Московский Физико-Технический Институт
Физтех-школа Аэрокосмических технологий
Институт Аэромеханики и Летательной Техники

МФТИ

ФАКТ

Архитектура Компьютера и Операционные Системы. Часть 2. Основы операционных систем

Лекция 5: Файловая система Unix. ч1

Новиков Андрей Валерьевич
д.ф.-м.н.

Жуковский

Логические разделы физического носителя информации

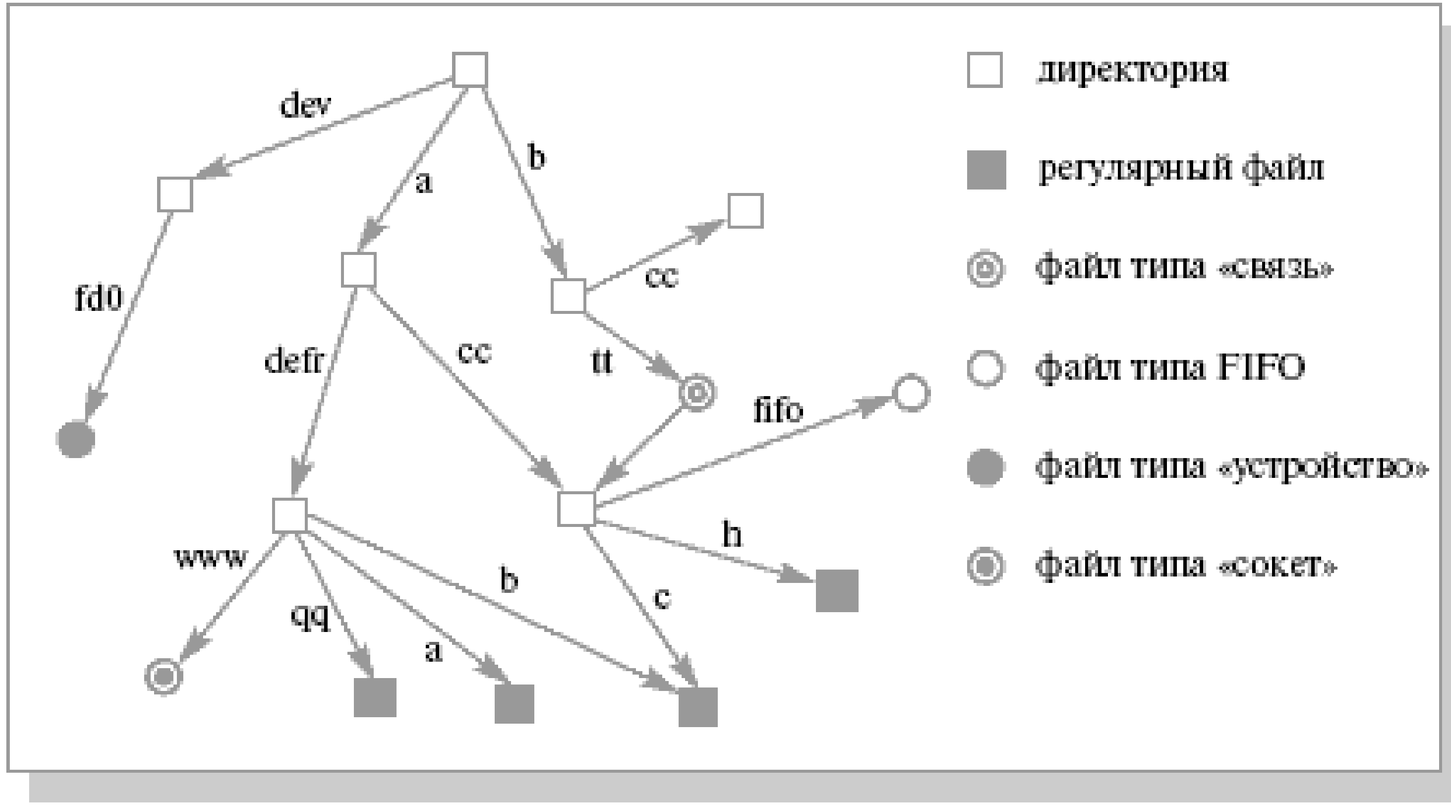
- Физический носитель информации (диск, лента) в ОС представляется в виде **partitions** – т.н. разделов или логических дисков
- Физический диск делится на несколько разделов, но и несколько дисков можно объединить в один раздел (raid-массив)
- Разделы упорядочены линейно (без вложенностей и т.п.)
- В рамках каждого раздела – своя файловая система
- Применяемые структуры разделов на диске
  - MBR (Master Boot Record) – традиционная, 4 раздела максимум, 2Тб на раздел
  - GPT (GUID Partition Table) – современная (2000+)

Необходимость partitions

- Несколько операционных систем на одном физическом диске
- Несколько видов файловых систем
- Размещение на разделах различных категорий файлов. Например, системные файлы – на одном разделе, на другом – пользовательские.
- Ограничение ОС, например на размер диска

Структура файловой системы

- *Ациклический граф* с однонаправленными ребрами
- **имя файла** связывается не с узлом, соответствующим файлу, а с входящим в него **ребром**.
- Ребра, выходящие из узлов, соответствующих файлам типа "связь", являются неименованными.
- В узел "директория" не может входить (обычно) более одного именованного ребра
- Полное имя файла -- имя, получающееся при прохождении по ребрам от корневого узла до узла файла по любому пути, где имена рёбер разделяются символом «/»

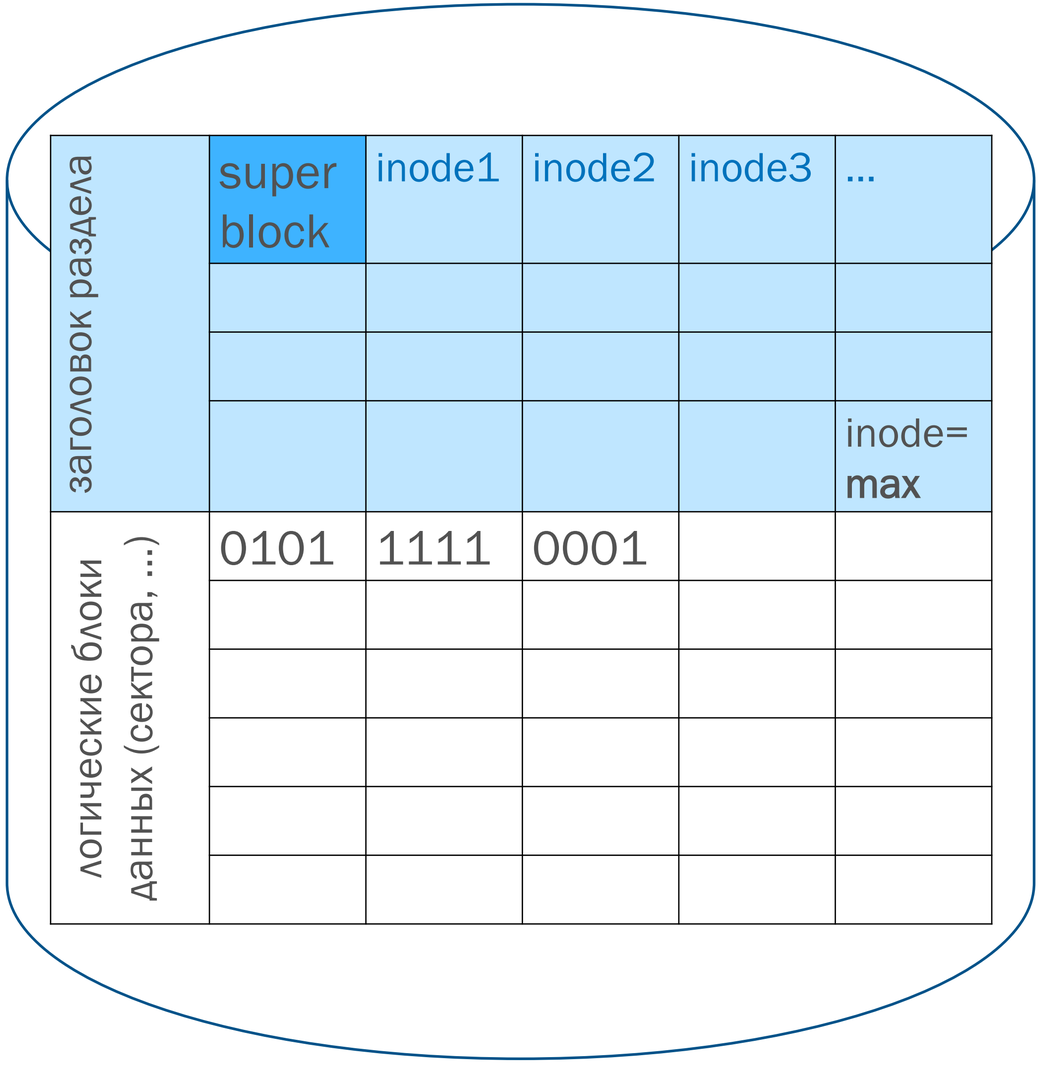


Типы файловых систем

- Extended File System (linux): ext2 (1993+), ext3 (2001+), ext4 (2008+)
- xfs (Silicon Graphics, 1994)
- zfs (Zettabyte File System), Sun, 2005 (поддержка в Linux ограничена)
- ReiserFS (2001), Reiser4 (2004)
- btrfs (B-tree FS, «Better FS»), Oracle, 2008
- ...

Файловая система s5fs

- Partition делится на:
  - **заголовок раздела**: служебная информация для работы ФС: массив индексных узлов (inodes) с атрибутами файлов, размер *фиксирован* (65535 шт)
  - **логические блоки**: содержательная часть файлов.
- Логический блок = **кластер**
  - минимальная адресуемая часть
  - 512 Б – 64киБ, обычно 4кБ
  - формируется из физических **секторов** диска, 512 или 4094 байт

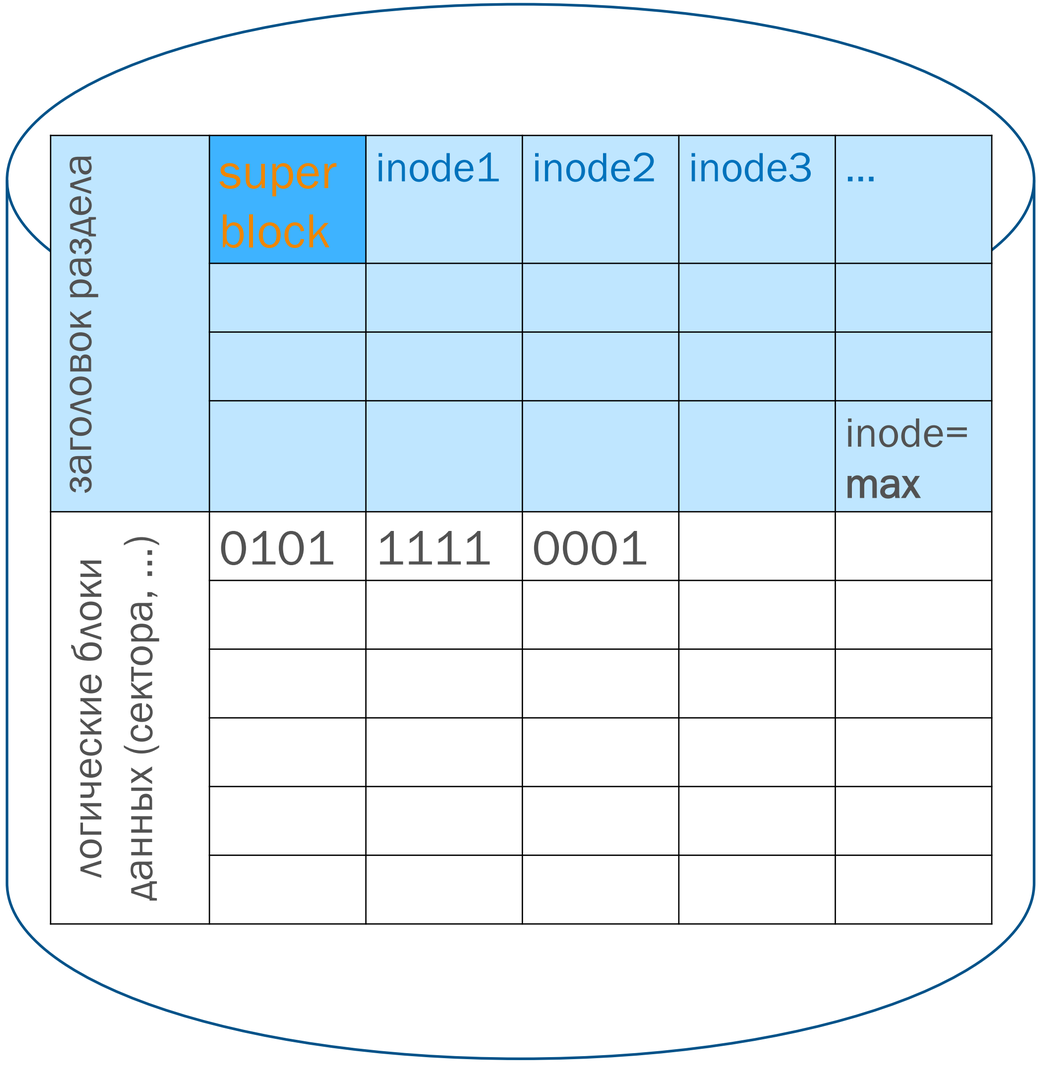


Файловая система s5fs

- **Суперблок**

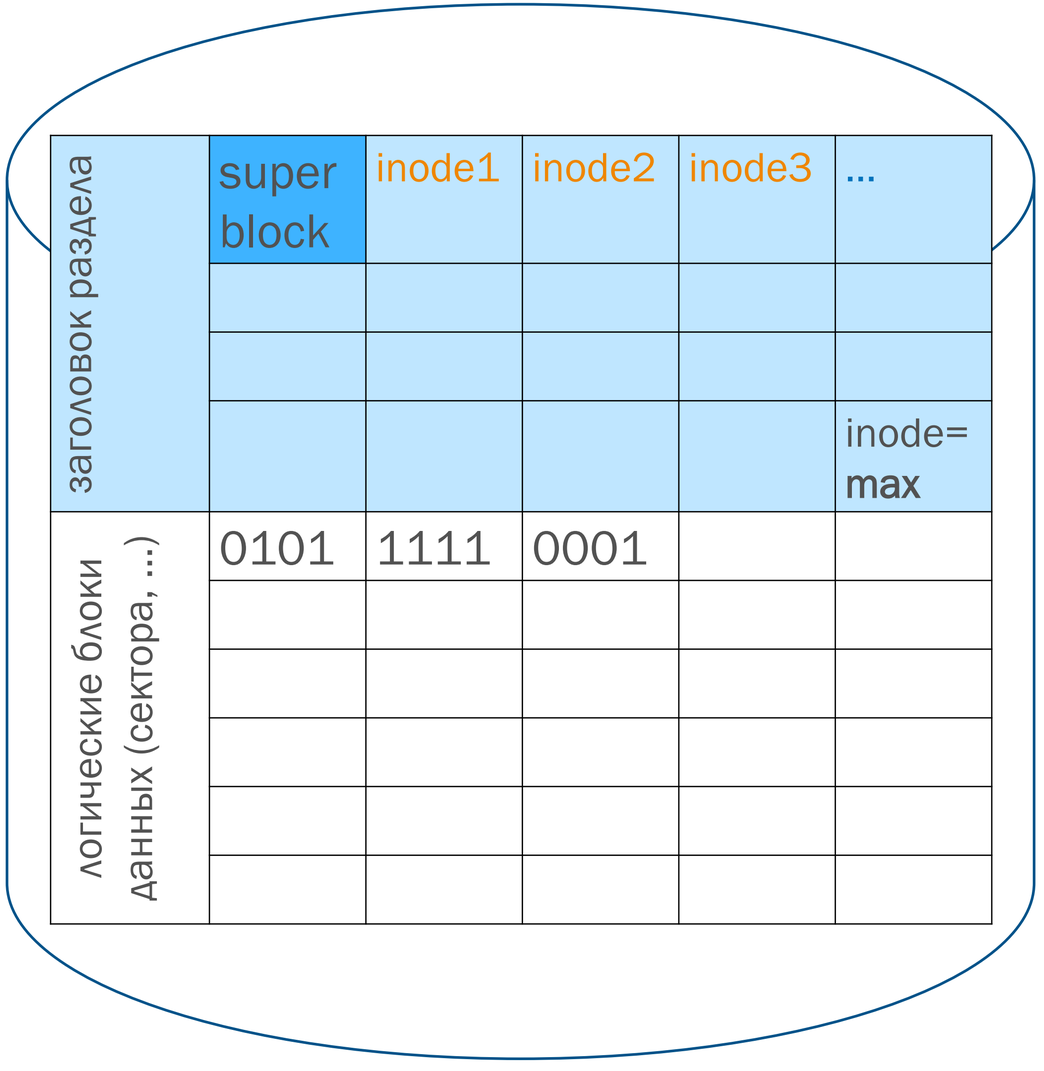
  информация о файловой системы в целом:

  - Тип файловой системы.
  - Флаги состояния файловой системы.
  - Размер логического блока в байтах (обычно кратен 512 байтам).
  - Размер файловой системы в логических блоках (включая сам суперблок и массив inode).
  - Размер массива индексных узлов (т.е. сколько файлов может быть размещено в файловой системе).
  - Число свободных индексных узлов (сколько файлов еще можно создать).
  - Число свободных блоков для размещения данных.
  - Часть списка свободных индексных узлов.
  - Часть списка свободных блоков для размещения данных.



Файловая система s5fs

- Индексный узел - index node – **inode** содержит *атрибуты файла*:
  - Тип файла и права различных категорий пользователей для доступа к нему.
  - Идентификаторы владельца-пользователя и владельца-группы.
  - Размер файла в байтах (только для регулярных файлов, директорий и файлов типа "связь").
  - Время последнего доступа к файлу.
  - Время последней модификации файла.
  - Время последней модификации самого индексного узла.
  - Количество связей у файла
  - **Координаты логических блоков.**
- **Один inode ↔ один объект ФС**



Организация каталогов

- Объект файловой системы типа **directory** («директория», каталог):
  - имена файлов, лежащих непосредственно в каталоге + соответствующие номера индексных узлов

  т.е.
  - имена ребер, выходящих из узла каталога вместе с индексными номерами узлов, к которым они ведут.
- **.** — сам каталог
- **..** — каталог на уровень выше

Каталог		
.	inode=7777	другая инф.
..	inode=8888	
имя_файла1	inode=**1234**	
имя_файла2	inode=5678	
abc	inode=2222	
def	inode=5555	
…	…	
filename	inode=**1234**	

Организация каталогов

- Имя файла в каталоге:
  - фиксированный размер 14 байт в ФС f5fs
  - переменный размер до 255 байт в современных ФС

- В зависимости от ФС в каталоге **может** дополнительно храниться длина записи, длина имени файла

Каталог		
.	inode=7777	другая инф.
..	inode=8888	
имя_файла1	inode=**1234**	
имя_файла2	inode=5678	
abc	inode=2222	
def	inode=5555	
…	…	
filename	inode=**1234**	

Чтение содержимого каталога

```
#include <sys/types.h>
#include <dirent.h>


DIR* opendir(char* name);
```

Открывает каталог на чтение, как поток информации

- Возвращаемое значение
  указатель на хэндл каталога,
  NULL – ошибка, код ошибки в **errno**
- **name** –имя каталога

Чтение содержимого каталога

```
#include <sys/types.h>
#include <dirent.h>

struct dirent* readdir(DIR* dir);

struct dirent {
    char d_name[ ];
    …
};
```

Читает **очередную** запись из каталога

- Возвращаемое значение
  данные о записи каталога,
  NULL – конец каталога

Чтение содержимого каталога

```
#include <sys/types.h>
#include <dirent.h>


void rewinddir(DIR* dir);
```

Позиционирует потока информации для директории на первой записи (начале) директории.

- **dir** – указатель на открытый каталог

Чтение содержимого каталога

```
#include <sys/types.h>
#include <dirent.h>


int closedir(DIR* dir);
```

Закрытие потока информации для директории, очистка внутренних структур.

- Возвращает
  0 – успех, -1 – ошибка
- **dir** – указатель на открытый каталог

Создание жёсткой связи

```
$ ln source dest
$ ln src1 src2 .. dest_dir/
```

- добавляет в каталог запись, указывающую на тот же самый inode исходного файла (файлов)
- счётчик связей в индексном узле увеличивается
- объект ФС получает дополнительное имя, **имена равнозначны**
- невозможно создание жёсткой связи между различными ФС
- в UNIX (несмотря на стандарт POSIX) запрещено создание жестких связей к каталогам (предотвращаются бесконечные циклы)
- в Linux запрещено создание жестких связей к специальным файлам устройств

Создание жёсткой связи

```
#include <unistd.h>


int link(char* pathname,
         char* linkpathname);
```

Создаёт жёсткую связь

- Возвращаемое значение
  0 – успех, -1 – ошибка, код ошибки в **errno**
- pathname –исходное имя файла
- linkpathname – имя создаваемой связи

Создание «мягких» связей

```
$ ln –s source dest
$ ln –s src1 src2 .. dest_dir/
```

Создаёт «мягкие связи»:

- ❑ soft link, symbolic link, мягкая ссылка
- ❑ создаётся специальный файл типа «связь»
- ❑ различаются оригинал и файла-связь
- ❑ возможны ссылки между различными ФС

Создание мягкой связи

```
#include <unistd.h>


int symlink(
   char* pathname,
   char* linkpathname
);
```

Создаёт мягкую связь

- Возвращаемое значение
  0 – успех, -1 – ошибка, код ошибки в **errno**
- pathname –исходное имя файла
- linkpathname – имя создаваемой связи

Удаление объектов ФС

```
#include <unistd.h>


int unlink(char *pathname);
```

Удаляет имя объекта ФС, счётчик ссылок файла уменьшается

- Возвращаемое значение
  0 – успех, -1 – ошибка, код ошибки в **errno**
- pathname –исходное имя файла
- Если счётчик ссылок =0, то:
  - Если нет процессов, которые открыли файл, то он полностью удаляется (если есть процессы открывшие файл, то продолжает существовать)
  - Если имя относится к файлу типа socket, FIFO или к специальному файлу устройства, то файл удаляется независимо от наличия процессов, держащих его открытым, но процессы, открывшие данный объект, могут продолжать пользоваться им.
  - Если имя относится к файлу типа "связь", то он удаляется, и мягкая связь оказывается разорванной.

Чтение атрибутов файла

```
#include <sys/stat.h>
#include <unistd.h>

int stat(char *filename,
   struct stat *buf);
int fstat(int fd,
   struct stat *buf);
int lstat(char *filename,
   struct stat *buf);
```

Читает информацию об атрибутах файла, информацию из индексного узла

- Возвращаемое значение
  0 – успех, -1 – ошибка, код ошибки в **errno**
- **filename** –имя файла
- **fd –** файловый дескриптор открытого файла (для fstat)
- **buf** – указатель на структуру, в которую помещается информация

Чтение атрибутов файла

```
struct stat {
    dev_t st_dev; /* устройство, на котором расположен файл */
    ino_t st_ino; /* номер индексного узла для файла */
    mode_t st_mode; /* тип файла и права доступа к нему */
    nlink_t st_nlink; /* счетчик числа жестких связей */
    uid_t st_uid; /* идентификатор пользователя владельца */
    gid_t st_gid; /* идентификатор группы владельца */
    dev_t st_rdev; /*тип устройства для специальных файлов устройств*/
    off_t st_size; /* размер файла в байтах (если определен для данного типа файлов) */
    unsigned long st_blksize; /* размер блока для файловой системы */
    unsigned long st_blocks;  /* число выделенных блоков */
    time_t st_atime, st_mtime, st_ctime; /* время последнего доступа, последней модификации, создания*/
}
```

Тип файла

```
#include <sys/stat.h>
#include <unistd.h>


struct stat {
    …
    mode_t st_mode;
    …
};
```

Макросы для извлечения типа из поля `st_mode`

mode_t m = buf.st_mode;

- S_ISLNK(m) – файл типа "связь"?
- S_ISREG(m) – *регулярный файл*?
- S_ISDIR(m) – директория?
- S_ISCHR(m) – специальный файл *символьного устройства*?
- S_ISBLK(m) – специальный файл *блочного устройства*?
- S_ISFIFO(m) – *файл типа FIFO*?
- S_ISSOCK(m) – файл типа "socket"?