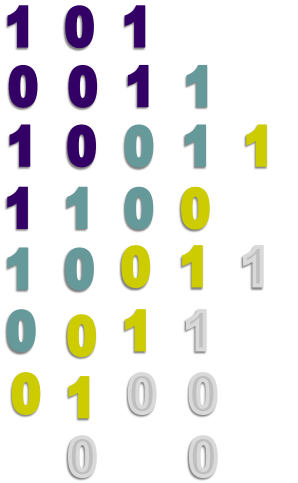
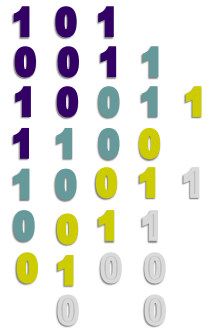
Операционные системы

Хранение данных.

Файловые системы.

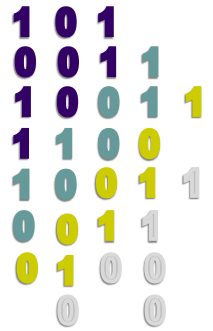


Хранение данных



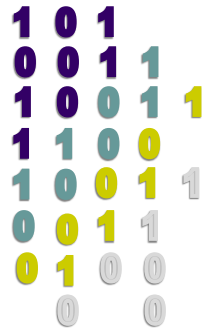
- Оперативная память не подходит для долговременного хранения больших объёмов данных
- Устройства внешней (вторичной) памяти, большой ёмкости
  - Для хранения данных на них необходимы специальные структуры и алгоритмы

История



- До появления жёстких дисков
  - Магнитные ленты
  - Программы сами выбирали способ хранения данных на носителе
  - У каждой программы свои структуры данных
    - Способы размещения и именования
  - Сложно поддерживать данные и архивы
- Централизованные системы управления файлами!

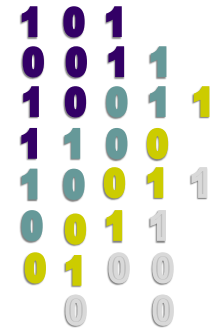
Файловая система



- Компонента операционной системы, организующая
  - эффективную работу с данными,
    - хранящимися во внешней памяти
  - и обеспечивающая пользователю интерфейс для работы с этими данными

- Чтобы избавить пользователя от сложностей взаимодействия с аппаратурой

Основные идеи построения



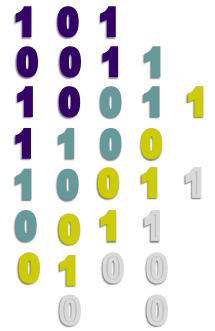
- Внешняя память разделена ОС на блоки фиксированного размера (4096 байт)
- Файл — набор блоков (несмежных)
- Адреса блоков данных файла хранятся в отдельном блоке внешней памяти
  - В так называемом индексе или индексном узле
  - Индекс файла — список элементов, с номером блока в файле и его положением
  - По номеру байта вычисляют блок => его адрес

Индексация

9000	9001	9002	9003	9004	…	…			

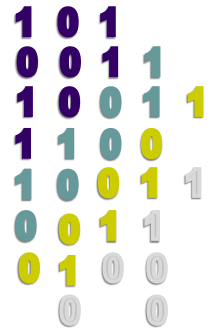
Номер блока	Положение	Атрибуты
0	100000	
1	17	
2	6000	

Иерархия каталогов



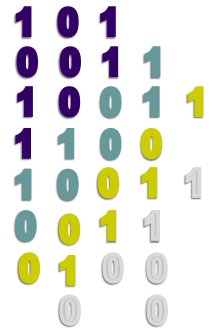
- ФС представляет собой иерархическую структуру каталогов
- Путь в иерархии представляет собой уникальное имя объекта в ФС
- Сами каталоги играют роль индексов с ссылками на свои подкаталоги
  - Вся ФС является большим индексным файлом

Файловая система



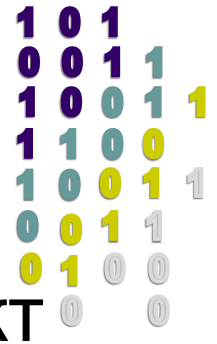
- Включает в себя:
  - совокупность всех файлов на диске,
  - структуры данных для управления файлами
    - Например, каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске
  - комплекс системных программных средств, реализующих управление файлами
    - создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами

Оптимизация ФС



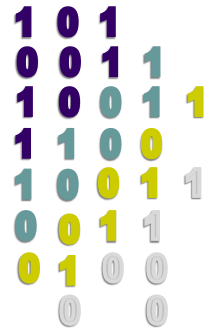
- Алгоритмы работы ФС включают операции с
  - Процессором
  - Памятью
  - Жёстким диском (ЖД)
- Обращение к ЖД в 100000+ раз медленнее чем обращение к памяти
- Оптимизируется с точки зрения количества операций с жёстким диском

Имена файлов



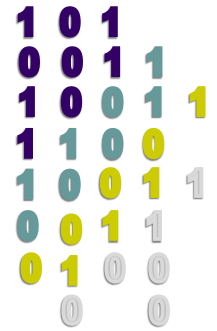
- Файл — абстрактный именованный объект
  - Имя ~ единственный способ доступа к файлу
- Именование
  - Регистронезависимое и регистрозависимое
  - С расширением и без
    - Формат «8.3»
  - Длина имени — 255 символов
  - Символы — любые символы Юникода, кроме NUL
    - NTFS: любые символы Юникода, кроме «NUL», «/», «\», «"», «*», «?», «<», «>», «|», «:»

Типы и атрибуты файлов



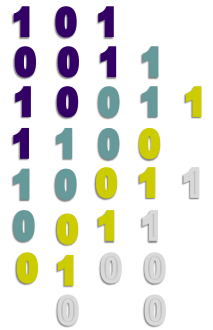
- Директории и обычные файлы
- Обычные файлы: бинарные и текстовые
  - Бинарные имеют сложную структуру
    - Исполняемый Unix файл имеет пять секций: заголовок, текст, данные, биты реаллокации и символьную таблицу
  - Прикладное ПО, распознаёт тип файла по его имени в соответствии с общепринятыми соглашениями
- Атрибуты файла
  - Даты, размер, защита, скрытый или нет и т.д.

Доступ к данным в файле



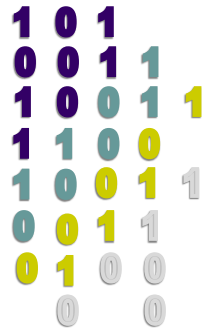
- Последовательный
  - Модель ленты, текущая позиция, **rewind**
  - Может быть эмулирован
- Случайный
  - Random
  - Адресация относительно начала или текущей позиции, **seek**
- Методы основанные на индексации

Операции над файлами



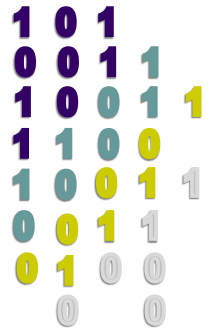
- Create
- Delete
- Open
- Close
- Seek
- Read
- Write
- Get attributes
- Set attributes
- Rename

Директории



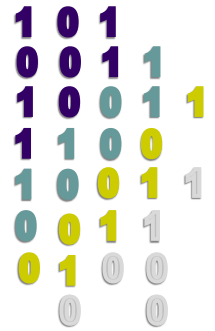
- Большие объёмы данных требуют эффективного управления и структуру
- Многоуровневое именование — каталоги
  - Путь от корня ФС до файла — **полное имя** файла
  - Полное имя (абсолютный путь) уникально!
  - Относительный путь
    - Использует идею рабочего каталога
    - Не уникальны
    - Короче в записи, более удобны в некоторых случаях

Каталоги



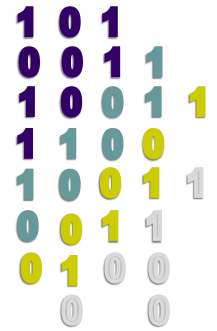
- Относительные каталоги: «.» и «..»
- Корень ФС
  - Одна ФС на одном разделе
    - Пути начинаются с имени устройства
    - C:\temp\games\CS\hl.exe
  - Единое дерево с одним корнем
    - Все части ФС с разных разделов должны быть подключены в общее дерево
    - **mount**
    - /mnt/win/temp/games/CS/hl.exe

Операции над директориями



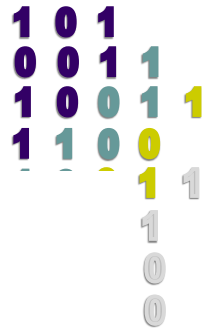
- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Unlink

Контроль доступа к файлам



- Необходимо контролировать выполнение операций над объектами
  - ACL — access control list
  - Кто и что может/не может выполнять
  - Заранее неизвестен список пользователей
  - Запись в директории переменного размера
- Unix Way
  - Owner:Group:Universe
  - ACL фиксированного размера

UNIX Permissions



- Каждому элементу ACL соответствует 3 вида привилегий:
  - r - чтение, w - запись, x – исполнение (e**x**ecute)
  - owner – rwx; group – rwx; universe – rwx
  - rwxrwxrwx
  - rwxr-x---
  - Чтобы хранить эти [illegible] необходимо их закодировать числ[illegible]

9 бит на весь ACL

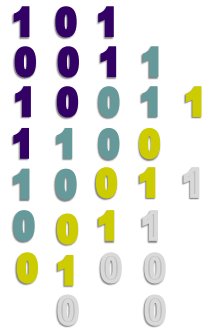
owner			group			other		
r	w	x	r		x			
4	2	1	4	2	1	4	2	1
7			**5**			**0**		

chmod 750 /dir/somefile

chown user /dir/somefile

chmod .group /dir/somefile

Интерфейс ФС



- Создание ФС
  - Форматирование — создание на ЖД структур описывающих элементы ФС
- Операции с директориями
  - Создание структуры каталогов
- Операции с файлами
  - Заполнение готовой иерархии необходимыми файлами

Схема работы ФС

API

Программа

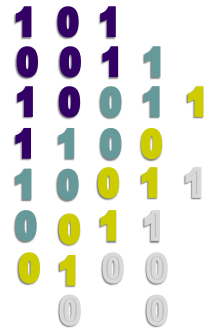
Логическая структура

Базисная подсистема
- Кеширование
- Логические элементы носителя

Система ввода-вывода
- Драйвера устройств, обработчики прерываний

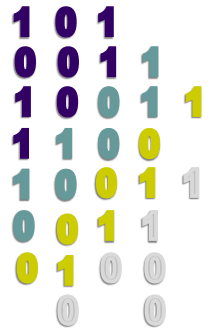
Оборудование

Схема работы ФС



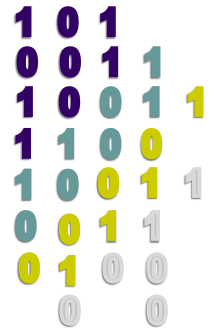
- Нижний уровень
  - Размещение и адресация байтов на постоянных носителях
  - LBA vs CHS (cylinder, head, sector)
- Система ввода-вывода
  - Работает с физическими блоками
  - Обрабатывает прерывания и т.д.
  - Может предоставлять логический адрес блока

Схема работы ФС



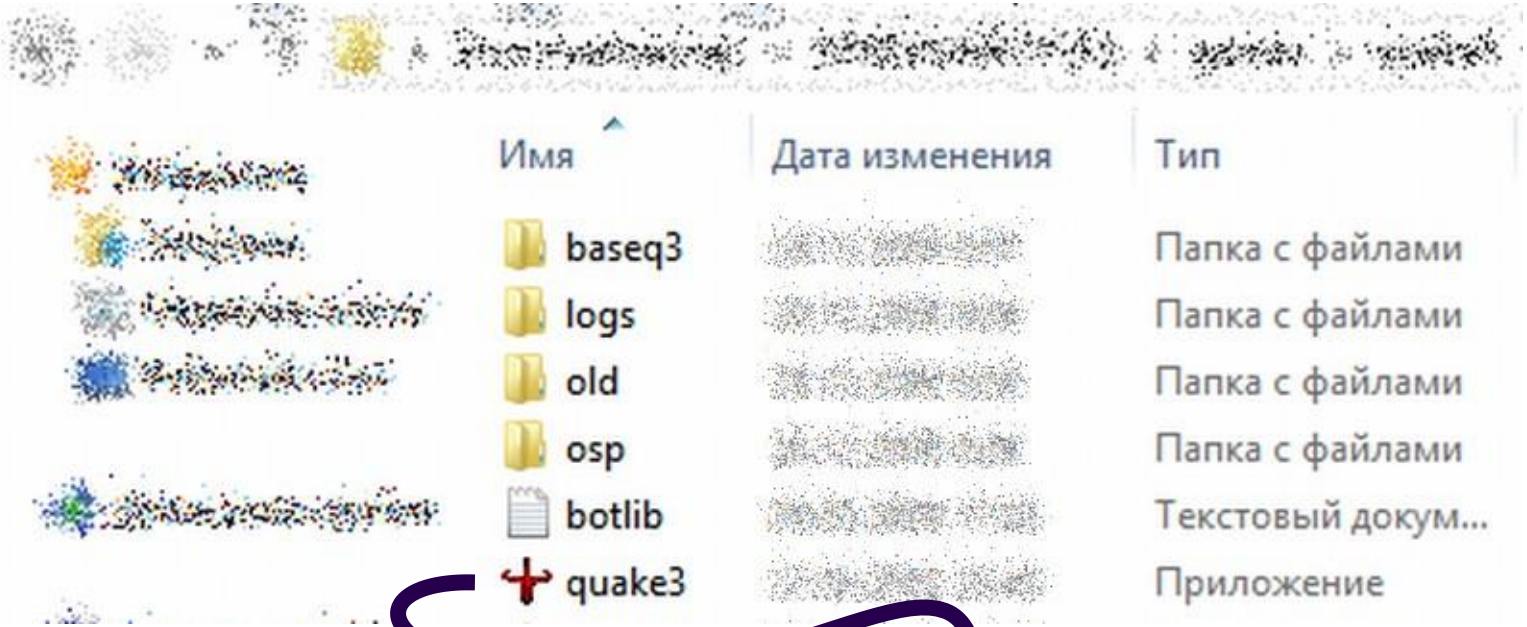
- Базисная подсистема
  - Работает в рамках разделов диска
  - Управляет размещением файлов на ЖД
  - Управляет свободным пространством
- Логическая подсистема
  - Управляет иерархией каталогов
  - Доступ к файлам по абсолютному имени
  - Проверка прав доступа

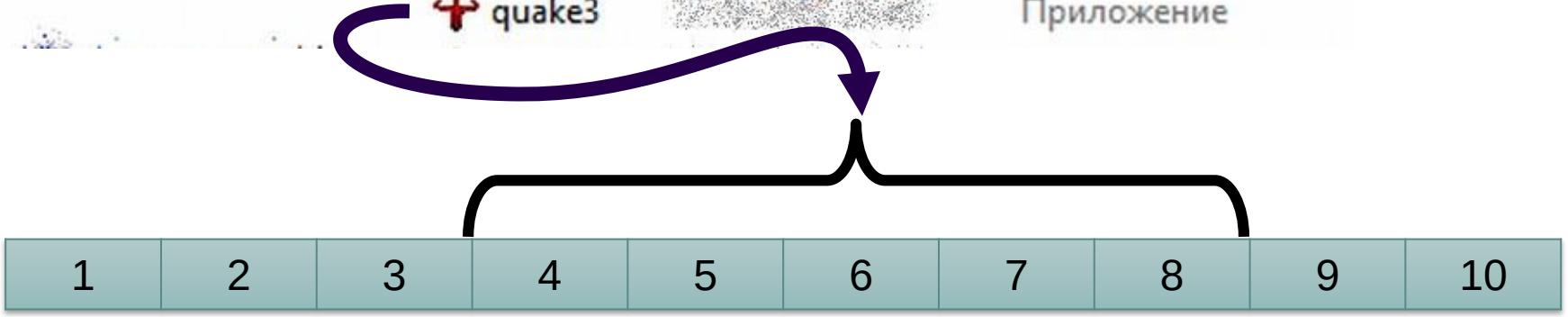
Способ выделения пространства для файлов



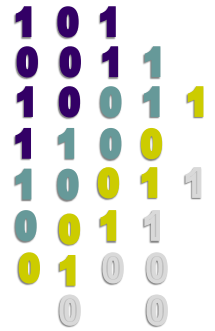
- Непрерывная последовательность блоков

Непрерывная последовательность блоков





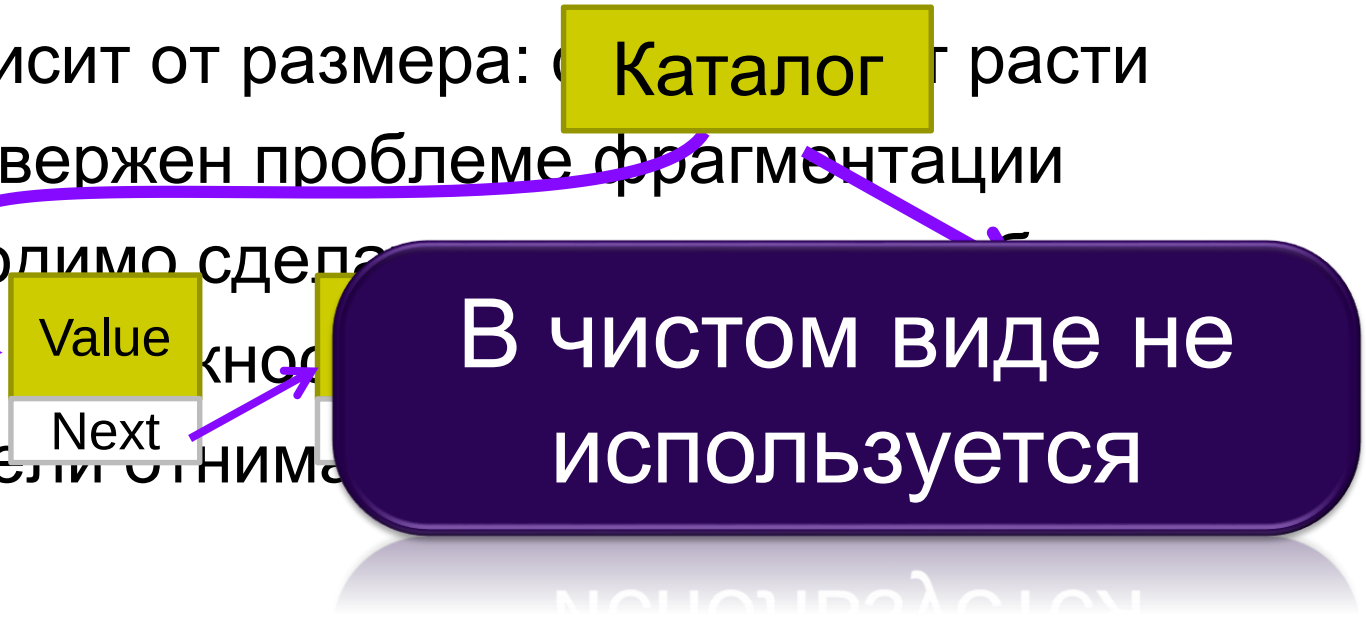
Способ выделения пространства для файлов



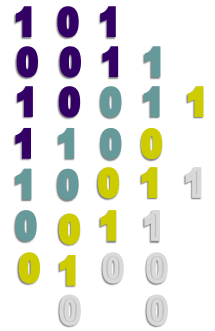
- Непрерывная последовательность блоков
  - Указатель на начало + размер
  - \+ Простота в реализации
  - \+ Производительность
  - – Фрагментация/проблема поиска места
    - First fit, best fit и worst fit
  - Невозможно оценить размер файла заранее
    - Резервные блоки

Связный список

- В директории ссылки на первый и последний фрагмент файла
  - В самих фрагментах хранятся ссылки на следующий фрагмент
  - Не зависит от размера: ф т расти
  - Не подвержен проблеме фрагментации
  - Необходимо сдела
  - Низкая кно
  - Указатели отним

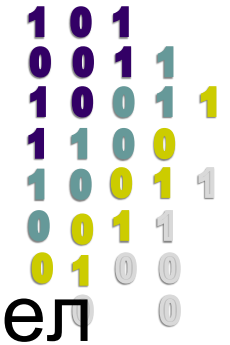


Связный список с использованием индекса



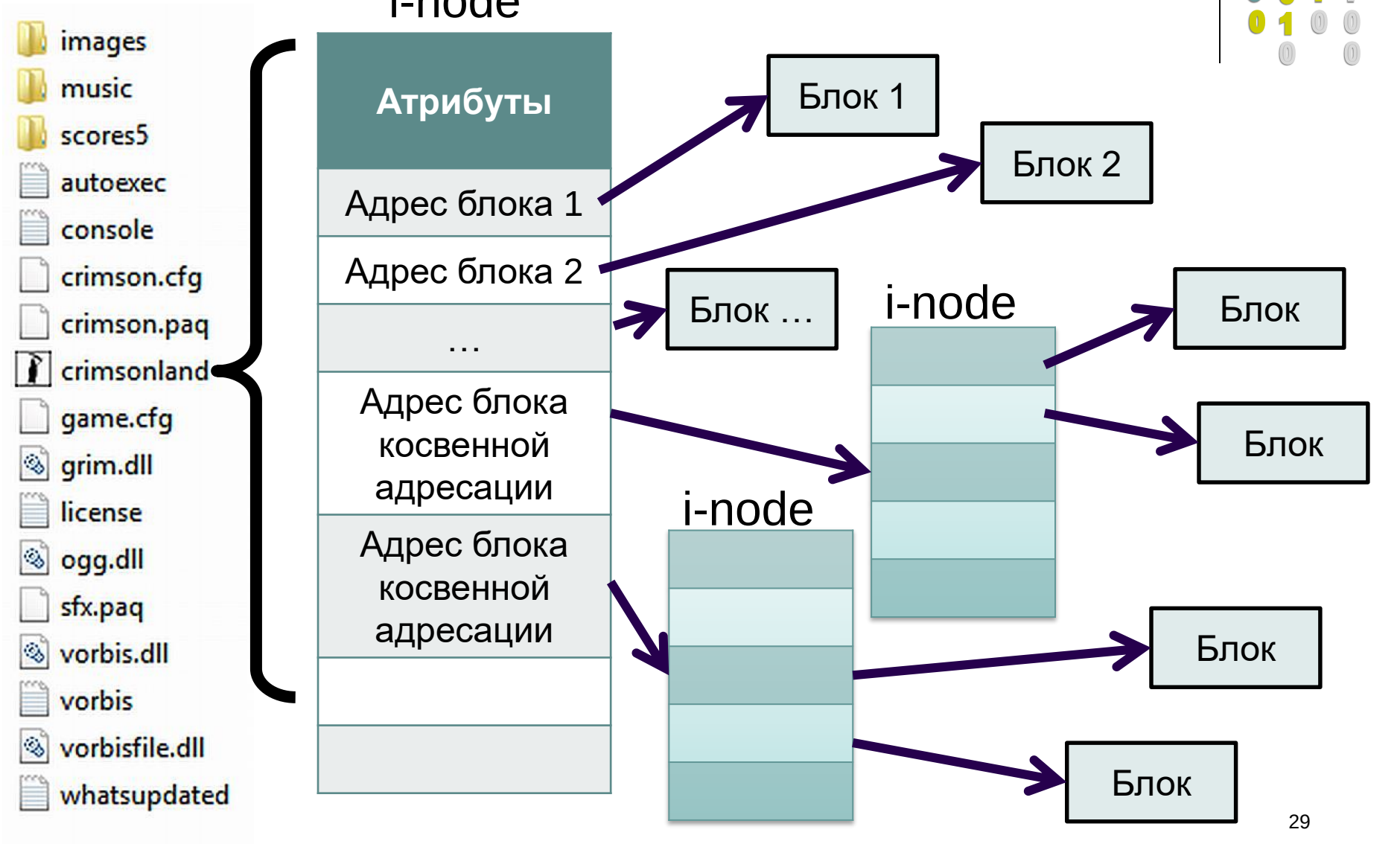
- Указатели на следующий блок хранятся не в самом блоке, а в отдельной индексной таблице в памяти
- FAT — File Allocation Table
- Наследует все плюсы предыдущего
- К недостаткам относится расход памяти на таблицу
- MS-DOS, OS/2, MS Windows

Индексные узлы

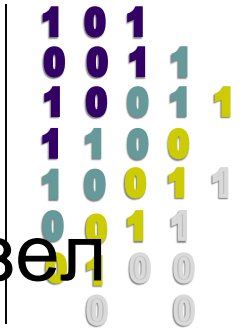


- С каждым файлом связан индексный узел
  - **i-node**
  - Небольшая таблица с атрибутами файла и его блоками на диске
  - Запись файла в директории содержит указатель на его i-node

Индексные узлы

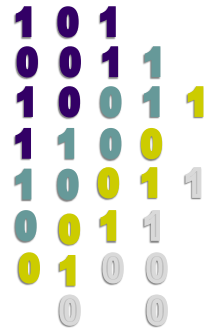


Индексные узлы



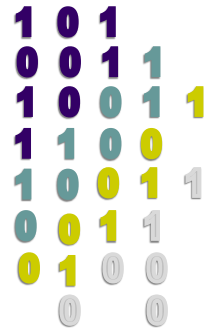
- С каждым файлом связан индексный узел
  - **i-node**
  - Небольшая таблица с атрибутами файла и его блоками на диске
    - Не подвержен внешней фрагментации
  - Указатели на первые блоки хранятся в самом i-node
    - Иначе используются блоки косвенной адресации
    - Блок двойной косвенной адресации, блок тройной косвенной адресации
  - UNIX, NTFS

Индексные узлы Особенности



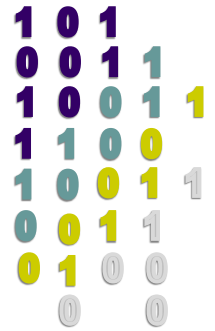
- Иноды часто располагаются в виде массива
- Иерархия каталогов полностью отделена от массива инодов
- На один инод может указывать несколько файлов в иерархии
  - Жёсткие ссылки
  - Нет обратной ссылки: от инода к имени файла
    - Невозможно определить как называется открытый или удалённый файл
  - Номер инода не изменяется при операциях
    - Программа может работать даже с удалённым/перемещённым файлом если он уже открыт!

Контроль свободного пространства



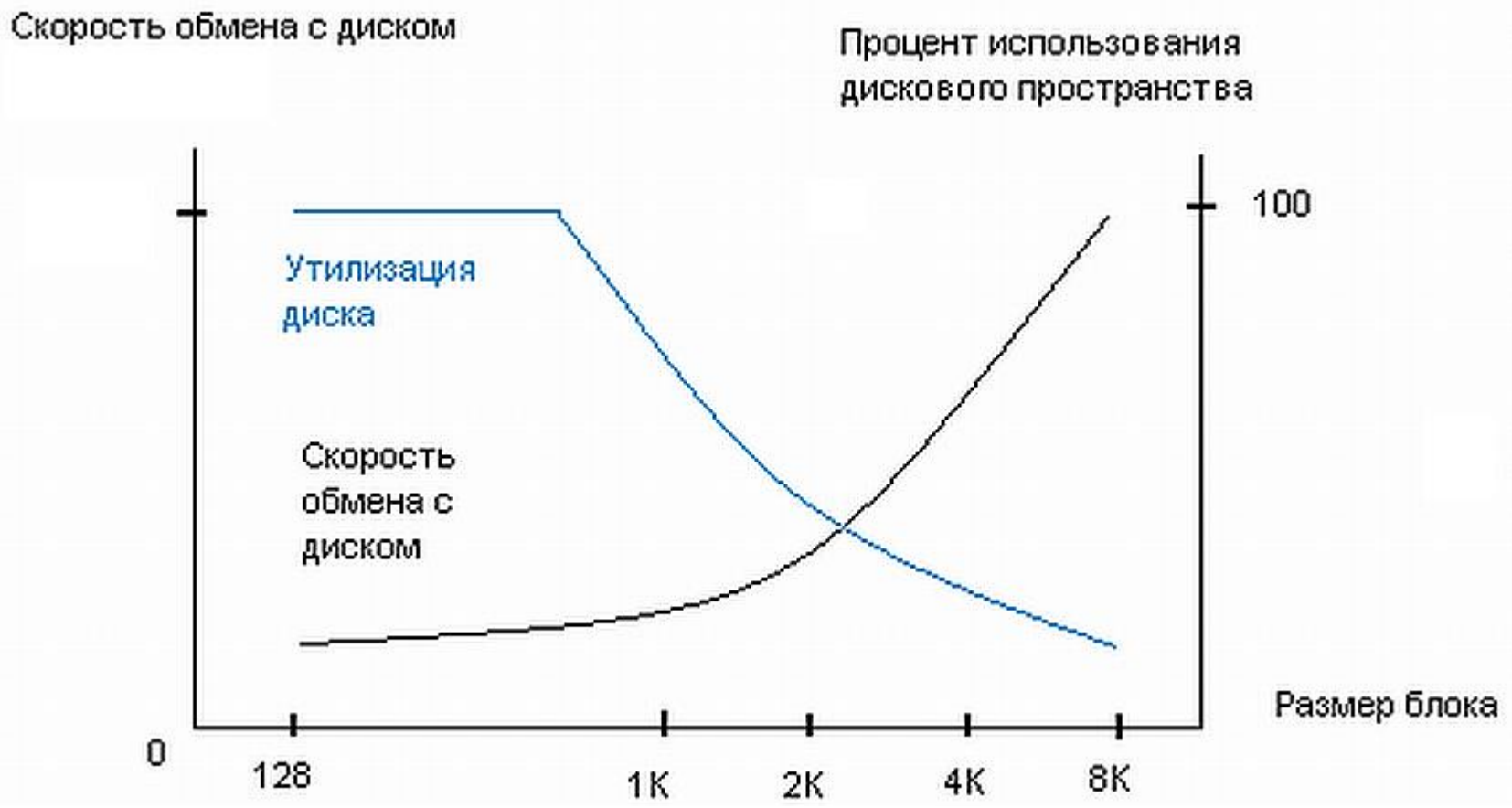
- Битовый вектор
  - Каждый блок — бит
  - 0 или 1 — занят или свободен
  - 00111100111100011000001 ....
- Связный список
  - Список всех свободных блоков
  - Храним указатель на первый свободный блок
  - Необходим обход всего списка

Размер блока

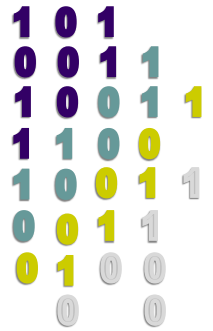


- Размер блока ФС не соответствует размеру физического блока на носителе
- Чем меньше блок
  - Тем больше блоков в одном файле
    - Медленнее операции
  - Тем меньше потерь пространства
- Даже если файл 1 байт — будет потерян 1 блок

Размер блока



Структура файловой системы на диске



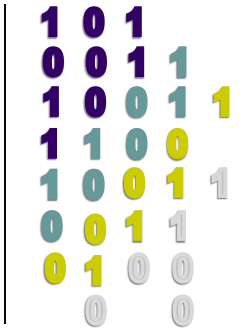
- Создаются при форматировании

Суперблок	Контроль свободного пространства	Массив индексных узлов	Блоки данных

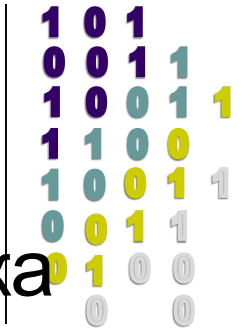
- Суперблок
  - Содержит общее описание файловой системы
    - Тип файловой системы
    - Размер файловой системы в блоках
    - Размер массива индексных узлов
    - Размер логического блока
    - …

Суперблок

```
# dumpe2fs /dev/sda1
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          52872df9-2917-440f-97c5-af172b87db85
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_
Filesystem flags:         signed_directory_hash
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              1831424
Block count:              7323624
Reserved block count:     366181
Free blocks:              5603250
Free inodes:              1603044
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      1022
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8176
Inode blocks per group:   511
RAID stride:              1
Filesystem created:       Fri Oct  7 21:09:40 2011
Last mount time:          Mon Apr 20 08:01:32 2015
Last write time:          Tue Mar 17 13:54:00 2015
Mount count:              4
Maximum mount count:      34
Last checked:             Tue Mar 17 13:54:00 2015
Check interval:           15552000 (6 months)
Next check after:         Sun Sep 13 14:54:00 2015
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
```

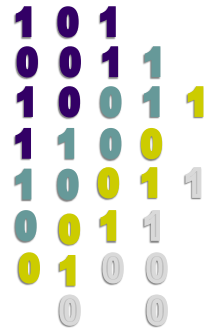


Структура ФС на диске



- Существует несколько копий суперблока
  - Он модифицируется относительно редко
- Размер массива индексных узлов определяется при форматировании
  - В системе невозможно создать файлов больше чем индексных узлов
- Один инод может принадлежать нескольким именам файлов
- Один блок данных принадлежит **только** одному иноду

Директории ФС



- Нет операции write: createFile, removeFile, …
  - Структура директории скрыта
  - Доступ посредством указания пути
    - Основная задача: соответствие между абсолютным именем файла и его данными
  - Атрибуты могут храниться в директории или в самом объекте
    - Предусмотрена ли возможность хранения атрибутов в объекте

Пример

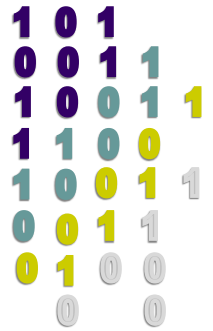
- DOS

Имя файла	Расширение	Атрибуты (файл/каталог)	Резерв	Время	Дата	Номер первого блока	Размер

- UNIX

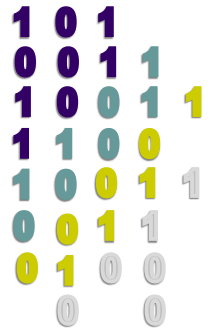
Inode	Имя файла

Монтирование файловых систем



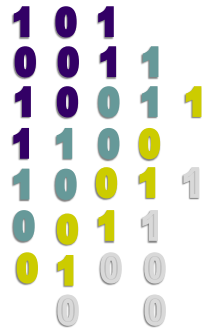
- ФС необходимо подключить, чтобы она была доступна для процессов
- mount <устройство> <точка подключения>
  - Система пытается определить тип ФС, найти её структуры и обработать их
  - ФС подключается в общее дерево каталогов
  - Особенностью работы является переход точки монтирования
    - Вниз по иерархии
    - Вверх по иерархии

Ссылки



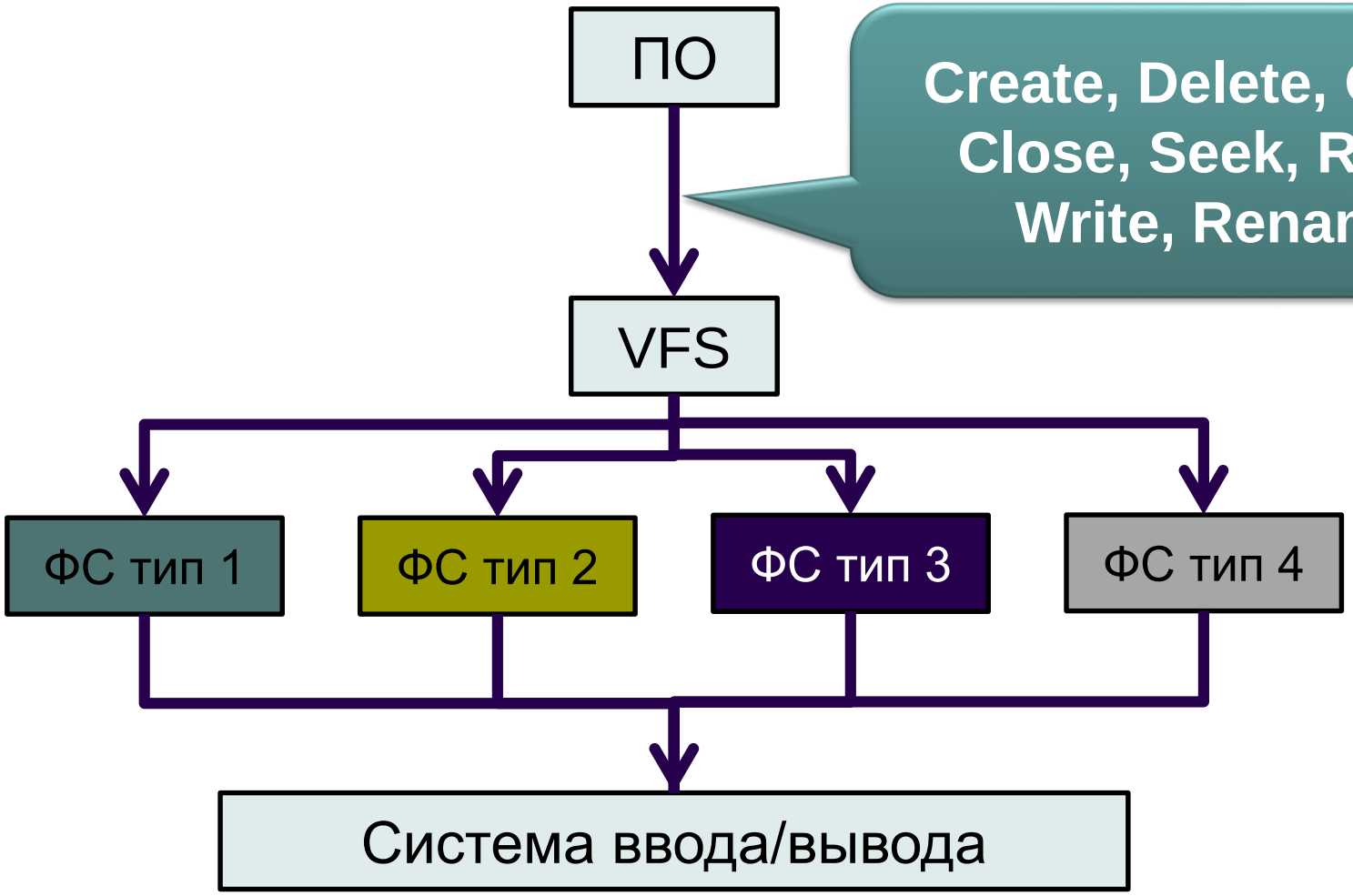
- Необходимость иметь несколько копий файла в разных директориях
  - Не дублируя информацию
  - Жёсткие ссылки: очевидная реализация в ФС с индексными файлами
    - Несколько имён ссылается на один и тот же инод
    - Счётчик ссылок
  - Символические ссылки
    - Файл с информацией о том, какой файл нужно открыть в случае обращения к данному

Организация работы нескольких ФС в Linux



- К одной ОС подключены несколько типов ФС
  - Различные структуры, операции, ACL и т.д.
  - Подавляющее большинство программ выполняют только стандартный набор: **Create, Delete, Open, Close, Seek, Read, Write, Rename**
- ОС реализует прослойку под названием VFS
  - Virtual File System — виртуальная ФС
  - VFS — представляет программам стандартный API, вне зависимости от того где находится файл
  - Для каждой ФС всё ещё можно выполнять специальные команды

VFS


ПО
Create, Delete, Open, Close, Seek, Read, Write, Rename
VFS
ФС тип 1
ФС тип 2
ФС тип 3
ФС тип 4
Система ввода/вывода

Вопросы?