отмонтирование и переименование, так как в этот момент кто-то еще может пользоваться ей, а мы ее монтируем; или чтобы 2 раза ее не монтировали(2 юзера одновременно). Тогда идет блокировка этих действий.

------//------------------- s\_writers\_key (SB\_FREEZE\_LEVELS);  
------//------------------- i\_lock\_key;  
------//------------------- i\_mutex\_key;  
------//------------------- i\_mutex\_dir\_key;

}

//исключение паскаля – про связные списки

По стандарту С99:

Struct file\_system\_type my\_fs\_type =

{

* Owner = THIS\_MODULE;
* Name = “my\_fs”;
* Mount = my\_mount;
* Kill\_sb = my\_kill\_superblock;
* Fs\_flags = FS\_REQUIRES\_DEV;

};

Struct fill\_operations\_fops =

{

Read: device\_read;  
write: device\_write;  
open: device\_open  
release: device\_release;

};

Owner – счетсчик ссылок – нужен, чтобы …, не позволит выгрузить модуль, пока файловая система не будет отмонтирована с помощью маунт(с командной строки)

Bdev – блок девайса, так как блочное устройство у нас жесткий диск, остальные устройства это символьные

Extern struct dentry \*mount\_bdev (struct file\_system\_type \*fs\_type, int flags, const char \*dev\_name, void \*data, int (\*fill\_super) (struct super\_block \*, void \*, int));

Если мы должны создать корневой каталог, то мы должны создать файл, соответственно inode.

Extern struct dentry \*mount\_nodev (struct fill\_system\_type \*fs\_type, int flags, void \*data, int (\*fill\_super)(struct super\_block \*, void \*, int));

Fill\_superblock – заполнить поля суперблока. Vfsmount структура заполняется автоматически.

Struct vfsmount  
{  
 struct dentry \*mnt\_root; //точка монтирования  
 struct super\_block \*mnt\_sb; //указатель на суперблок  
 int mnt \_flags;//флаги монтирования  
}

Флаги могут быть:

MNT\_NOSUID – запрещает использования флагов setuid, setgid для файлов в соответствующей файловой системе

MNT\_NODEV - Флаг запрещает доступ к флагу устройств

MNT\_NOEXEC - Запрещает выполнение программ

Для того, чтобы наш тип стал действовать, необходимо зарегестрировать файловую систему. Когда нужна в ней пропадает, ее надо разрегестрировать.

Static int\_\_\_init my\_init (void)  
{  
 int ret;  
 my\_inode\_cachep = kmem\_cache\_create(“my\_inode\_cachep”, sizeof(my\_inode), 0, (SLAB\_RECLAIM\_ACCOUNT | SLAB\_MEM\_SPREAD), NULL);  
 if (!my\_inode\_cachep)  
 {  
 return =ENOMEM;  
 }  
 ret = register\_fillsystem (&my\_fs\_type);  
 if (likely (ret == 0))  
 printk(KERN\_INFO “success\_n”);  
 else  
 printk(KERN\_ERR “Failed.Err = %d\n”, ret);  
 return ret;  
}

Просьба для глубоко интересующихся программированием – сделать кеш для лабы.

Static void \_exit my\_cleanup(void)  
{  
 int ret;  
 ret = unregister\_filesystem (&my\_fs\_type);  
 kmem\_cache\_destroy(my\_cache)

//дописать у кого-нибудь

Static struct dentry\* gfsmount (struct fill\_system\_type \*fst, int flags, const thas \*dev\_name, void \*data)  
{  
 struct dentry \*ret = mount\_nodev (fst, flags, data, gfsfillsuper);  
 if (IS\_ERR (ret))  
 …  
 else  
 …  
 return ret;  
}

Static int gfsfillsuper (struct super\_block \*sb, void \*data, int silent)  
{  
 sb -> s\_blocksize = PAGE\_SIZE;  
 sb -> s\_blocksize\_bits = PAGE\_SHIRT;  
 sb -> s\_op = gfssbop;  
 sb -> s\_magic = GFS\_MAGIC\_NUMB; //магическое число, любое, хоть 1 или 2  
 struct inode \*rootinode = gfsgetinode (sb, NULL, S\_IFDIR, NULL);  
 if (!rootinode)  
 {  
 …  
 return -1;  
 }  
 struct dentry \*rootdentry = d\_make\_root (rootinode);  
 if (!rootdentry)  
 {  
 …  
 return -1;  
 }  
 sb -> s\_root = rootdentry;  
 return 0;  
}