Файлы

Что такое файл?

Что такое файл?

- Сущность, которая содержит данные и имеет имя

Everything is a file!

Имя файла

- Не более `PATH_MAX` символов: 4 Кб на современных ОС, 256 байт для portability
- `PATH_MAX` включает `\0` в конце
- Разделитель пути – `/`
- Части пути не более 255 символов каждая

Имя файла

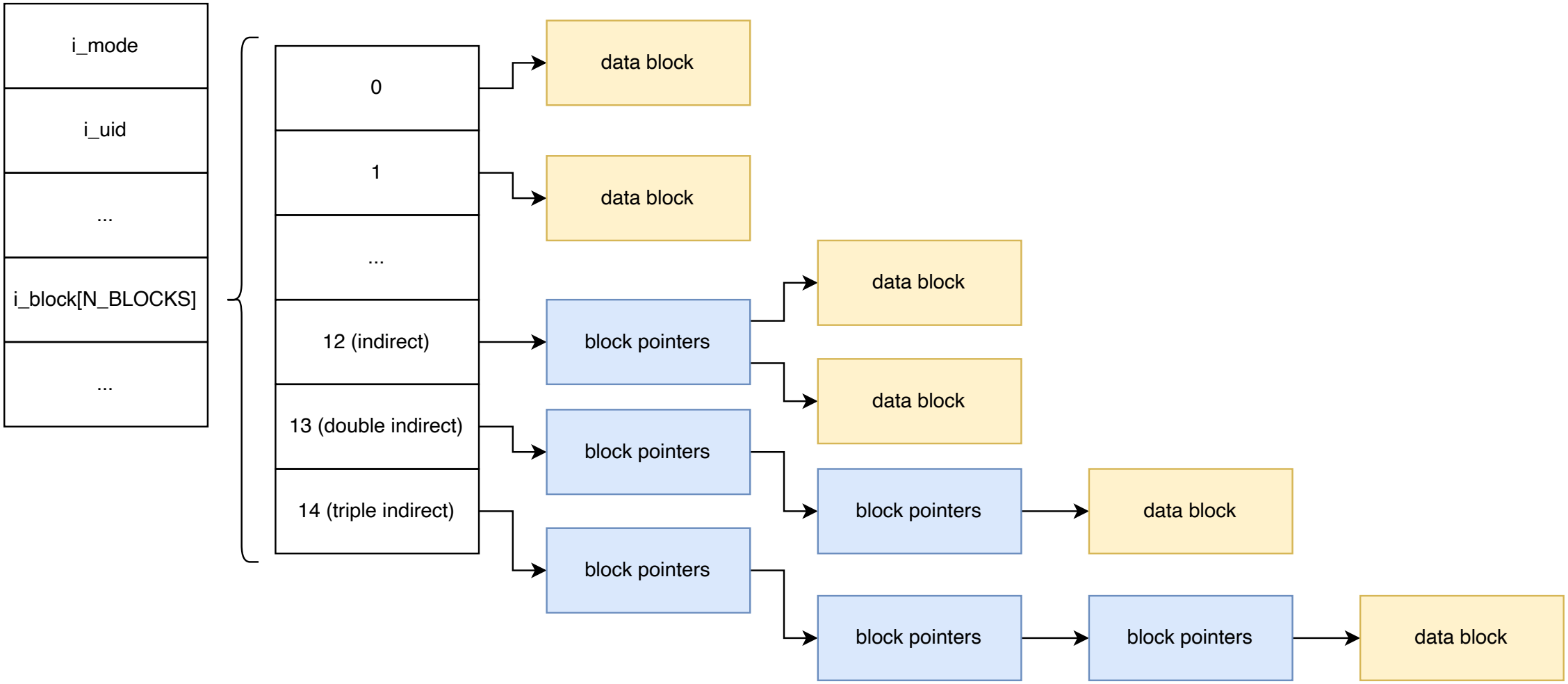
- Абсолютный путь: начинается с корня (например, `/Users/carzil/mipt`)
- Относительный путь: вычисляется от текущей директории (например, `carzil/mipt`)
- `.` – текущая директория (`./carzil/mipt` = `carzil/mipt` и `./carzil/./././mipt` = `./carzil/mipt`)
- `..` – директория выше (`/Users/carzil/mipt/..` = `/Users/carzil`)

Файловая система

- Структура данных для организации хранения информации
- Метаданные – информация о файле: дата последнего изменения, права доступа, создатель итд
- Работают поверх хранилища (HDD, SSD, NVMe)
- Хранилище традиционно разбивается на *блоки*
- Размер блоков обычно 512 байт или 4 Кб

ext2

- Linux, 1993 год
- inode – физическое представление файла на диске: заголовок с метаинформацией + информация где он хранится
- Директории тоже хранятся в inode (т.к. директория – файл!)


i_mode
i_uid
...
i_block[N_BLOCKS]
...
0
1
...
12 (indirect)
13 (double indirect)
14 (triple indirect)
data block
data block
block pointers
data block
data block
block pointers
block pointers
data block
block pointers
block pointers
data block

ext4

- 2006 год
- Де-факто стандартная файловая система для Linux
- Журналируемая
- Для больших директорий используется HTree

Другие файловые системы

- FAT32
- NTFS
- ReiserFS
- ZFS

sysfs и procfs

- «Метафайловые системы»
- Не имеют никаких данных на диске, возвращают информацию напрямую из ядра Linux
- Часто используются, чтобы не добавлять новые сисколлы

FUSE

- Код файловой системы обычно расположен в ядре – это неудобно
- FUSE = file system in userspace

Файловые дескрипторы

- Преобразование имени файла в inode – очень дорогая операция: может требовать много обращений к диску
- Этот процесс «кэшируют» с помощью файловых дескрипторов
- Файловый дескриптор – число от 0
- Новый файловый дескриптор будет минимальным доступным числом

Файловые дескрипторы

- За каждым файловым скрывается специальная структура в ядре
- Указатель на inode, позиция в файле, флаги (чтение/запись/блокирование), различные локи, итд

Работа с данными файла

```c
#include <unistd.h>

int open(const char *pathname, int flags, mode_t mode);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int close(int fd);
```

Работа с метаданными файла

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

int stat(const char* path, struct stat* buf);
int fstat(int fd, struct stat *statbuf);
int lstat(const char* pathname, struct stat* statbuf);

struct stat {
    dev_t     st_dev;
    ino_t     st_ino;
    mode_t    st_mode;
    nlink_t   st_nlink;
    uid_t     st_uid;
    gid_t     st_gid;
    dev_t     st_rdev;
    off_t     st_size;
    blksize_t st_blksize;
    blkcnt_t  st_blocks;
    struct timespec st_atime/st_mtime/st_ctime;
};
```

Права доступа

- rwx = **R**ead/**W**rite/e**X**ecute
- 9 бит, 3 группы: права владельца, права группы и права для остальных
- Часто записываются как числа в восьмиричной системе счисления
- $777_8 = 111111111_2$ = `rwxrwxrwx`
- $644_8 = 110100100_2$ = `rw-r--r--`

Права доступа для директорий

- r – листинг директории
- w – создание файлов внутри директории
- x – возможность перейти в директорию (cd), а также доступ к файлам

Регулярные файлы

- `S_ISREG(stat.st_mode)`
- Обычные файлы с данными

Директории

- `S_ISDIR(stat.st_mode)`
- Специальный API для чтения, обычные read/write не работают
- Создание удаление: mkdir/rmdir

Директории

```c
#include <dirent.h>

struct dirent *readdir(DIR *dirp);

struct dirent {
    ino_t          d_ino;
    off_t          d_off;
    unsigned short d_reclen;
    unsigned char  d_type;
    char           d_name[256];
};

int closedir(DIR *dirp);
```

Символические ссылки

- `S_ISLNK(stat.st_mode)`
- Аналог `std::weak_ptr` для inode
- Могут быть dangling: то есть ссылаться на файл, которого нет
- Отдельный тип файла
- Путь на который она ссылается записан в блоках

Жёсткие ссылки

- Аналог `std::shared_ptr` для inode
- Только внутри одной файловой системы
- Если количество жёстких ссылок стало равно 0, то inode становится свободной
- Не файл, а сущность файловой системы

Символьные устройства (character device)

- `S_ISCHR(stat.st_mode)`
- Устройства, из которых можно последовательно читать
- Клавиатура, звуковая карта, сетевая карта
- Такие файлы создаются драйверами ядра

Блочные устройства (block device)

- `S_ISBLK(stat.st_mode)`
- Разбиты на блоки одинакового размера
- Можно прочитать любой блок
- HDD, SSD, NAS

Вопросы?