

## Лекция 5. Виртуальная файловая система.

Виртуальная файловая система<sup>1</sup> это абстракция, позволяющая представить многообразие различных файловых систем - располагающихся, возможно, на различных запоминающих устройствах, разделах или даже на различных компьютерах в сети, - в виде единой иерархии каталогов и файлов с унифицированным программным интерфейсом работы с файлами.

### Стандарт FHS

Официального единого стандарта для файловой системы UNIX не существует. Однако, основные элементы иерархии оказываются одинаковыми в различных версиях ОС. В Linux существует стандарт иерархии файловой системы FHS<sup>2</sup>. Файлы в FHS распределяются по двум независимым категориям:

- с совместным доступом (sharable) или без него (unsharable) и
- изменяющиеся (variable) или статичные (static)

Стандарт определяет следующие элементы иерархии.

1. Корневая файловая система “/” - предназначена для загрузки и восстановления системы, содержит следующие директории
  - /bin основные команды ОС
  - /boot статичные файлы загрузчика ОС
  - /dev файлы устройств
  - /etc конфигурационные файлы хоста
  - /lib разделяемые библиотеки и модули ядра
  - /media для монтирования флоппи, cd, usb и т.п. носителей
  - /mnt для временного монтирования файловых систем
  - /opt дополнительно установленные пакеты приложений
  - /sbin основные системные программы
  - /srv данные для сервисов, исполняющихся в системе
  - /tmp временные файлы
  - /usr иерархия системных ресурсов (Unix System Resources)
  - /var различные изменяющиеся файлыКроме того, возможно создание директория для суперпользователя /root, директория /home – использующегося в качестве точки монтирования домашних каталогов пользователей, а также дополнительных библиотек /lib<nnn>. Никакие приложения не должны создавать специальные файлы и директории в корневой файловой системе.
2. Иерархия /usr является совместно используемой, но доступной только для чтения секцией иерархии, не содержащей специфичной для данного хоста информации. Следующие директории являются обязательными
  - /usr/bin основные команды пользователя (включая компиляторы и интерпретаторы)

1 VFS – virtual file system

2 FHS – filesystem hierarchy standard <http://www.pathname.com/fhs/>

/usr/include    include файлы языка C  
/usr/lib       библиотеки языков программирования  
/usr/local     локально установленные программы  
/usr/sbin      дополнительные системные команды и утилиты.  
/usr/share     статичные данные не зависящие от архитектуры, например man страницы  
Кроме этого, /usr иерархия может содержать средства графического интерфейса  
/usr/X11R6, игровые приложения /usr/games, дополнительные библиотеки /usr/lib<nnn> и  
исходные коды /usr/src ядра и др.

3. Иерархия /var содержит изменяющиеся в процессе функционирования системы файлы. Некоторые из файлов используются только локально, например, /var/log, /var/lock, /var/run. Другие — совместно с другими хостами, например, /var/mail, /var/cache/man, /var/cache/fonts, /var/spool/news.
4. Виртуальная файловая система /proc (Linux) — используется для хранения системной информации и информации об исполняющихся процессах для ядра. Эта файловая система не представляется сохраняемым каталогом на запоминающем устройстве, а существует только в адресном пространстве ядра и строится заново при перезагрузке системы<sup>3</sup>.

### Физическое размещение файлов и структуры данных

Для физического размещения файлов и каталогов на разделах дисков ОС UNIX/Linux могут использовать смешанное разнообразие типов файловых систем: ext2fs, ext3fs, ext4fs, ufs, ffs, reiserfs и множество других, включая не-UNIX файловые системы, такие как, vfat (FAT16/32), ntfs и iso9660. Файлы всех типов файловых систем обрабатываются единообразно, что обеспечивается каркасом виртуальной файловой системы<sup>4</sup> Для определения, какие файловые системы поддерживаются, выполните команду

```
$ cat /proc/filesystems
```

а для определения доступных устройств и файловых систем

```
$ cat /etc/fstab
```

Для представления файловой структуры некоторого устройства в общей иерархии, эта файловая система заданного устройства должна быть смонтирована командой mount/8, т.е. связана с некоторым каталогом в иерархии. Для определения, где и какая файловая система смонтирована, выполните команду

```
$ mount
```

или

---

<sup>3</sup> Аналогично, файловая система /sys в Linux существует только в памяти и используется для отображения информации о драйверах устройств из области ядра в область пользователя.

<sup>4</sup> File System Framework

```
$ cat /proc/mounts
```

Каждый тип файловой системы описывается структурами данных ядра, определенных в заголовочных файлах `/usr/include/linux/*_fs.h`. Каждый файл файловой системы UNIX описывается метаданными в структуре `inode`, являющейся элементом массива `i-list` определенного размера. Собственно блоки с данными файла определяются ссылками в структуре `inode` файла и блоками косвенности.

Для наших целей достаточно рассмотреть упрощенную модель физического распределения блоков раздела диска под файловую систему `unix`



Boot сектор содержит загрузчик ОС, если это устройство или раздел являются загрузочными.

Superblock содержит структуру данных, сохраняющую метаданные о файловой системе в целом. Эта структура данных определена в соответствующем заголовочном файле, например, `/usr/include/linux/ext2_fs.h`, фрагмент которого приводится ниже. Комментарии полей достаточно информативны

```
struct ext2_super_block {
    __le32 s_inodes_count;    /* счетчик i-nodes */
    __le32 s_blocks_count;    /* счетчик блоков */
    __le32 s_r_blocks_count;  /* счетчик зарезервированных блоков */
    __le32 s_free_blocks_count; /* счетчик свободных блоков */
    __le32 s_free_inodes_count; /* счетчик свободных inodes */
    __le32 s_first_data_block; /* первый блок данных */
    __le32 s_log_block_size;  /* размер блока */
    __le32 s_log_frag_size;   /* размер фрагмента */
    __le32 s_blocks_per_group; /* кол-во блоков в группе */
    __le32 s_frags_per_group;  /* кол-во фрагментов в группе */
    __le32 s_inodes_per_group; /* кол-во inodes в группе */
    __le32 s_mtime;           /* время монтирования */
    __le32 s_wtime;           /* время записи */
    __le16 s_mnt_count;       /* счетчик монтирования */
    __le16 s_max_mnt_count;   /* максимальное число монтирований */
    __le16 s_magic;           /* магическое число */
    __le16 s_state;           /* состояние файловой системы */
    __le16 s_errors;          /* поведение при ошибке */
    __le16 s_minor_rev_level; /* нижний уровень проверки */
    ...
}
```

Существует определенная степень взаимозависимости некоторых полей, что позволяет легко проверять целостность файловой системы. При монтировании файловой системы командой `mount/8`, суперблок копируется в кэш и область ядра. При работе с файловой системой информация в суперблоке изменяется и должна быть синхронизирована между различными образами суперблока — в памяти ядра, в кэше и на диске. Для повышения устойчивости в современных файловых системах сохраняются несколько копий суперблока в различных местах физического раздела, которые также должны быть синхронизированы.

`i-list` это массив определенного размера `s_inodes_count + s_free_inodes_count` из элементов структур типа `inode`, определенных в том же самом заголовочном файле, например, в `/usr/include/linux/ext2_fs.h`, представленный следующим фрагментом

```
struct ext2_inode {
    __le16 i_mode;           /* режим доступа к файлу */
    __le16 i_uid;            /* идентификатор владельца UID */
    __le32 i_size;           /* размер файла в байтах */
    __le32 i_atime;          /* время последнего доступа */
    __le32 i_ctime;          /* время создания */
    __le32 i_mtime;          /* время последней модификации */
    __le32 i_dtime;          /* время удаления */
    __le16 i_gid;            /* идентификатор группы владельца GUID */
    __le16 i_links_count;    /* счетчик жестких ссылок */
    __le32 i_blocks;         /* счетчик блоков */
    __le32 i_flags;          /* контрольные флаги файла */
    ...
    __le32 i_block[EXT2_N_BLOCKS]; /* указатели на блоки с данными */
    ...
};
```

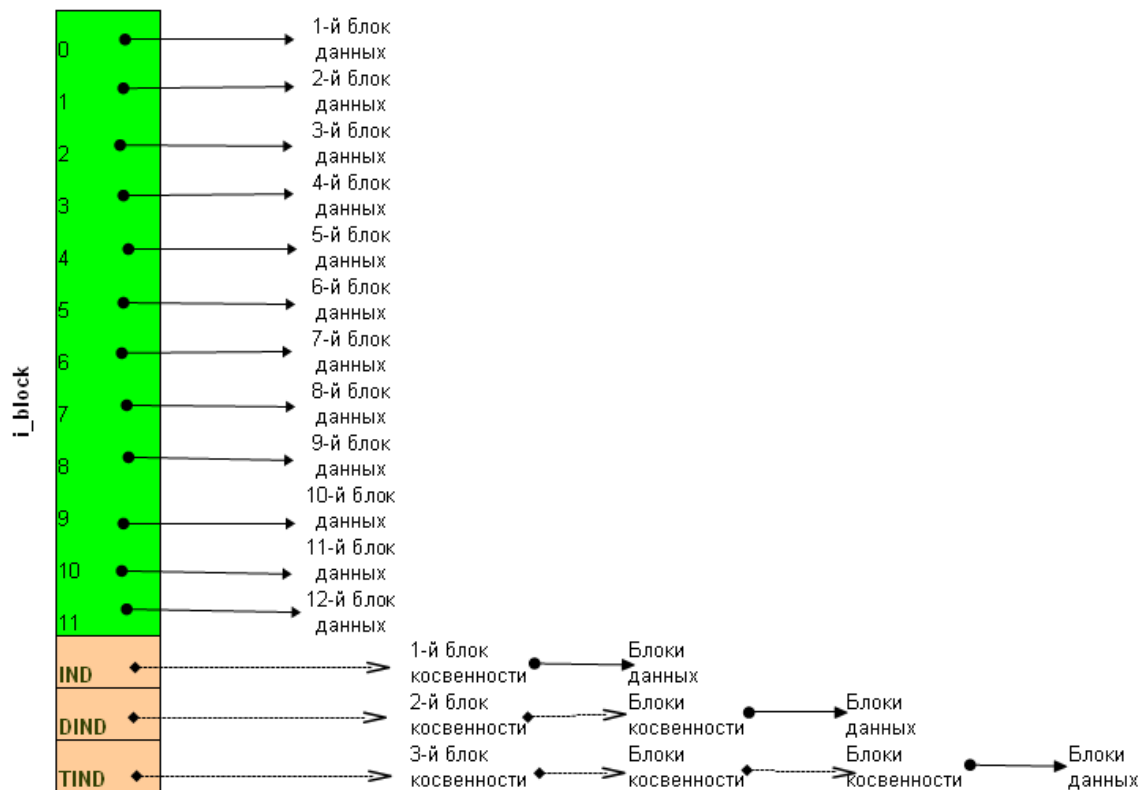
При создании нового файла, создается соответствующая структура `inode`, которая сохраняется в первом свободном элементе массива `i-list` на диске. Обратите внимание, что имя файла не включается в структуру `inode`, вместо этого, структура данных для директория связывает имя файла с индексом `inode` в `i-list`, который называется индексным дескриптором файла

```
struct ext2_dir_entry_2 {
    __le32 inode;            /* индексный дескриптор inode */
    __le16 rec_len;          /* размер записи в директории */
    __u8 name_len;           /* длина имени */
    __u8 file_type;
    char name[EXT2_NAME_LEN]; /* имя файла */
};
```

Каждый файл имеет владельца, идентификатор его группы пользователей, и использует определенное количество блоков диска для хранения данных, доступ к которым предоставляется через массив указателей `i_blocks` из `EXT2_N_BLOCKS` элементов. Элементы `i_blocks` с 0-го по

11-й содержат прямые ссылки на первые 12 блоков с данными, Другие элементы содержат указатели на блоки косвенности, как показано на рисунке ниже.

Таким образом, inode является основной структурой описывающей файл и размещение данных. Имя файла используется только для поиска соответствующего ему индексного дескриптора в директории.



Определить индексные дескрипторы файлов в текущем каталоге можно командой

```
$ ls -li
```

Полная информация о файле, выводимая командой

```
$ ls -li файл
-rwxr-xr-- 1 root root 193 Feb 13 10:55 файл
позиц. 12345678910 11 12 13 14 15 16
```

означает следующее:

- 1-я позиция — тип файла: «-» - регулярный файл, или возможные значения: d, l, p, c, b, s
- 2-10 позиции — права доступа к файлу по чтению «r», записи «w» и выполнению «x», для владельца, членов группы владельца и для всех остальных - 3 группы символов gwx, соответственно. Знак «-» вместо буквы означает запрет операции для соответствующей

категории пользователей

- позиция 11 — количество жестких ссылок на файл
- позиция 12 — имя владельца файла
- позиция 13 — имя группы владельца файла
- позиция 14 — размер файла в байтах
- позиция 15 — дата и время последнего доступа к файлу
- позиция 16 — имя файла

## Права доступа к файлу

Права доступа `gwx` удобно представить в виде восьмеричного числа:

“r”	= 4
“w”	= 2
“x”	= 1
“-”	= 0

так что, сумма числовых значений `r+w+x` определяет числовое значение прав доступа к файлу для данной категории пользователей (владелец=`u`, группа=`g`, другие=`o`); например, `0765` означает `u=gwx=7`, `g=rw=6` `o=rx=5`.

При создании файла, права доступа автоматически назначаются с учетом установленной файловой маски, которая отображается или может быть изменена командой `umask.`, например,

```
$ umask
0022
```

или

```
$ umask -S
u=rwx, g=rx, o=rx
```

При этом, при создании каталогов начальные значения прав доступа вычисляются

права для каталога = `0777 — umask`

а для всех остальных файлов, при их создании назначаются права

права для файла = `0666 — umask`

Изменение прав доступа к файлу осуществляется командой

```
$ chmod режим файл
```

где «режим» доступа может задаваться числовым или символьным способом. Для символьного задания используются обозначения u, g, o, как выше, и a — для всех категорий пользователей одновременно, и + «разрешить» и - «запретить» для указанных категорий пользователей указанные операции. Например, команда

```
$ chmod a+x файл
```

делает указанный файл доступным для выполнения для всех категорий пользователей.

### **Жесткие и символические ссылки**

Ссылки на файл создаются командой ln. Жесткая ссылка (hard link) не является специальным файлом, а представляет из себя только новую запись в файле-каталоге, связывающую существующий индексный дескриптор с новым именем. Например, создадим новый файл

```
$ touch new.file
```

команда

```
$ ls -il new.file
```

показывает индексный дескриптор и количество жестких ссылок (=1) на созданный файл (размером 0 байт). Теперь создадим жесткую ссылку на файл

```
$ ln new.file new.hard
```

и проверим, что имя «new.hard» является ссылкой на тот же индексный дескриптор, что и файл «new.file»

```
$ ls -li new.*
```

При удалении любой жесткой ссылки, удаляется только запись в директории, до тех пор, пока счетчик жестких ссылок не окажется равным 1. Только удаление последнего имени приведет к физическому удалению файла — освобождению индексного дескриптора для нового inode и потери ссылок на блоки диска с данными.

Команда ln не позволяет создавать жесткие ссылки на директории, хотя каждый новый директорию при создании уже имеет две жестких ссылки — имя, заданное для директория, например, в команде mkdir, и относительное имя текущего директория «.». При создании подкаталогов в данной директории счетчик жестких ссылок на текущий директорию автоматически увеличивается каждый раз на 1, т.к. в каждом новом подкаталоге появляется ссылка на текущий (родительский) директорию с относительным именем «..». Кроме того, создание жестких ссылок для любых файлов ограничено текущей файловой системой.

Ключ -s в команде ln позволяет создавать специальные файлы — симлинки, или символические

ссылки, которые резервируют для себя новый индексный дескриптор и сохраняют адресную ссылку на исходный файл, например,

```
$ ln -s файл симлинк
```

Символическая ссылка отображается с символом l в первой позиции и ее размер равен 4 байта в выводе команды

```
$ ls -l симлинк  
lrwx----- 1 user group 4 Feb 13 10:55 симлинк → файл
```

Удаление симлинка приводит к удалению файла ссылки, но не исходного файла. Если удален исходный файл, то симлинк оказывается нарушенным, т.к. ссылается на несуществующий файл. Кроме того, следует быть осторожным при создании симлинка при задании пути к исходному файлу. Относительное имя файла позволяет только одновременное перемещение исходного файла и симлинка на него. Абсолютный путь к файлу позволяет перемещать только симлинк без изменения расположения исходного файла. Символические ссылки могут пересекать границы файловых систем.

### Рекомендуемая литература

1. Р. Лав. Linux Системное программирование. Спб., Питер, 2008, 416 с.
2. Б. Моли. UNIX/Linux Теория и практика программирования. М., КУДИЦ-ОБРАЗ, 2004, 576 с.

### и ссылки

1. <http://www.ibm.com/developerworks/ru/library/l-linux-filesystem/>
2. <http://fuse.sourceforge.net/>
3. <http://www.ibm.com/developerworks/ru/library/l-fuse/index.html>