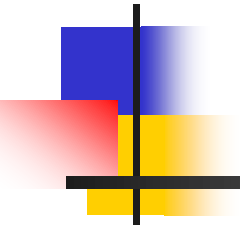


Тема 3

Файловые системы

Файловые системы

Часть операционной системы, работающей с файлами, называется файловой системой.

Наиболее важным аспектом ФС ***с точки зрения пользователя*** является ее внешнее представление, т.е.:

- именование и защита файлов;

- операции с файлами;

Именование файлов

Все современные ОС в качестве имен файлов используют 8-ми символьные текстовые строки. ***Также в именах файлов также разрешается использование цифр и специальных символов. Многие файловые системы поддерживают имена файлов до 255 символов.***

Именование файлов

В некоторых файловых системах есть различие между прописными и строчными буквами (например, Unix). В ОС MS-DOS такого различия нет.

Именование файлов

Во многих ОС имена файлов могут состоять из 2-х частей, разделенных точкой (***имя файла. расширение файла***) и обычно означает тип файла. В MS-DOS имя файла содержит 8 символов + ***3 символа отводится на расширение файла***.

В некоторых ОС (например, Unix) расширения файлов являются просто соглашениями, которые могут придерживаться пользователи.

Типы файлов

Регулярные файлы – к ним относятся все файлы, содержащие информацию пользователя.

Каталоги – системные файлы, обеспечивающие поддержку структуры ФС.

Типы файлов

Символьные специальные файлы – имеют отношение к вводу-выводу и используются для моделирования последовательных устройств ввода-вывода, таких, как терминалы, принтеры, сети.

Блочные специальные файлы – используются для моделирования дисков.

Структура файла

Файлы могут быть структурированы несколькими различными способами.

1) Неструктурированная последовательность байтов. В этом случае ОС не интересуется содержимым файла. Все что она видит, это байты. Значения этим байтам придается программами уровня пользователя. Такой подход используется в ОС Windows, Unix.

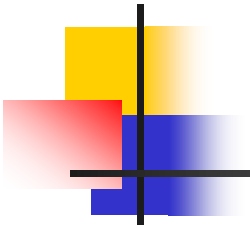
Такой подход обеспечивает максимальную гибкость.

Структура файла

2) Последовательность записей – первый шаг к структурированию файлов. В данной модели файл представляет собой последовательность записей фиксированной длины, каждая со своей внутренней структурой. В этом подходе важным является то, что операция чтения возвращает одну запись, а операция записи перезаписывает или дополняет одну запись.

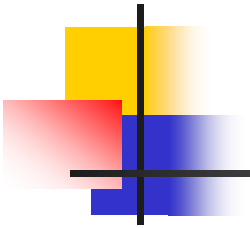
Структура файла

Третий вариант – файл представляет собой дерево записей, не обязательно одной и той же длины. Каждая запись в фиксированной позиции содержит поле ключа. Дерево сортировано по ключевому полю, что обеспечивает быстрый поиск заданного ключа. Такой вариант представления файловых систем широко применяются на больших мэйнфреймах, которые используются для коммерческой обработки данных.



Системные вызовы

Основной функцией операционной системы является скрытие особенностей дисков и других устройств ввода-вывода и предоставление пользователю понятной и удобной абстрактной модели независимых от устройств файлов. Системные вызовы очевидно необходимы для создания, удаления, чтения или записи файлов.

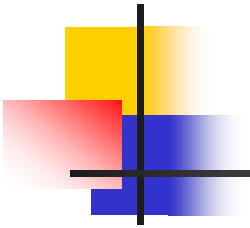


Системные вызовы

Систе́мный вы́зов (англ. system call) в программировании и вычислительной технике — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

Системные вызовы

Современные операционные системы (ОС) предусматривают разделение времени между выполняющимися вычислительными процессами (многозадачность) и разделение полномочий, препятствующее исполняемым программам обращаться к данным других программ и оборудованию. Ядро ОС исполняется в привилегированном режиме работы процессора. Для выполнения межпроцессной операции или операции, требующей доступа к оборудованию, программа обращается к ядру, которое, в зависимости от полномочий вызывающего процесса, исполняет либо отказывает в исполнении такого вызова.



Системные вызовы

С точки зрения программиста системный вызов обычно выглядит как вызов подпрограммы или функции из системной библиотеки. Однако системный вызов как частный случай вызова такой функции или подпрограммы следует отличать от более общего обращения к системной библиотеке, поскольку последнее может и не требовать выполнения привилегированных операций.

Системные вызовы для работы с файлами

Create (Создание). Файл создается без данных. Этот системный вызов объявляет о появлении нового файла и позволяет установить некоторые его атрибуты.

***Delete (Удаление*)**. Удаление файла.

Open (Открытие) – перед работой с файлом, необходимо его открыть. Данный системный вызов позволяет системе прочитать в оперативную память атрибуты файла и список дисковых адресов для быстрого доступа к содержимому файла при последующих вызовах.

Системные вызовы для работы с файлами

Close (закрытие) – закрытие файла.

Read (чтение). Чтение данных из файла.

Write (запись). Если текущая позиция находится в конце файла, размер файла автоматически увеличивается. В противном случае запись производится поверх существующих данных, которые теряются навсегда.

Системные вызовы для работы с файлами

Append (добавление). Усеченная форма Write. Может добавлять данные только в конец файла.

Seek (поиск). Для файлов произвольного доступа. Устанавливает файловый указатель в определенную позицию в файле. После выполнения этого системного вызова данные могут читаться или записываться в этой позиции.

Системные вызовы для работы с файлами

Get_attributes (получение атрибутов).

Set_attributes (Установка атрибутов)

Rename (переименование) – изменение имени файла.

Организация каталогов

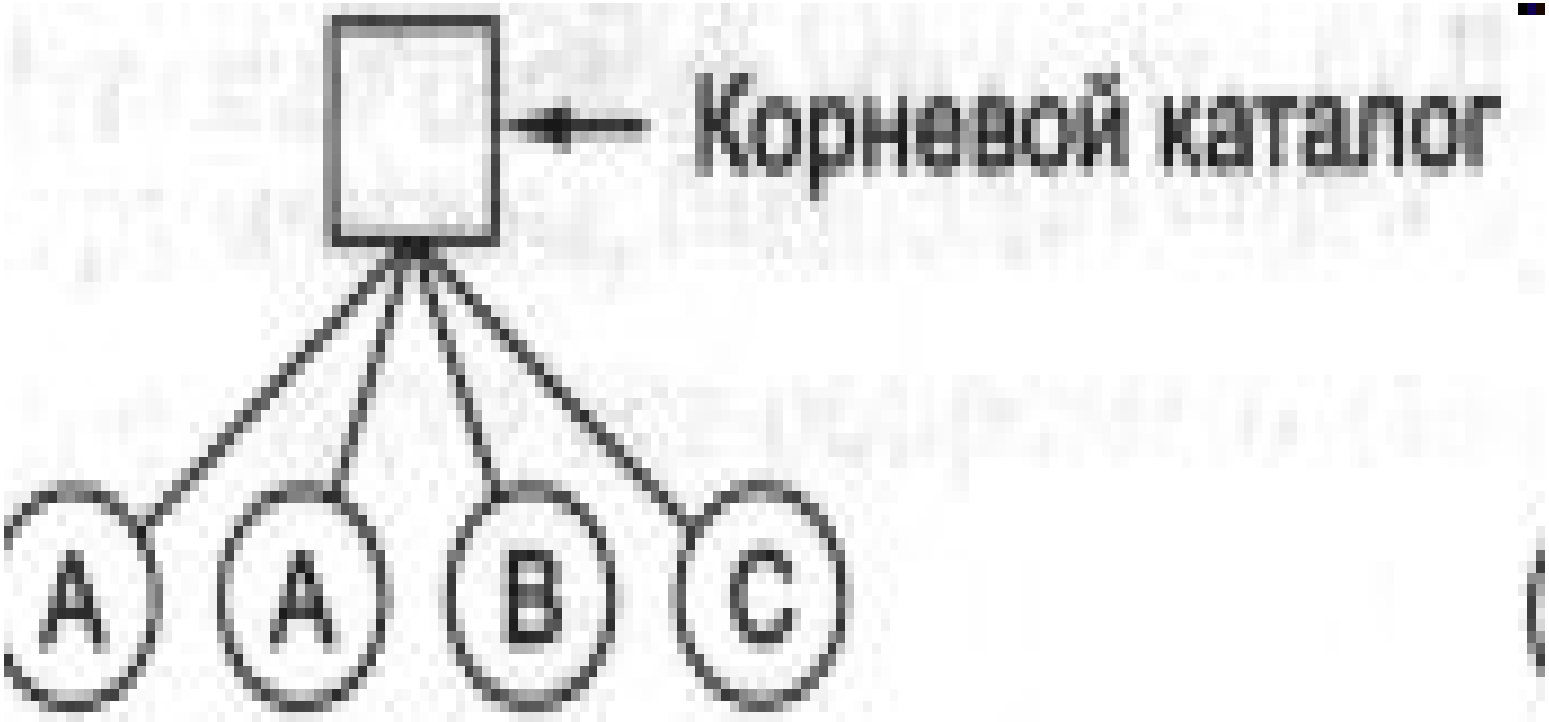
В файловых системах файлы обычно организуются в каталоги или папки, которые, в свою очередь, в большинстве операционных систем также являются файлами.

Организация каталогов

Одноуровневые каталоговые системы.

Простейшая форма системы каталогов состоит в том, что имеется один каталог, в котором содержатся все файлы. Иногда его называют корневым каталогом, но поскольку он в таких системах единственный, его название не имеет значение.

Организация каталогов



Организация каталогов

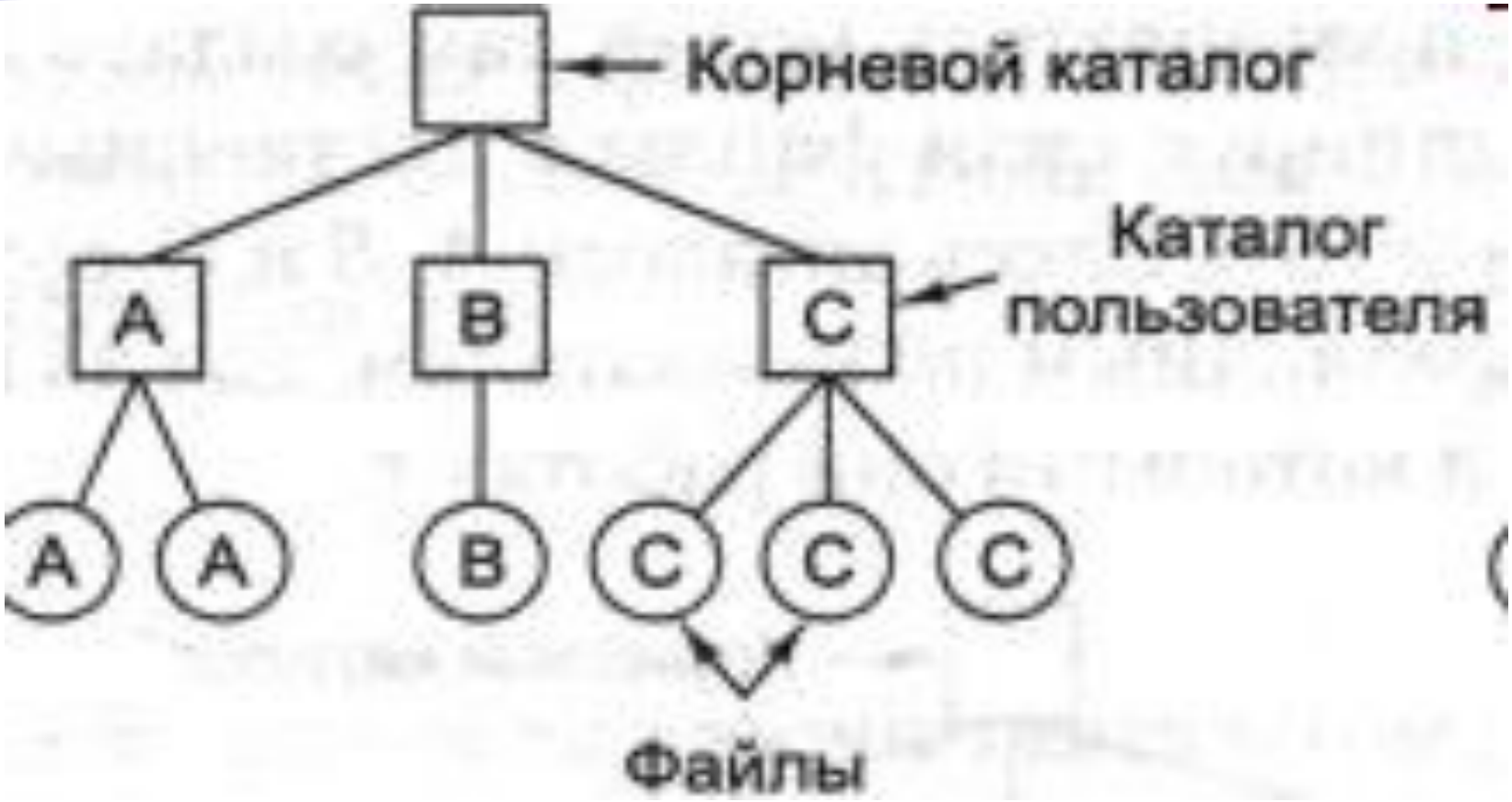
Недостаток системы с одним каталогом и несколькими пользователями состоит в том, что различные пользователи могут случайно использовать для своих файлов одинаковые имена.

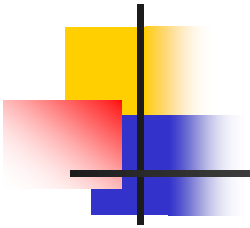
Организация каталогов

Двухуровневая система каталогов

Первым этапом в деле решения проблемы одинаковых имен файлов, созданных различными пользователями, можно считать систему, в которой каждому пользователю выделяется один каталог. При этом имена файлов, созданных одним пользователем, не конфликтуют с именами файлов другого пользователя.

Организация каталогов





Организация каталогов

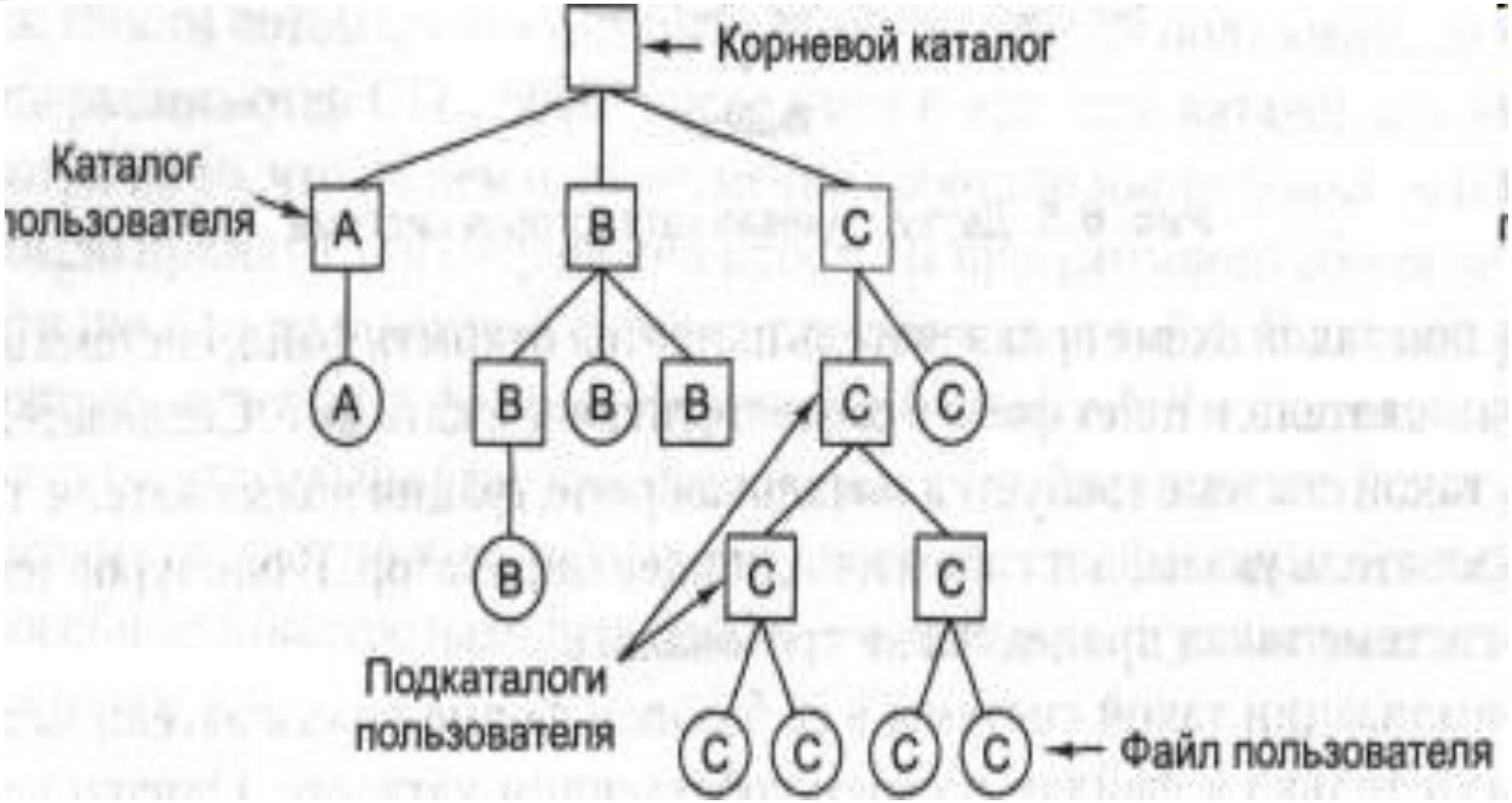
При реализации такой системы в ее базовой форме пользователи могут получать доступ только к файлам в своем собственном каталоге. Однако небольшая модификация основной схемы позволяет пользователям получать доступ к файлам других пользователей.

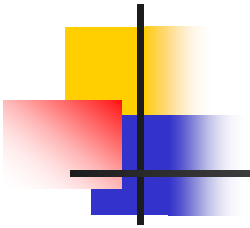
Организация каталогов

Иерархические каталоговые системы

Благодаря двухуровневой иерархии исчезают конфликты имен файлов между различными пользователями, но ее недостаточно для пользователей с большим числом файлов. Обычно пользователям бывает необходимо логически группировать свои файлы.

Организация каталогов





Организация каталогов

Возможность создавать произвольное количество подкаталогов является мощным структурирующим инструментом, позволяющим пользователям организовать свою работу. По этой причине почти все современные файловые системы организованы подобным образом.

Системные вызовы

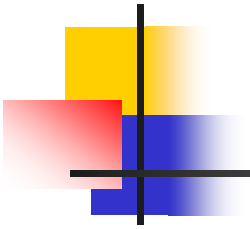
Системные вызовы для работы с каталогами:

Createdir() – создание каталога.

Deletedir() – удаление каталога.

Opendir ()– открытие каталога.

Closedir ()– закрытие каталога.



Системные вызовы

Readdir – чтение следующего элемента открытого каталога.

Rename – переименование каталога.

Link – Установление связей.

Unlink – Удаление ссылки на файл из каталога.

Атрибуты файла

Защита - Кто и каким образом может получить доступ к файлу

Пароль - Пароль для получения доступа к файлу

Создатель - Идентификатор пользователя, создавшего файл

Владелец - Текущий владелец

Атрибуты файла

Флаг «Только чтение» - 0 –для чтения/записи; 1- только чтение

Флаг «Скрытый» 0 – нормальный, 1 – не отображать в перечне файлов каталога

Флаг «Системный» 0 – нормальный; 1- системный

Атрибуты файла

Флаг «Архивный» - 0 – заархивирован; 1 – требуется архивация

Флаг ASCII/двоичный - 0 – ASCII; 1- двоичный

Атрибуты файла

Флаг произвольного доступа - 0 – только последовательный доступ; 1 – произвольный доступ

Флаг «временный» - 0 – нормальный, 1- для удаления файла по окончании процесса

Атрибуты файла

Флаги блокировки - 0 – неблокированный; отличный от нуля для блокированного

Длина записи - Количество байтов в записи

Время создания - Дата и время создания файла

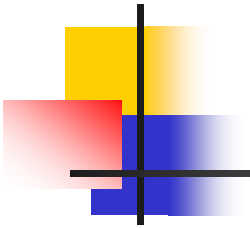
Атрибуты файла

Время последнего доступа - Дата и время последнего изменения файла

Текущий размер - Количество байтов в файле

Максимальный размер - Кол-во байтов до которого можно увеличивать размер файла

Файловые системы с точки зрения разработчика



Большинство дисков могут делиться на несколько разделов с независимой ФС на каждом разделе.

Сектор 0 диска называется главной загрузочной записью (MBR, Master Boot Record) и используется для загрузки компьютера.

Возможная структура ФС

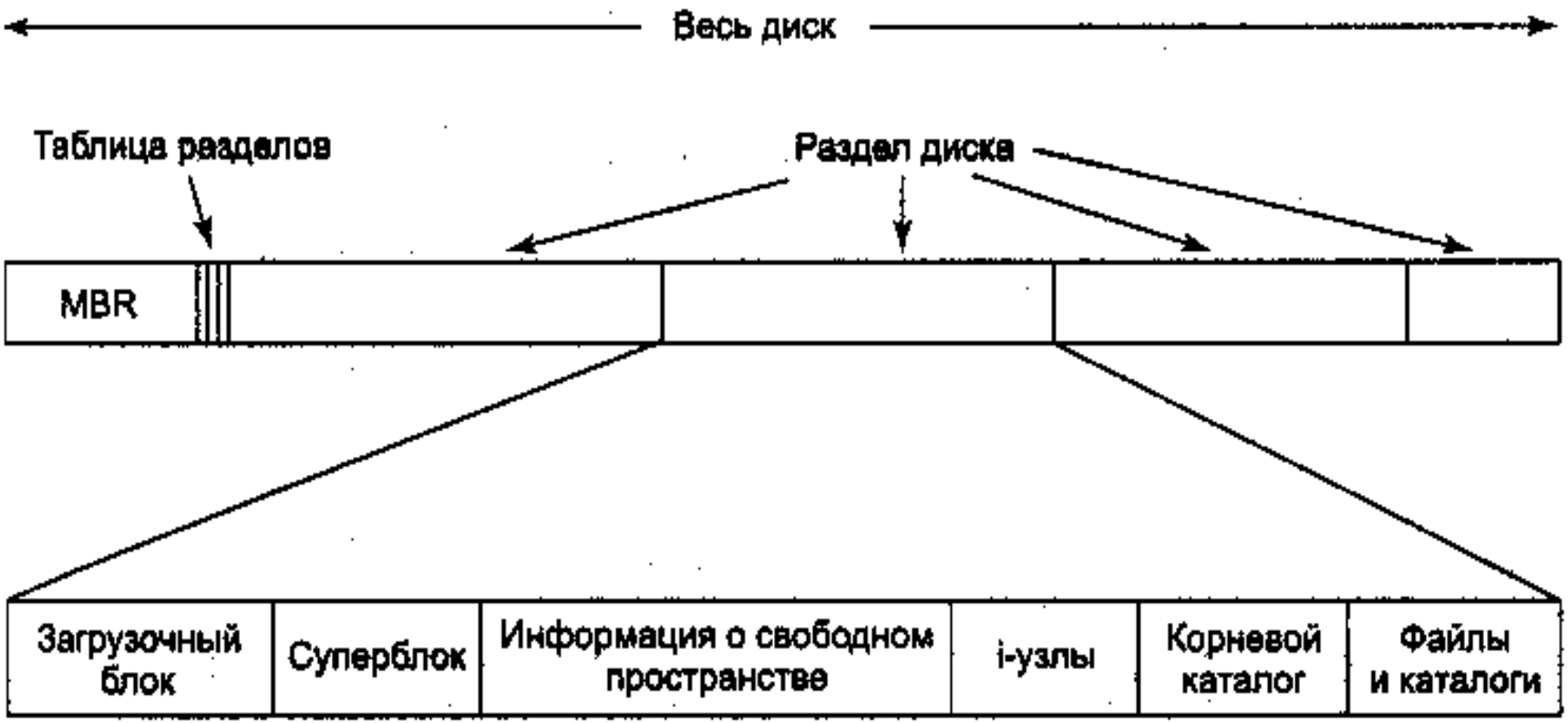
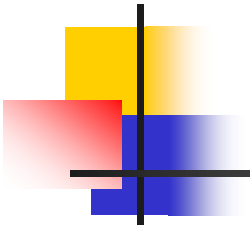


Рис. 6.8. Возможная структура файловой системы

Файловые системы с точки зрения разработчика

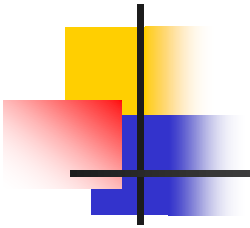


В конце главной загрузочной записи содержится таблица разделов.

В этой таблице хранятся начальные и конечные адреса (номера блоков) каждого раздела.

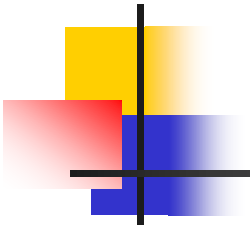
Один из разделов отмечен в таблице как активный.

Файловые системы с точки зрения разработчика



При загрузке компьютера BIOS считывает и исполняет MBR – запись, после чего загрузчик в MBR-записи определяет активный раздел диска, считывает его первый блок, называемый загрузочным, и исполняет его.

Файловые системы с точки зрения разработчика



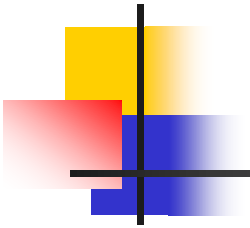
Программа, находящаяся в загрузочном блоке, загружает ОС, находящуюся в этом разделе.

Каждый дисковый раздел начинается с загрузочного блока, даже если в нем не содержится загружаемой ОС.

Возможная структура ФС

Суперблок - содержит ключевые параметры ФС и считывается в память при загрузке компьютера или при первом обращении к ФС.

Информация, хранящаяся в суперблоке, включает магическое число, позволяющее различать системные файлы, количество блоков в ФС, а также ключевую административную информацию.



Возможная структура ФС

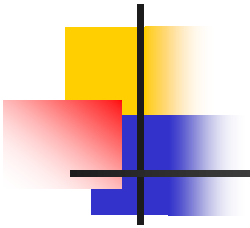
Следом располагается информация о свободных блоках файловой системы, например в виде битового массива или списка указателей.

Возможная структура ФС

i-узлы, представляющих собой массив структур данных, по одной структуре на файл, содержащих всю информацию о файлах.

Возможная структура ФС

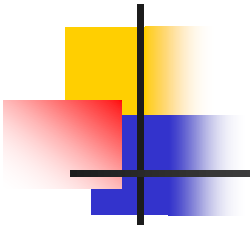
Корневой каталог, содержащий вершину дерева ФС. Остальное место дискового раздела занимают все остальные файлы и каталоги.



Реализация файлов

Для определения того, какой блок какому файлу принадлежит, в различных операционных системах применяются различные методы.

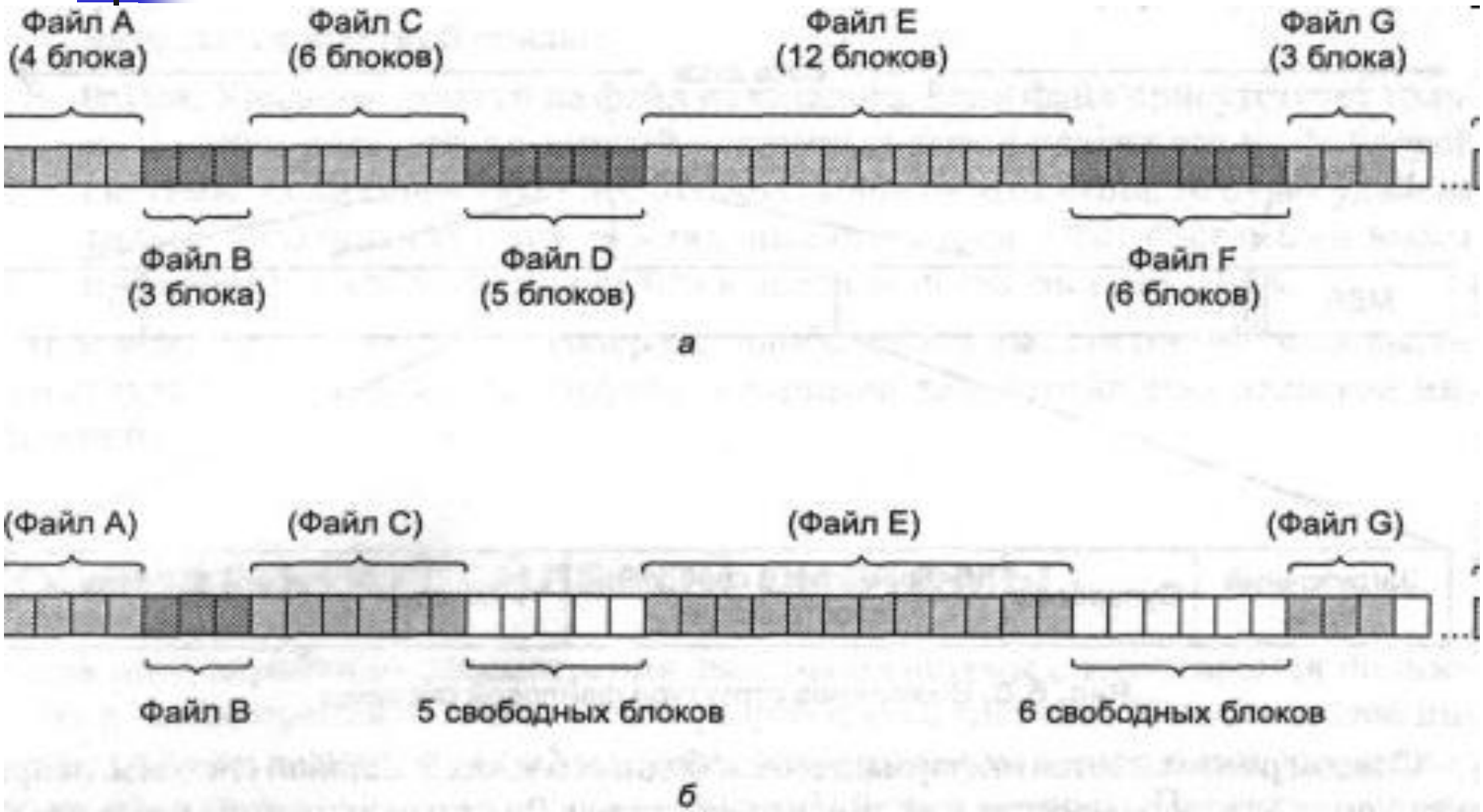
Рассмотрим некоторые из них.



Непрерывные файлы

Простейшей схемой выделения файлам определенных блоков на диске является система, в которой файлы представляют собой непрерывные наборы соседних блоков диска. Тогда на диске, состоящем из блоков по 1 Кбайт, файл размером в 50 Кбайт будет занимать 50 последовательных блоков.

Непрерывные файлы



Непрерывные файлы

Преимущества:

- ***Легко реализовать***, так как системе, чтобы определить, какие блоки принадлежат тому или иному файлу, нужно следить всего лишь за двумя числами: номером первого блока файла и числом блоков в файле.

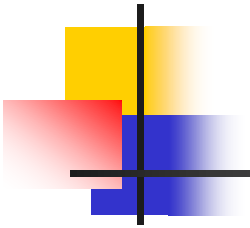
Непрерывные файлы

- ***Производительность высока***, так как весь файл может быть прочитан с диска за одну операцию. Требуется только одна операция поиска (для первого блока). После этого более не нужно искать цилиндры и тратить время на ожидания вращения диска, поэтому данные могут считываться с максимальной скоростью, на которую способен диск.

Непрерывные файлы

Недостаток: со временем диск становится фрагментированным.

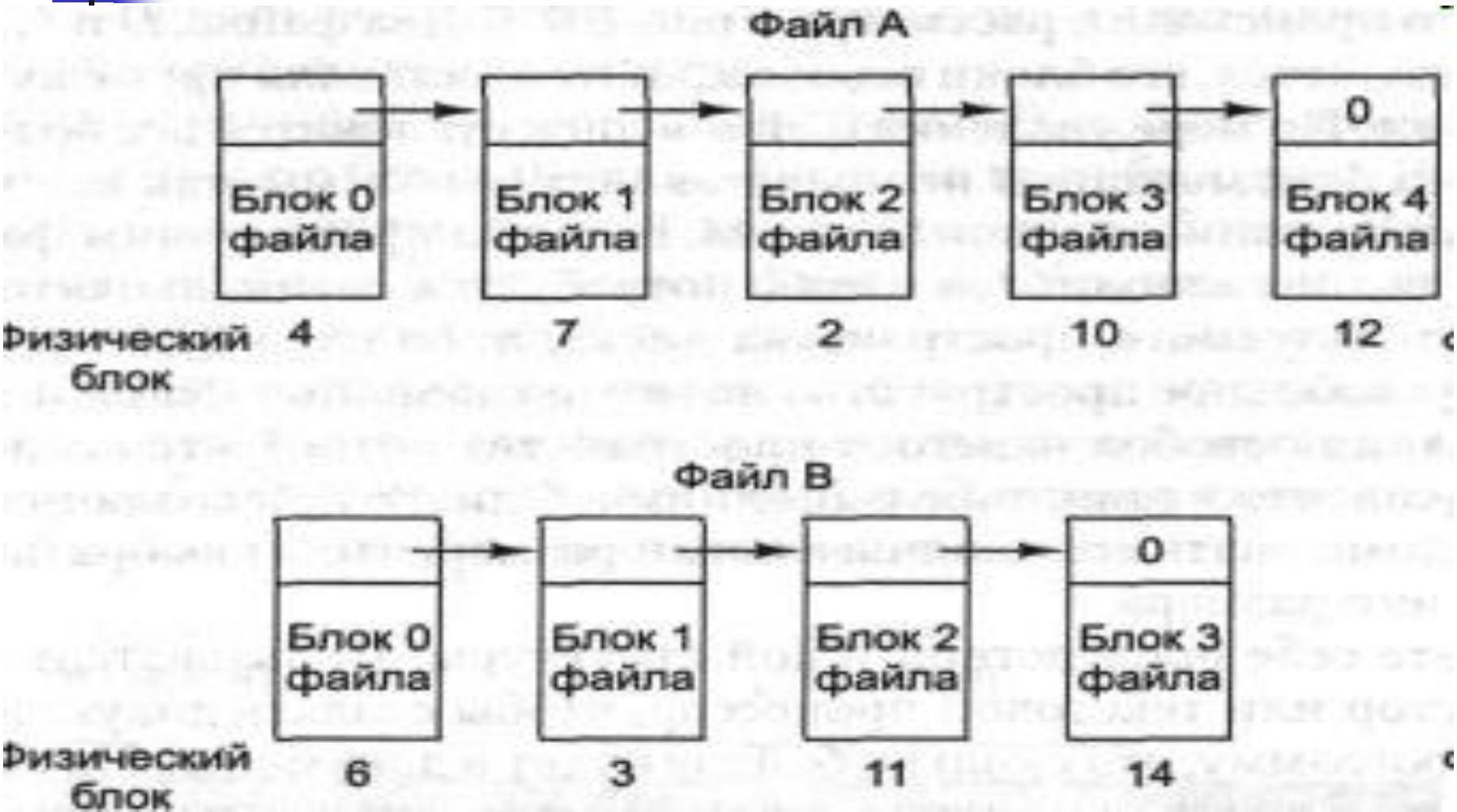
Есть ситуации, в которых непрерывные файлы могут применяться и в самом деле широко используются: на компакт-дисках. Здесь все размеры файлов известны заранее и не могут меняться при последующем использовании файловой системы CD-ROM.



Связные списки

Второй метод размещения файлов состоит в ***представлении каждого файла в виде связного списка из блоков диска***, как показано на рис. ***Первое слово каждого блока используется как указатель на следующий блок. В остальной части блока хранятся данные.***

Связные списки



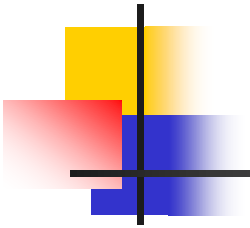
Связные списки

Такой метод позволяет использовать каждый блок диска. Нет потерь дискового пространства на фрагментацию (кроме потерь в последних блоках файла). Кроме того, ***в каталоге нужно хранить только адрес первого блока файла.*** Всю остальную информацию можно найти там.

Связные списки

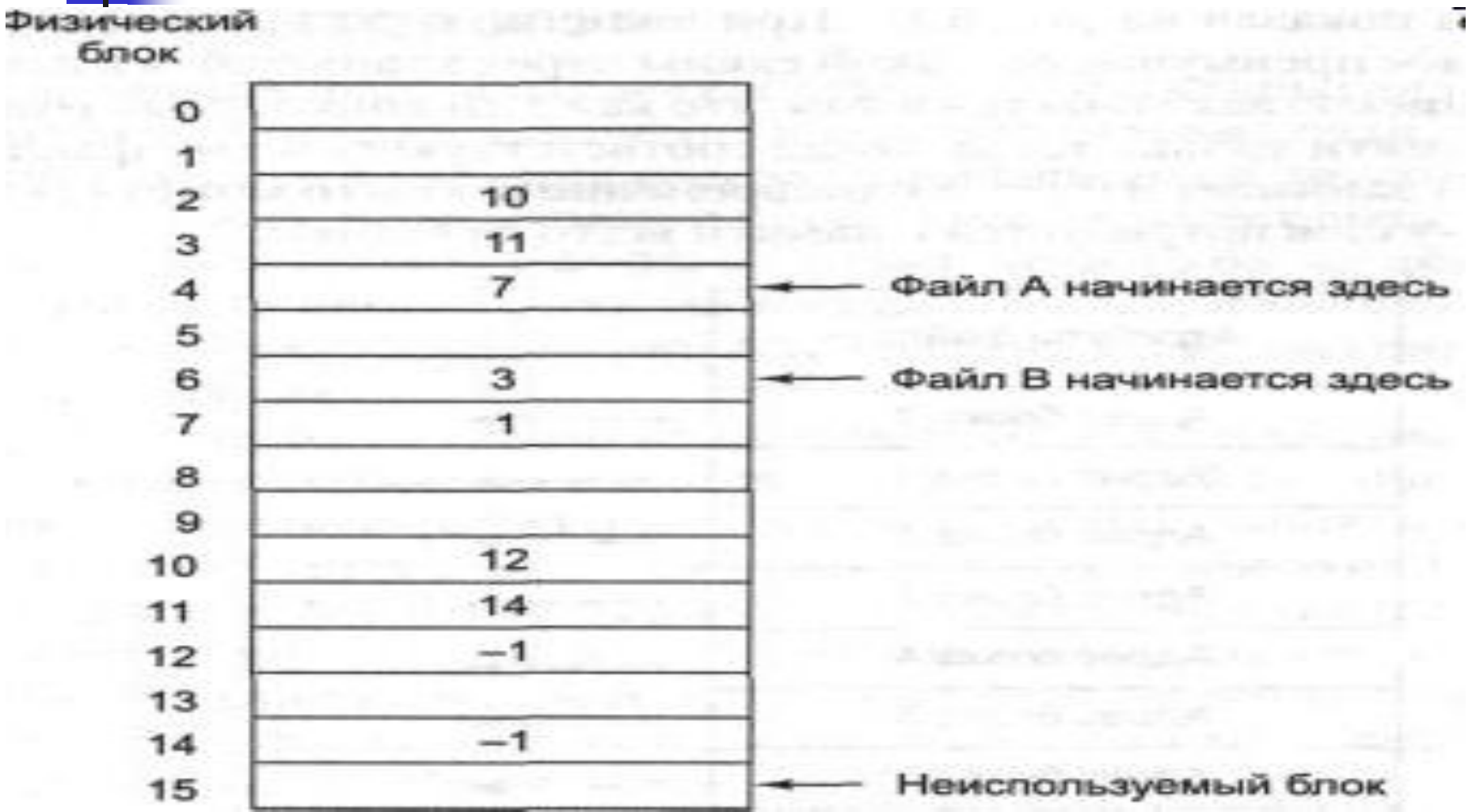
Последовательный доступ к такому файлу несложен, ***произвольный доступ будет довольно медленным***. Чтобы получить доступ к блоку n, операционная система должна сначала прочитать первые n - 1 блоков по очереди. Очевидно, такая схема оказывается очень медленной.

Связный список при помощи таблицы в памяти



Оба недостатка предыдущей схемы организации файлов в виде связных списков могут быть устранены, если указатели на следующие блоки хранить не прямо в блоках, а в отдельной таблице, загружаемой в память.

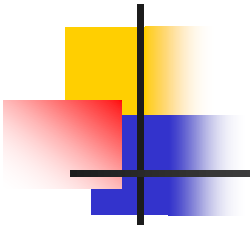
Связный список при помощи таблицы в памяти



Связный список при помощи таблицы в памяти

Такая таблица, загружаемая в оперативную память, называется FAT-таблицей (File Allocation Table — таблица размещения файлов).

Связный список при помощи таблицы в памяти



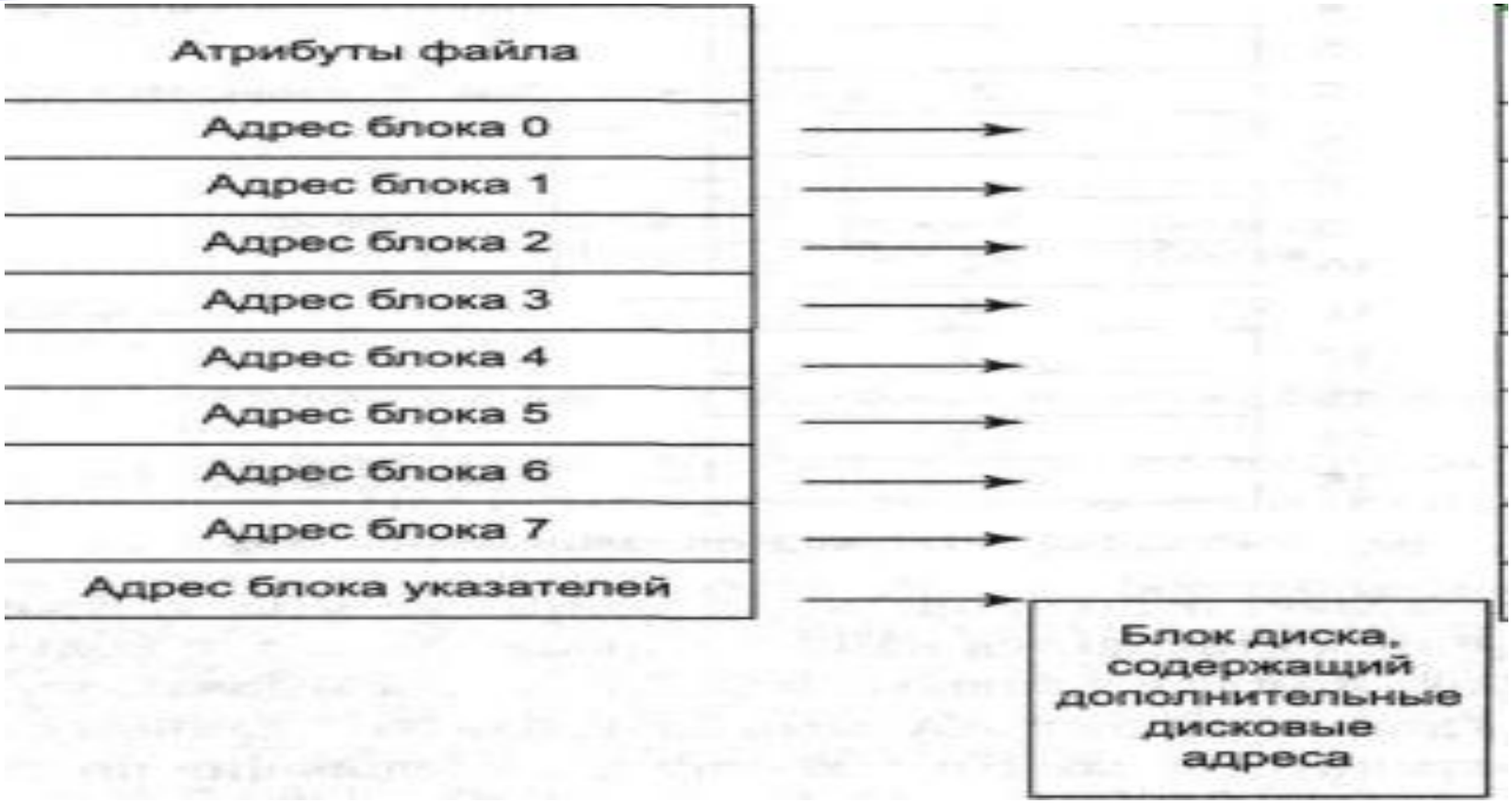
В каталоге достаточно хранить одно целое число (номер начального блока файла) для обеспечения доступа ко всему файлу.

Основной недостаток этого метода состоит в том, что вся таблица должна постоянно находиться в памяти.

I-узлы

Последний метод отслеживания принадлежности блоков диска файлам состоит в ***связывании с каждым файлом структуры данных, называемой i-узлом (index node — индекс-узел), содержащей атрибуты файла и адреса блоков файла.*** Простой пример i-узла показан на рис.

I-узлы


Атрибуты файла
Адрес блока 0
Адрес блока 1
Адрес блока 2
Адрес блока 3
Адрес блока 4
Адрес блока 5
Адрес блока 6
Адрес блока 7
Адрес блока указателей
Блок диска, содержащий дополнительные дисковые адреса

I-узлы

При наличии i-узла можно найти все блоки файла. Большое ***преимущество такой схемы*** перед хранящейся в памяти таблицей из связных списков заключается ***в том, что каждый конкретный i-узел должен находиться в памяти только тогда, когда соответствующий ему файл открыт.***

I-узлы

С такой схемой связана ***проблема***, заключающаяся в том, что ***при выделении каждому файлу фиксированного количества дисковых адресов этого количества может не хватить***. ***Одно из решений заключается в резервировании последнего дискового адреса не для блока данных, а для следующего адресного блока***

Реализация каталогов

При открытии файла операционная система использует поставляемое пользователем имя пути, чтобы найти запись в каталоге. ***Запись в каталоге содержит информацию, необходимую для нахождения блоков диска.*** В зависимости от системы это может быть дисковый адрес всего файла (для непрерывных файлов), номер первого блока файла (обе схемы связных списков) или номер i-узла. ***Во всех случаях основная функция каталоговой системы состоит в преобразовании ASCII-имени в информацию, необходимую для нахождения данных.***

Реализация каталогов

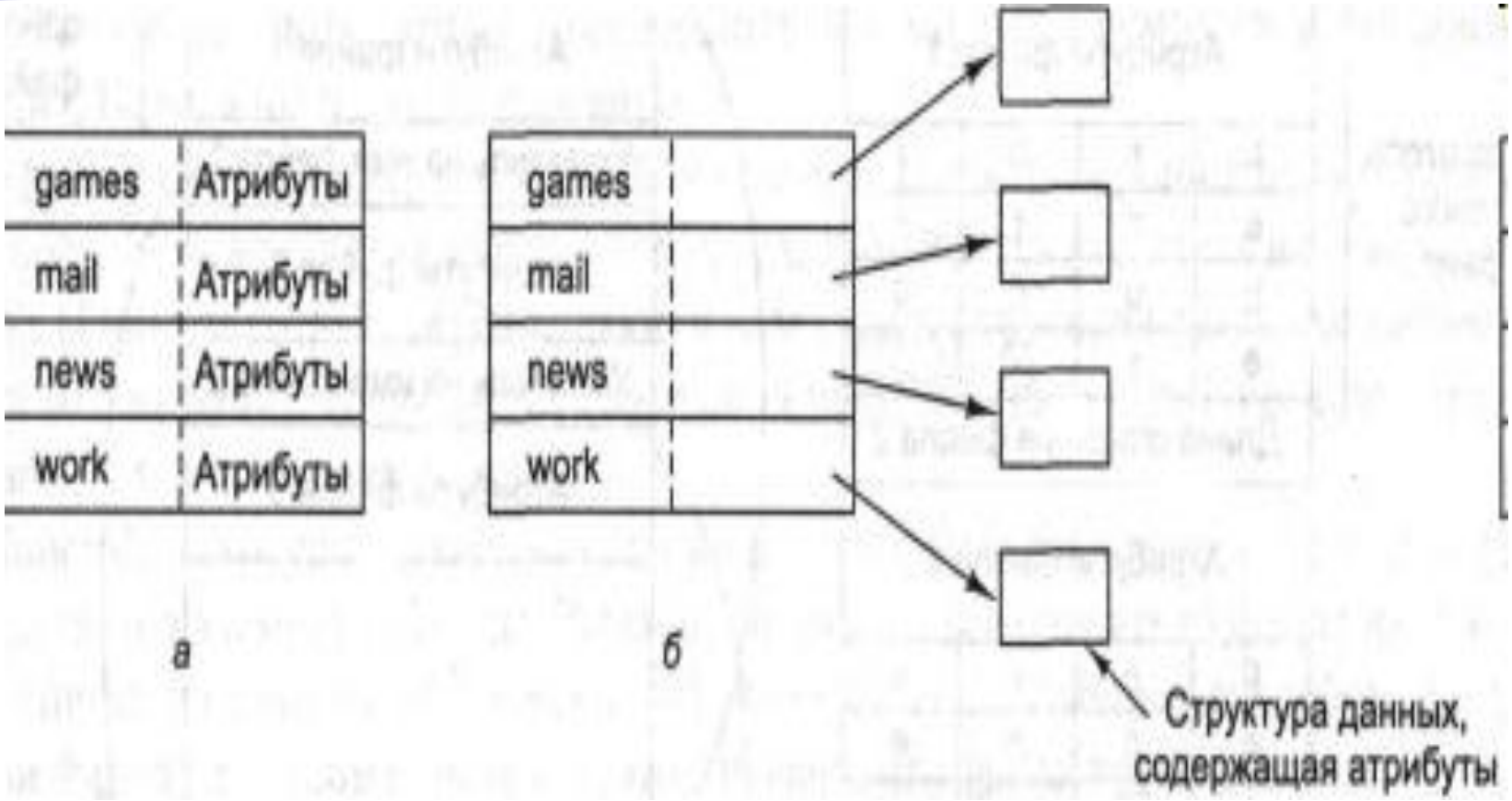
С этой проблемой тесно связан вопрос размещения атрибутов файла. Каждая файловая система поддерживает различные атрибуты файла, такие как дату создания файла, имя владельца файла и т. д., и всю эту информацию нужно где-то хранить.

Один из возможных вариантов состоит в хранении этих сведений прямо в каталоговой записи. Многие файловые системы именно так и поступают.

Реализация каталогов

Системы, использующие i-узлы, могут хранить атрибуты в i-узлах, а не в записях каталога. В этом случае ***запись в каталоге может быть короче: просто имя файла и номер i-узла.***

Реализация каталогов



Реализация каталогов

Проблема длинных имен файлов!

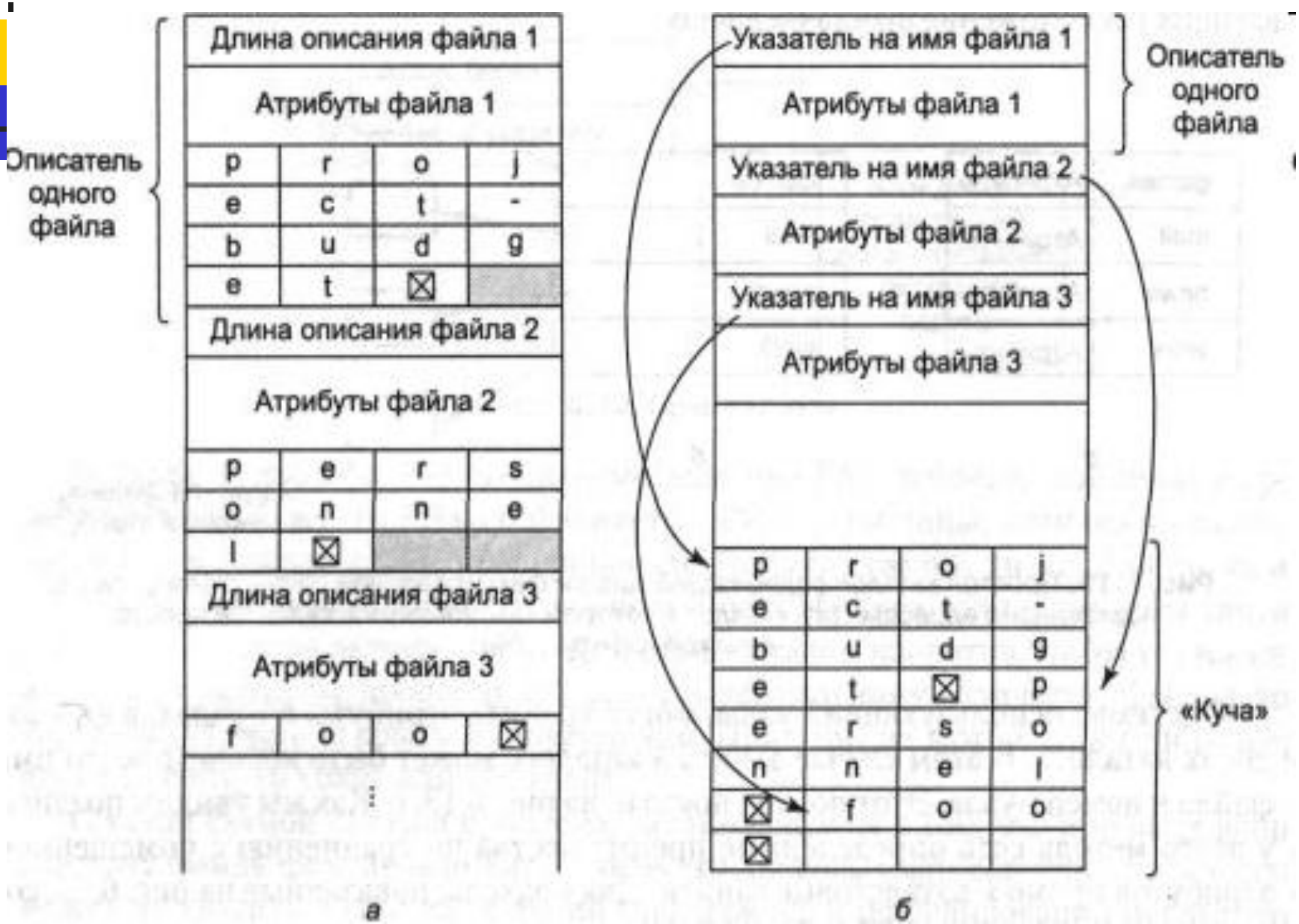
Файлы могут иметь короткие имена фиксированной длины. В системе MS-DOS файл может иметь имя длиной от 1 до 8 символов, а также расширение длиной до 3 символов. В системе UNIX Version 7 имена файлов могут быть от одного до 14 символов, включая расширение. ***Современными операционными системами поддерживаются более длинные имена файлов переменной длины.***

Реализация каталогов

Простейший метод состоит в установке ограничения на длину имени файла, обычно 255 символов, и использовании одной из схем, показанных на рис. ниже.

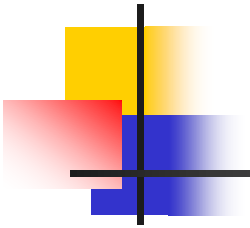
Такой способ прост, но он расходует много места в каталоге, так как ***длинные имена обычно бывают далеко не у всех файлов.***

Реализация каталогов



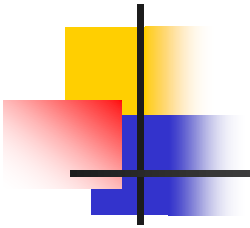
Реализация каталогов

Один из альтернативных подходов состоит в отказе от предположения о том, что все записи в каталоге должны иметь один и тот же размер. При таком подходе каждая запись в каталоге начинается с порции фиксированного размера, обычно начинающейся с длины записи, за которой следуют данные в фиксированном формате — идентификатор владельца, дата создания, информация о защите и прочие атрибуты. Следом за заголовком фиксированной длины идет часть записи переменной длины, содержащая имя файла (рис а)



Реализация каталогов

Недостаток этого метода состоит в том, что при удалении файла в каталоге остается промежуток переменной длины, в который описатель следующего файла может не поместиться. Эта проблема аналогична проблеме хранения на диске непрерывных файлов, хотя уплотнить каталог значительно легче, чем весь диск.

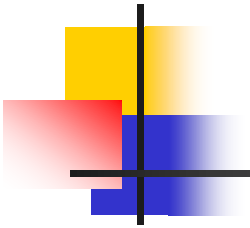


Реализация каталогов

Другая проблема связана с тем, что каталоговые записи переменной длины могут занимать сразу две страницы памяти. При чтении такой каталоговой записи может возникнуть прерывание из-за отсутствия в оперативной памяти следующей страницы.

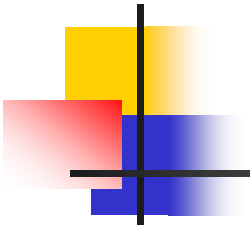
Реализация каталогов

Другой метод реализации длинных имен файлов заключается в том, чтобы сделать все записи каталога фиксированной (равной) длины и хранить в них только указатели на имена, а сами имена хранить отдельно в «куче», в конце каталога, как показано на рис. (б).



Реализация каталогов

Преимущество этого метода состоит в том, что при удалении файла освободившееся место в каталоге точно подойдет для нового описателя файла. Тем не менее «кучу» придется все так же «разгребать», и при чтении длинного имени из нее также может возникнуть прерывание из-за отсутствия страницы памяти. Однако имена файлов уже не должны начинаться с границы слов, поэтому символы-заполнители не потребуются.



Реализация каталогов

Во всех рассмотренных схемах при поиске файла каталоги просматриваются линейно сверху вниз. Для очень больших каталогов, содержащих много тысяч файлов, такой поиск может занять довольно много времени. ***Один из способов ускорить поиск файла состоит в использовании хэш-таблицы в каждом каталоге.***

Файловая подсистема ОС UNIX - s5fs

В Unix разделы выступают в качестве независимых устройств, доступ к которым осуществляется как к различным носителям данных.

Файловая система s5fs занимает раздел диска и состоит из трех основных компонентов как показано на рисунке:

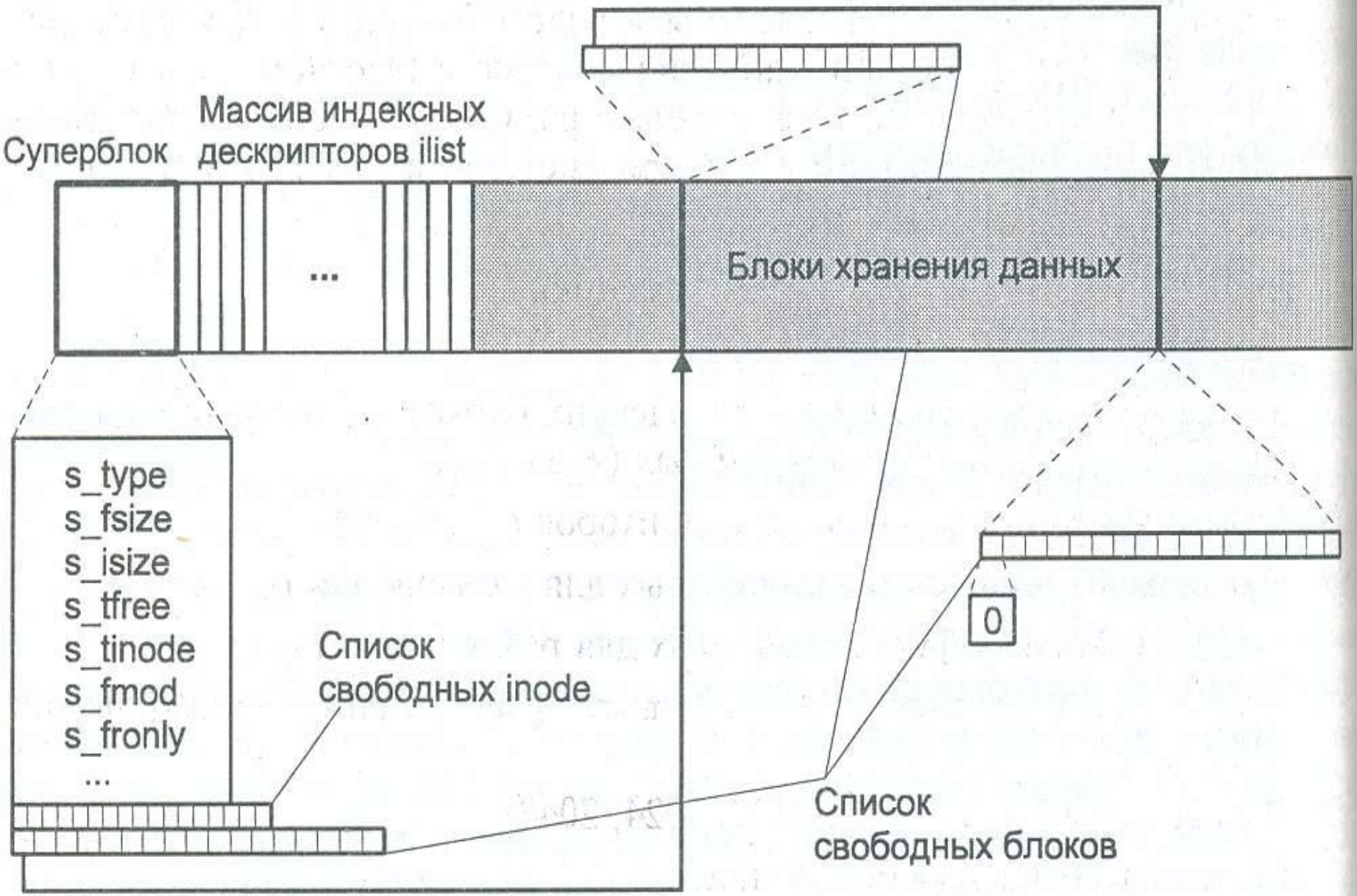


Рис. 4.1. Структура файловой системы s5fs

Файловая подсистема ОС UNIX - s5fs

Суперблок (superblock) – содержит общую информацию о файловой системе, например, об ее архитектуре, общем числе блоков и индексных дескрипторов, или метаданных (inode).

Файловая подсистема ОС UNIX - s5fs

Суперблок содержит следующую информацию:

- ***Тип файловой системы*** (s_type)
- ***Размер файловой системы*** в логических блоках, включая сам суперблок, ilist и блоки хранения данных (s_fsize)
- ***Размер массива индексных дескрипторов*** (s_isize)
- ***Число свободных блоков***, доступных для размещения (s_tfree)

Файловая подсистема ОС UNIX - s5fs

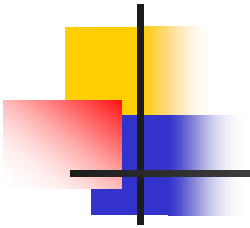
- ***Число свободных inode***, доступных для размещения (s_tinode)
- ***Флаги*** (флаг модификации s_fmod, флаг режима монтирования s_fronly)
- ***Размер логического блока*** (512, 1024, 2048)
- ***Список номеров свободных inode***
- ***Список адресов свободных блоков***

Файловая подсистема ОС UNIX - s5fs

Хранение списка номеров свободных inode и списка адресов свободных блоков организовано следующим образом:

Поскольку число свободных inode и блоков хранения данных может быть значительным, целиком эти два списка в суперблоке не хранят.

Файловая подсистема ОС UNIX - s5fs

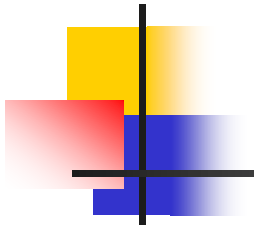


Для индексных дескрипторов хранится только часть списка. Когда количество свободных inode почти = 0, ядро просматривает ilist и вновь формирует список свободных inode. Если поле di_mode индексного дескриптора = 0, то inode свободен.

Файловая подсистема ОС UNIX - s5fs

Такой подход не применим к свободным блокам хранения данных. Поэтому ***хранят адреса свободных блоков.*** ***Список адресов свободных блоков может занимать несколько блоков хранения данных, но суперблок содержит только один блок этого списка. Первый элемент этого блока указывает на блок, хранящий продолжение списка и т.д.***

Файловая подсистема ОС UNIX - s5fs



Выделение свободных блоков для размещения файла производится с конца списка суперблока.

Файловая подсистема ОС UNIX - s5fs

Индексные дескрипторы

Inode содержит информацию о файле, необходимую для обработки данных, т.к. метаданные файла. ***Каждый файл ассоциирован с одним inode.***

Индексный дескриптор не содержит:

- Имя файла; (имя файла содержится в блоках хранения данных каталога)

- Содержимого файла, которое размещено в блоках хранения данных.

Файловая подсистема ОС UNIX - s5fs

При открытиии файла ядро помещает копию дискового inode в память в таблицу in-core inode.

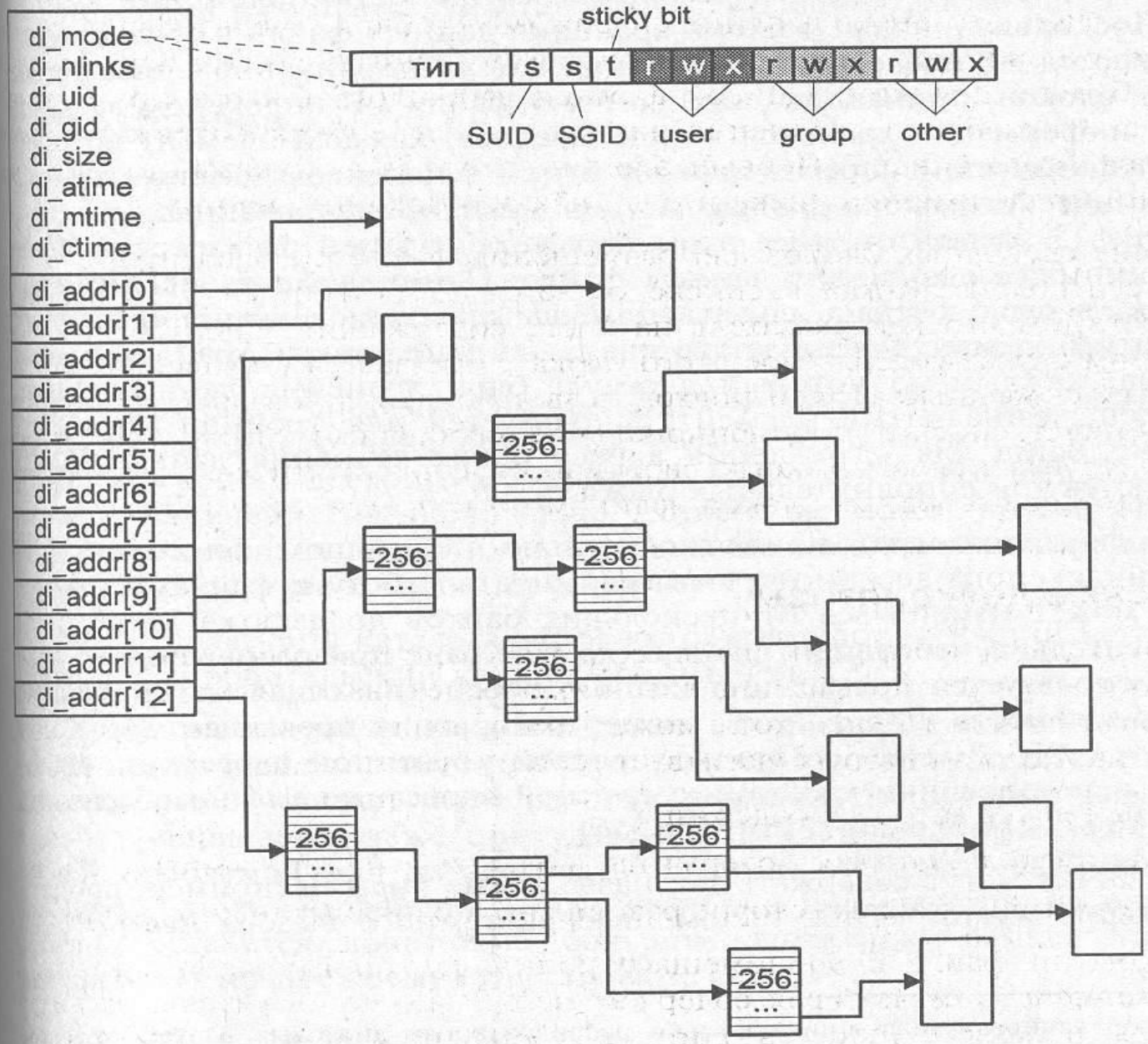
Основные поля дискового inode следующие:

- di_mode – тип файла, дополнительные атрибуты и права доступа.
- di_nlinks – число ссылок на файл, т.е. количество имен, которое имеет файл в ФС
- di_uid, di_gid – идентификаторы владельца-пользователя, и владельца-группы

Файловая подсистема ОС UNIX - s5fs

- di_size – размер файла в байтах.
- di_atime – время последнего доступа к файлу
- di_mtime – время последней модификации
- di_ctime – время последней модификации inode
- di_addr – массив адресов дисковых блоков хранения данных.

di_addr[13] Массив адресов дисковых блоков хранения данных.



Файловая подсистема ОС UNIX - s5fs

Массив индексных дескрипторов (ilist). - Содержит метаданные всех файлов файловой системы. Индексный дескриптор содержит статусную информацию о файле и указывает на расположение данных этого файла. ***Ядро обращается к inode по индексу в массиве ilist. Один inode является корневым (root) inode ФС, через него обеспечивается доступ к структуре каталогов и файлов после монтирования ФС.***

Файловая подсистема ОС UNIX - s5fs

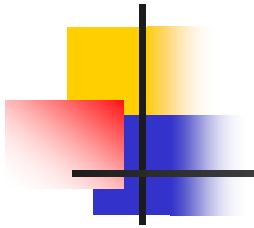
Размер массива ilist является фиксированным и задается при создании ФС.

Т.о., ФС s5fs имеет ограничение по числу файлов, которые могут храниться в ней, независимо от размера этих файлов.

Файловая подсистема ОС UNIX - s5fs

Блоки хранения данных. – Данные обычных файлов и каталогов хранятся в блоках. Обработка файла осуществляется через inode, содержащего ссылки на блоки данных. ***Блоки хранения данных занимают большую часть дискового раздела, и их число определяет максимальный суммарный объем файлов данной файловой системы***. ***Размер блока кратен 512 байтам.***

Файловая подсистема ОС UNIX - s5fs

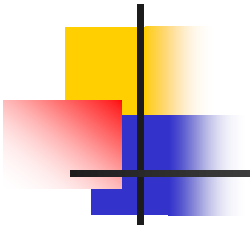


Имя файла хранится в файлах специального типа – каталогах. Такой подход позволяет любому файлу, т.е. данным иметь теоретически неограниченное число имен в ФС.

Файловая подсистема ОС UNIX - s5fs

Каталог файловой системы s5fs **представляет собой таблицу, каждый элемент которой имеет фиксированный размер в 16 байтов: 2 байта хранят номер индексного дескриптора файла, 14 байтов – его имя.**

При удалении имени файла из каталога номер inode соответствующего элемента устанавливается равным 0



Структура ФС MS-DOS

Диск имеет равномерное покрытие, в котором запоминаются данные. DOS располагает данные последовательностями *по 512 байт*, которые называются *секторами*, но в принципе операционная система может организовывать данные на диске как угодно.

Структура ФС MS-DOS

Диск делится на секторы, и соответственно, дорожки делятся на участки. Секторы создаются при форматировании диска.

Структура ФС MS-DOS

Низкоуровневое форматирование определяет секторы, записывая последовательность специальных кодов, говорящих контроллеру, где начинается сектор. Затем записываются специальные идентификационные номера, так что у каждого сектора появляется своя метка.

Структура ФС MS-DOS

Файл – понятие, которое используется DOS для изоляции программ от сложностей размещения данных на дисковой поверхности. Это не что иное, как цепочка секторов, заполненных данными.

Структура ФС MS-DOS

DOS держит на диске списки файлов, называемые *каталогами.* **В них находятся некоторая информация о файле и номер сектора, с которого начинается файл.** Каталог ничего не говорит о размещении файла на диске, кроме адреса его начала. Все остальное приходится на *таблицу размещения файлов*.

Структура ФС MS-DOS

Кластер - единица распределения дискового пространства (1, 2 сектора).

Начальный кластер дает номер первого и, возможно, единственного кластера, отведенного файлу. DOS берет этот номер, преобразует его в соответствующие номера секторов и начинает доступ к файлу.

Структура ФС MS-DOS

Таблица размещения файлов указывает путь к следующему кластеру, занимаемому файлом.

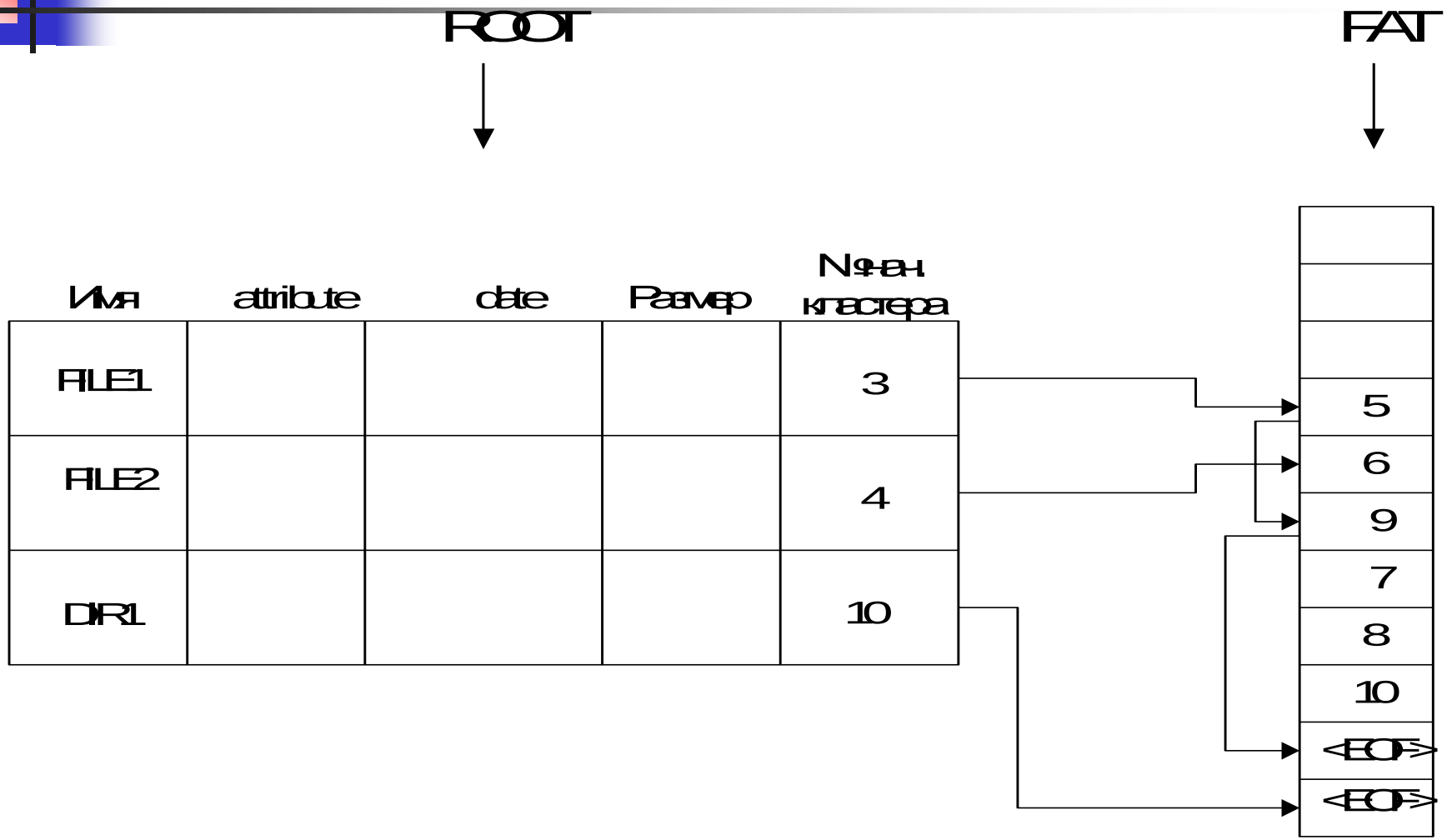
Структура ФС MS-DOS

Из каталога можно узнать только начальный кластер файла. Но большинство файлов занимает свыше одного кластера. **При форматировании диска, DOS создает таблицу размещения файлов (FAT – file allocation table).** ***DOS содержит на диске копию FAT***

Структура ФС MS-DOS

FAT – представляет собой таблицу чисел, причем каждая позиция в таблице соотносится с кластером дискового пространства. Каждая позиция содержит число – номер следующего кластера.

Структура ФС MS-DOS



Структура ФС MS-DOS

В процессе работы из-за ошибок пользователя, сбоев в работе программ, сбоев электропитания, действий вирусов и т.д. возможно повреждение данных на дисках. ***Можно выделить следующие группы повреждений:***

- Ошибочное удаление файла;

- Повреждение файлов на диске;

- Порча системных областей диска;

Структура ФС NTFS

Структура NTFS начинается с понятия тома.

Том (volume) соответствует логическому разделу на диске и создается при форматировании диска или его части под NTFS.

Структура ФС NTFS

Том состоит из набора файлов и оставшегося свободного пространства.

В томе NTFS все данные файловой системы вроде битовых карт, каталогов и даже начального загрузочного кода хранятся как обычные файлы.

Структура ФС NTFS

Размер кластера на томе NTFS, или кластерный множитель (cluster factor), устанавливается при форматировании. Размер кластера по умолчанию определяется размером тома, но всегда содержит целое число физических секторов с дискретностью n2 (1 сектор, 2 сектора, 4 сектора…). Кластерный множитель выражается числом байтов в кластере, например 512 байт, 1Кб, 4Кб.

Структура ФС NTFS

Внутренне NTFS работает только с кластерами. **NTFS использует кластер как единицу выделения пространства для поддержания независимости от размера физического сектора.** Это позволяет работать эффективно с очень большими дисками, используя кластеры большего размера, и поддерживать диски с размером секторов отличным от 512 байтов.

Структура ФС NTFS

Применение больших кластеров на больших томах уменьшает фрагментацию и ускоряет выделение свободного пространства за счет небольшого проигрыша в эффективности использования дискового пространства.

Структура ФС NTFS

NTFS адресуется к конкретным местам на диске, используя логические номера кластеров (Logical cluster numbers, LCN).

Структура ФС NTFS

Для этого все кластеры на томе просто нумеруются по порядку – от начала до конца. Для преобразования LCN в физический адрес на диске NTFS умножает LCN на кластерный множитель и получает байтовое смещение от начала тома, воспринимаемое интерфейсом драйвера диска.

Структура ФС NTFS

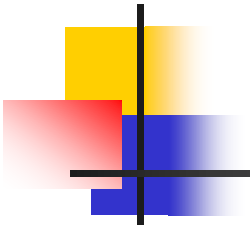
На данные внутри файла NTFS ссылается по виртуальным номерам кластеров (virtual cluster numbers, VCN), нумеруя кластеры которые принадлежат конкретному файлу (от 1 до m). VCN не обязательно должны быть физически непрерывными.

Структура ФС NTFS

В файловой системе NTFS все атрибуты файлов (имя, размер, расположение *экстентов* файла на диске и т.д.) хранятся в скрытом системном файле $MFT.

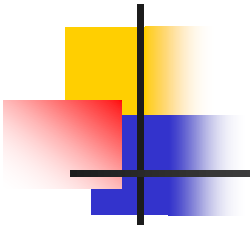
Экстенты

Во многих файловых системах, в общем случае файл хранится в виде «заголовка», то есть некой относительно небольшой структуры данных (например, inode или строки Master File Table в NTFS), который содержит указатели на участки носителя информации, где по кускам хранится содержимое файла.



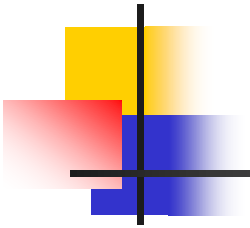
Экстенты

В традиционных файловых системах, это указатели на отдельные блоки (минимальные участки носителя, который можно прочесть или записать за раз). В **ряде современных файловых систем используются указатели не на блоки, а на экстенты.**



Экстенты

Использование указателей на экстенты имеет ряд преимуществ над схемой с указателями на отдельные блоки. **Поскольку все данные в одном экстенте расположены на диске подряд, повышается скорость чтения и записи файла и понижается степень фрагментации дискового пространства.**



Экстенты

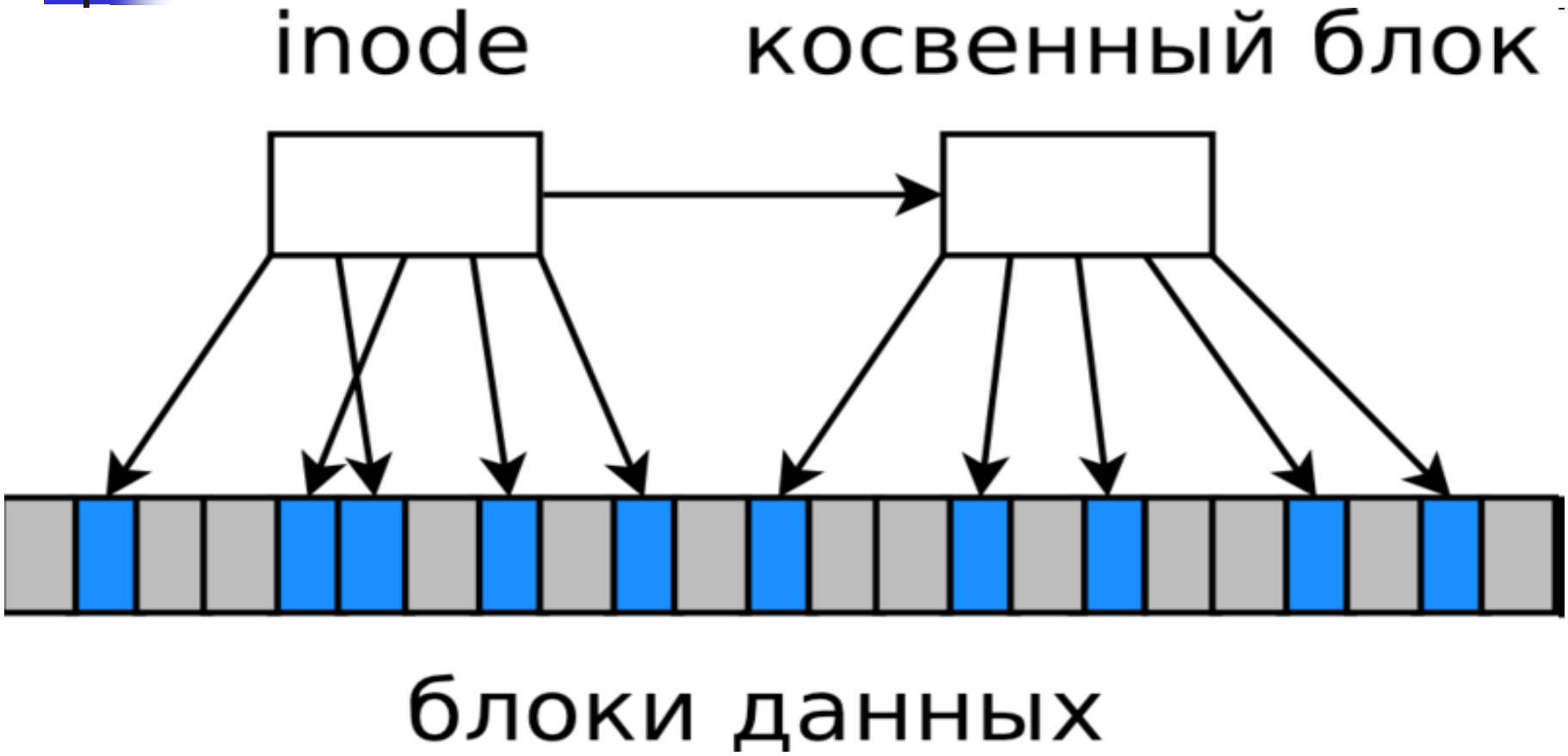
При одинаковом размере и организации структуры данных «заголовка» файла, файловая система с поддержкой экстентов будет иметь больший максимальный размер файлов.

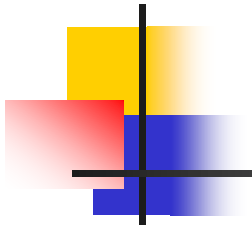
Экстенты

Главный недостаток экстентов — повышается сложность реализации файловой системы.

Скорость доступа к файлу также можно повысить, если заранее зарезервировать под файл по возможности непрерывный участок на диске

Файл в файловой системе без поддержки экстентов

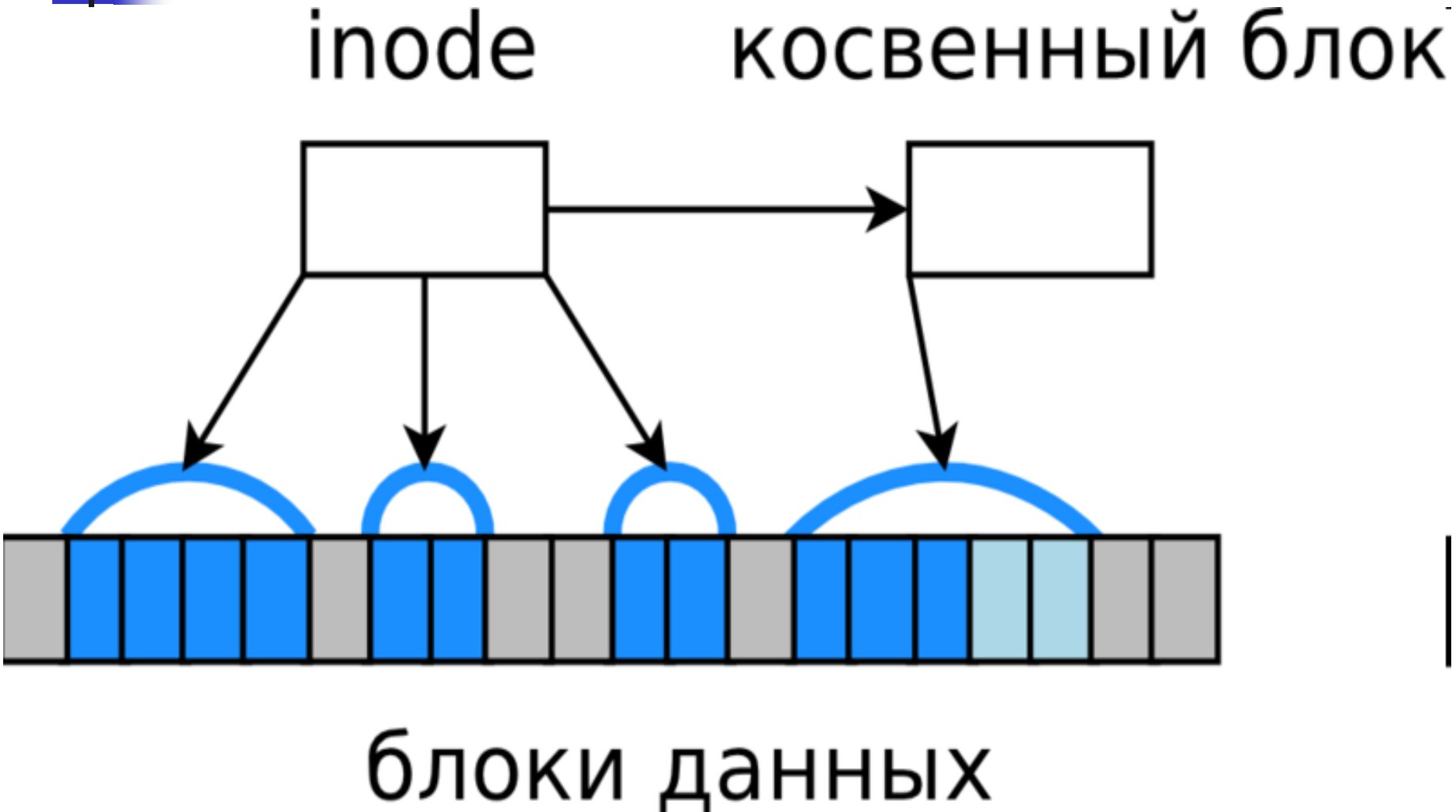


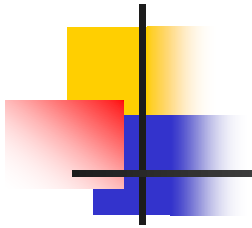


Экстенты

- Синие блоки используются под содержимое файла
- Серые блоки не используются под содержимое файла

Файл в файловой системе, которая поддерживает экстенты





Экстенты

- Синие блоки используются под содержимое файла
- Голубые блоки распределены в один из экстентов файла, но пока не используются
- Серые блоки не входят в экстенты файла

Структура ФС NTFS

На хранение информации о каждом файле (и каталоге) в $MFT (***MFT*** *(Master File Table) — главная файловая таблица (база данных, в которой хранится информация о содержимом тома NTFS, представляющая собой таблицу, строки которой соответствуют файлам тома, а столбцы — атрибутам файлов).*) отводится от одного до нескольких Кбайт.

Структура ФС NTFS

При большом количестве файлов, хранящихся на диске, объем файла $MFT может достигать десятков или даже сотен Мбайт.

Структура ФС NTFS

Файлы небольшого размера (порядка сотен байт) хранятся непосредственно в $MFT, что существенно ускоряет доступ к ним.

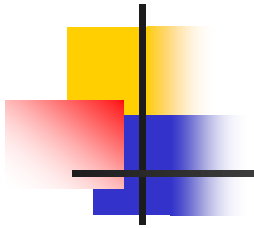
Структура ФС NTFS

файл $MFT обычно располагается ближе к середине диска, разрушение первых дорожек диска NTFS не приводит к таким фатальным последствиям, как разрушение начальных областей диска FAT!!!

Структура раздела

NTFS делит все полезное место на кластеры - блоки данных, используемые единовременно. NTFS поддерживает почти любые размеры кластеров - от 512 байт до 64 Кбайт, **неким стандартом же считается кластер размером 4 Кбайт.** Никаких аномалий кластерной структуры NTFS не имеет.

Структура раздела

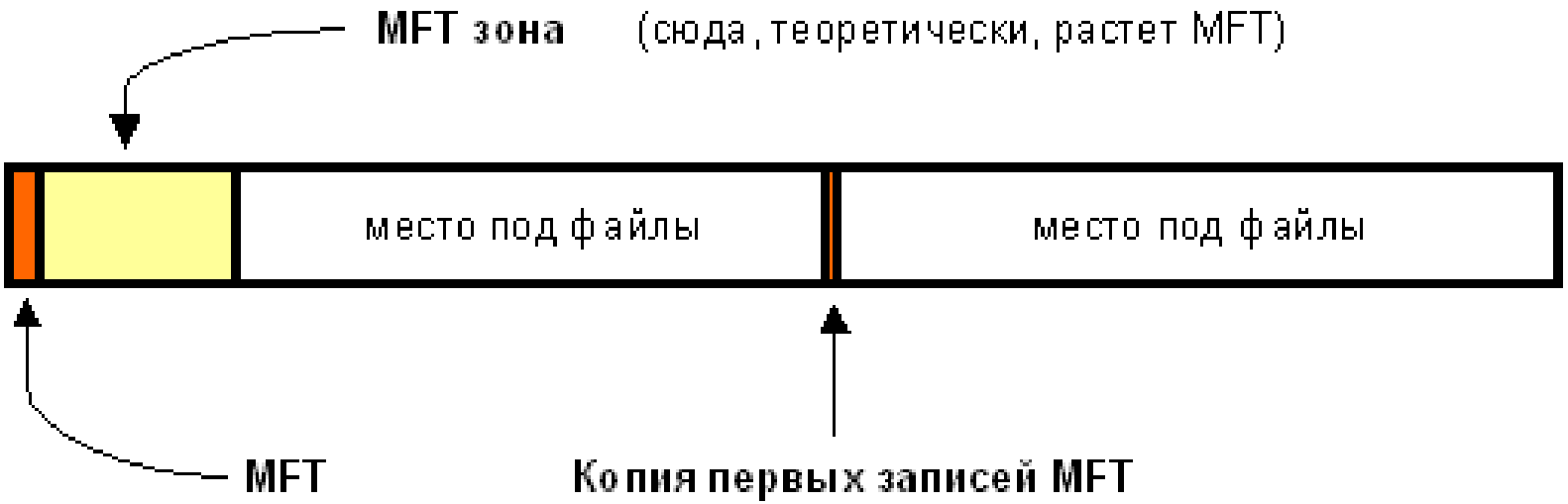


Диск NTFS условно делится на две части. Первые 12% диска отводятся под так называемую MFT зону - пространство, в которое растет метафайл MFT.

Структура раздела

Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой - это делается для того, чтобы самый главный, служебный файл (MFT) не фрагментировался при своем росте. Остальные 88% диска представляют собой обычное пространство для хранения файлов.

Структура диска

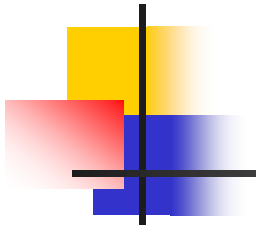


Структура раздела

Свободное место диска, включает в себя всё физически свободное место - незаполненные куски MFT-зоны туда тоже включаются.

Структура раздела

Механизм использования MFT-зоны таков: когда файлы уже нельзя записывать в обычное пространство, MFT-зона просто сокращается (в текущих версиях операционных систем ровно в два раза), освобождая таким образом место для записи файлов.



Структура раздела

Метафайл MFT может фрагментироваться, хоть это и было бы нежелательно.

MFT и его структура

Каждый элемент системы представляет собой файл - даже служебная информация. Самый главный файл на NTFS называется MFT, или Master File Table - общая таблица файлов. Именно он размещается в MFT зоне и представляет собой централизованный каталог всех остальных файлов диска, и себя самого.

MFT и его структура

MFT поделен на записи фиксированного размера (обычно 1 Кбайт), и каждая запись соответствует, какому либо файлу.

MFT и его структура

Первые 16 файлов носят служебный характер и недоступны операционной системе - они называются метафайлами, самый первый метафайл - сам MFT. Эти первые 16 элементов MFT - единственная часть диска, имеющая фиксированное положение.

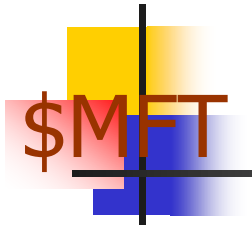
MFT и его структура

Вторая копия первых трех записей, для надежности хранится посередине диска. **Остальной MFT-файл может располагаться, как и любой другой файл, в произвольных местах диска - восстановить его положение можно с помощью его самого, "зацепившись" за самую основу - за первый элемент MFT.**

Метафайлы

Метафайлы находятся корневом каталоге NTFS диска - они начинаются с символа имени "$"

Метафайлы и их назначение



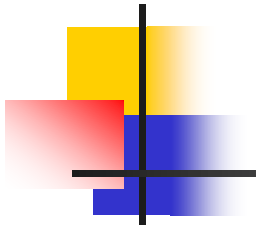
$MFT - сам MFT;

$MFTmirr - копия первых 16 записей MFT, размещенная посередине диска;

$LogFile - файл поддержки журналирования;

$Volume - служебная информация - метка тома, версия файловой системы, т.д.

Метафайлы и их назначение



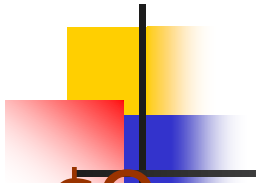
$AttrDef - Список стандартных атрибутов файлов на томе;

$. - корневой каталог;

$Bitmap - карта свободного места тома;

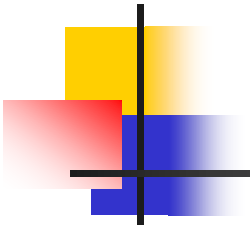
$Boot - загрузочный сектор (если раздел загрузочный);

Метафайлы и их назначение



$Quota - файл, в котором записаны права пользователей на использование дискового пространства (начал работать лишь в NT5);

Метафайлы и их назначение



$Upcase - файл - таблица соответствия заглавных и прописных букв в имен файлов на текущем томе. Нужен в основном потому, что в NTFS имена файлов записываются в Unicode, что составляет 65 тысяч различных символов, искать большие и малые эквиваленты которых очень нетривиально.

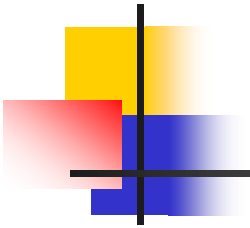
Файлы и потоки

Итак, у системы есть файлы :

- Обязательный элемент - запись в MFT. Здесь хранится вся информация о файле, за исключением собственно данных. Имя файла, размер, положение на диске отдельных фрагментов, и т.д. Если для информации не хватает одной записи MFT, то используются несколько, причем не обязательно подряд.

Файлы и потоки

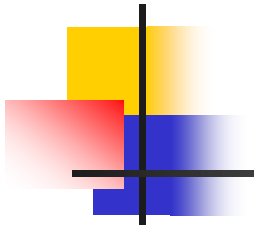
- Опциональный элемент - потоки данных файла. Файл может не иметь данных - в таком случае на него не расходуется свободное место самого диска. Файл может иметь не очень большой размер. Тогда идет в ход довольно удачное решение: данные файла хранятся прямо в MFT, в оставшемся от основных данных месте в пределах одной записи MFT. Файлы, занимающие сотни байт, обычно не имеют своего "физического" воплощения в основной файловой области - все данные такого файла хранятся в одном месте - в MFT.



Файлы и потоки

Каждый файл в NTFS имеет несколько абстрактное строение - у него нет как таковых данных, а есть потоки (streams).

Один из потоков и носит привычный нам смысл - данные файла. Но большинство атрибутов файла - тоже потоки



Файлы и потоки

Базовая сущность у файла только одна - номер в MFT, а всё остальное опционально.

Файлы и потоки

Дополнительные потоки не видны стандартными средствами: наблюдаемый размер файла - это лишь размер основного потока, который содержит традиционные данные.

Файлы и потоки

Имя файла может содержать любые символы, включая полый набор национальных алфавитов, так как данные представлены в Unicode - 16-битном представлении, которое дает 65535 разных символов. Максимальная длина имени файла - 255 символов.

Каталоги

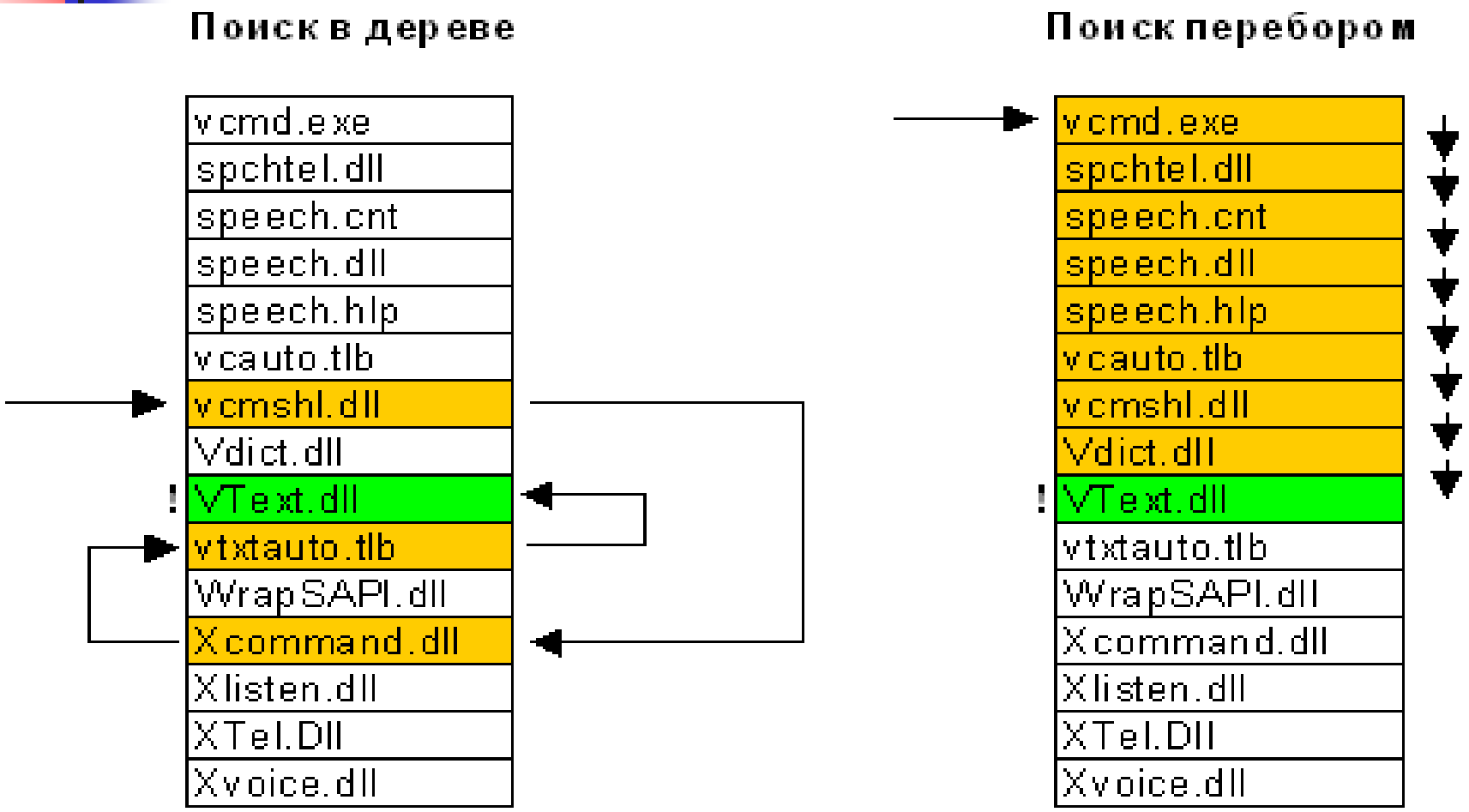
Каталог на NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске.

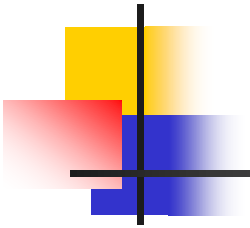
Каталоги

Файл каталога поделен на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога.

Внутренняя структура каталога представляет собой бинарное дерево.

Каталоги


Поиск в дереве
vcmd.exe
spchtel.dll
speech.cnt
speech.dll
speech.hlp
vcauto.tlb
vcmshl.dll
Vdict.dll
VText.dll
vtxtauto.tlb
WrapSAPI.dll
Xcommand.dll
Xlisten.dll
XTel.Dll
Xvoice.dll
Поиск перебором
vcmd.exe
spchtel.dll
speech.cnt
speech.dll
speech.hlp
vcauto.tlb
vcmshl.dll
Vdict.dll
VText.dll
vtxtauto.tlb
WrapSAPI.dll
Xcommand.dll
Xlisten.dll
XTel.Dll
Xvoice.dll

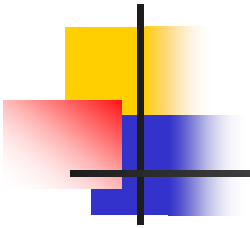


Каталоги

Какую информацию можно получить, просто прочитав файл каталога? Ту, что выдает команда dir. Для выполнения простейшей навигации по диску не нужно искать в MFT каждый файл, надо лишь читать самую общую информацию о файлах из файлов каталогов. **Главный каталог диска - корневой - ничем не отличается об обычных каталогов, кроме специальной ссылки на него из начала метафайла MFT.**

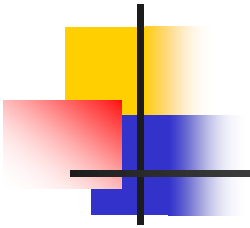
Журналирование

NTFS - отказоустойчивая система, которая вполне может привести себя в корректное состояние при практически любых реальных сбоях. Любая современная файловая система основана на таком понятии, как **транзакция - действие, совершаемое целиком и корректно или не совершаемое вообще.**



Журналирование

У NTFS просто не бывает промежуточных (ошибочных или некорректных) состояний - квант изменения данных не может быть поделен на до и после сбоя, принося разрушения и путаницу - он либо совершен, либо отменен.

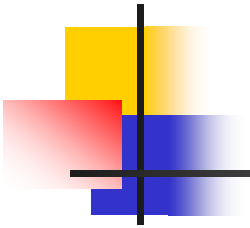


Журналирование

Журналирование не должно помешать работе файловой системы. Первый логичный шаг - **отменить полное журналирование** как абсолютно неприемлемое с точки зрения быстродействия.

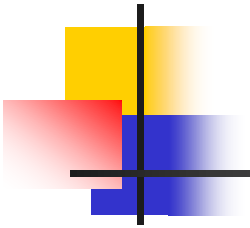
Журналирование

В NTFS применяется журналирование логических структур, а не данных пользователя - отсюда и идет фраза, что сохранность данных не гарантируется, но тем не менее корректное состояние самой системы будет поддерживаться.



Журналирование

Операции, которые тем не менее журналируются системой - это операции со структурами самой системы, то есть с файлами и каталогами: добавление файлов, переименование, перенос, создание и удаление (освобождение свободного места). Журналируются также и операции дефрагментации - то есть перемещения фрагментов файлов. Одним словом, все логические операции журналированы.

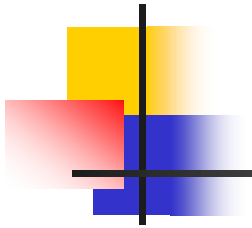


Журналирование

Пример 1: осуществляется запись данных на диск. Вдруг выясняется, что в то место, куда мы только что решили записать очередную порцию данных, запись не удалась - физическое повреждение поверхности. Поведение NTFS в этом случае довольно логично: транзакция записи откатывается целиком - система осознает, что запись не произведена. Пространство помечается как сбойное, а данные записываются в другое место - начинается новая транзакция.

Журналирование

Пример 2: более сложный случай - идет запись данных на диск. Вдруг, - отключается питание и система перезагружается. На какой фазе остановилась запись, где есть данные? На помощь приходит другой ***механизм системы - журнал транзакций. Система, уловив операцию записи на диск, пометила в метафайле $LogFile это состояние.*** При перезагрузке это файл изучается на предмет наличия незавершенных транзакций, которые были прерваны аварийной ситуацией и результат которых непредсказуем - все эти транзакции отменяются



Журналирование

Полный undo-файл, способный откатить абсолютно все операции - невозможен.

Достоинства NTFS

- Фрагментация файлов не имеет практически никаких последствий для самой файловой системы - работа фрагментированной системы ухудшается только с точки зрения доступа к самим данным файлов.

- Сложность структуры каталогов и число файлов в одном каталоге также не создает особых препятствий быстродействию.

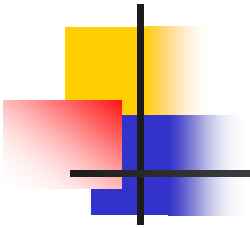
Достоинства NTFS

- Быстрый доступ к произвольному фрагменту файла (например, редактирование больших *.wav файлов).

- Очень быстрый доступ к маленьким файлам (несколько сотен байт) - весь файл находится в том же месте, где и системные данные (запись MFT).

Недостатки NTFS

- Существенные требования к оперативной памяти системы (64 МБ - абсолютный минимум, лучше - больше).
- Медленные диски и контроллеры сильно снижают быстродействие NTFS.



Недостатки NTFS

- Работа с каталогами средних размеров затруднена тем, что они почти всегда фрагментированы.

- Диск, долго работающий в заполненном на 80% - 90% состоянии, будет показывать крайне низкое быстродействие.

Защита на уровне файловых систем

NTFS

* Безопасность на уровне файлов и папок
* Сжатие дисков
* Дисковые квоты
* Шифрование файлов -

* Поддержка мультизагрузочной конфигурации
* Невозможно ограничить доступ на уровне файлов

Защита на уровне файловых систем
NTFS и FAT/FAT32

1) Безопасность на уровне файлов и папок. Файловая система NTFS позволяет управлять доступом к файлам и папкам.

Защита на уровне файловых систем
NTFS и FAT/FAT32

Дисковые квоты. Файловая система NTFS позволяет ограничивать использование дискового пространства для конкретного пользователя.

Защита на уровне файловых систем
NTFS и FAT/FAT32

Шифрование. Файловая система **NTFS** позволяет шифровать данные на физическом диске, используя систему шифрования Microsoft (EFS).

Защита на уровне файловых систем
NTFS и FAT/FAT32

Преобразование тома FAT или FAT32 в файловую систему NTFS без форматирования:

CONVERT volume /FS:NTFS [/V] [/CvtArea:filename] [/NoSecurity] [/X]

Особенности файловых систем NTFS и FAT/FAT32

/NoSecurity - Изменяет параметры защиты преобразованных файлов и каталогов, разрешая доступ к ним всем пользователям

Обеспечение безопасности ресурсов с помощью разрешений NTFS

Разрешения NTFS позволяют явно указать, какие пользователи и группы имеют доступ к файлам и папкам и какие операции с содержимым этих файлов или папок им разрешено выполнять.

Разрешения NTFS применимы только к томам, отформатированным с использованием файловой системы NTFS. Они не предусмотрены для томов, использующих файловые системы FAT или FAT32.

Разрешения для папок в NTFS

Чтение (Read) - Просматривать файлы и папки, а также список владельцев разрешения и атрибуты папки

Запись (Write) - Создавать новые файлы и папки внутри папки, изменять атрибуты папки и просматривать владельцев и разрешения для папки

Разрешения для папок и файлов в NTFS

Список содержимого папки (List Folder Contents) - Просматривать имена файлов и папок.

Чтение и выполнение (Read & Execute) – Перемешаться по структуре папок в поисках других файлов или папок, даже если пользователь не обладает разрешением на доступ к просматриваемым папкам. Выполнять все действия, право на которые дают разрешения Чтение (Read) и Список содержимого папки (List Folder Contents)

Разрешения для папок и файлов в NTFS

Изменить (Modify) - Удалять папки и выполнять все действия, право на которые дают разрешения Запись (Write) и Чтение и выполнение (Read & Execute)

Разрешения для папок и файлов в NTFS

Полный доступ (Full Control) - Изменять разрешения, менять владельца, удалять папки и файлы и выполнять действия, право на которые дают все остальные разрешения NTFS для папок

Разрешения для папок и файлов в NTFS

В NTFS хранится *список управления доступом* (access control list, ACL) для каждого файла и папки на томе NTFS.

В этом списке перечислены пользователи и группы, для которых установлены разрешения для файла или папки, а также сами назначенные разрешения.

Разрешения для папок в NTFS

Чтобы пользователь получил доступ к ресурсу, ***в ACL должна быть запись, называемая элемент списка управления доступом (access control entry. ACE),*** для этого пользователя или группы, к которой он принадлежит. Эта запись назначит запрашиваемый тип доступа

Множественные разрешения NTFS

Эффективные разрешения пользователя для ресурса — это совокупность разрешений NTFS, которые назначаются отдельному пользователю и всем группам, к которым он принадлежит. *Если у пользователя есть разрешение* ***Чтение*** *(Read) для папки и он входит в группу, у которой есть разрешение* ***Запись*** *(Write) для той же папки, значит, у этого пользователя есть оба разрешения.*

Множественные разрешения NTFS

1) Приоритет разрешений для файлов над разрешениями для папок:

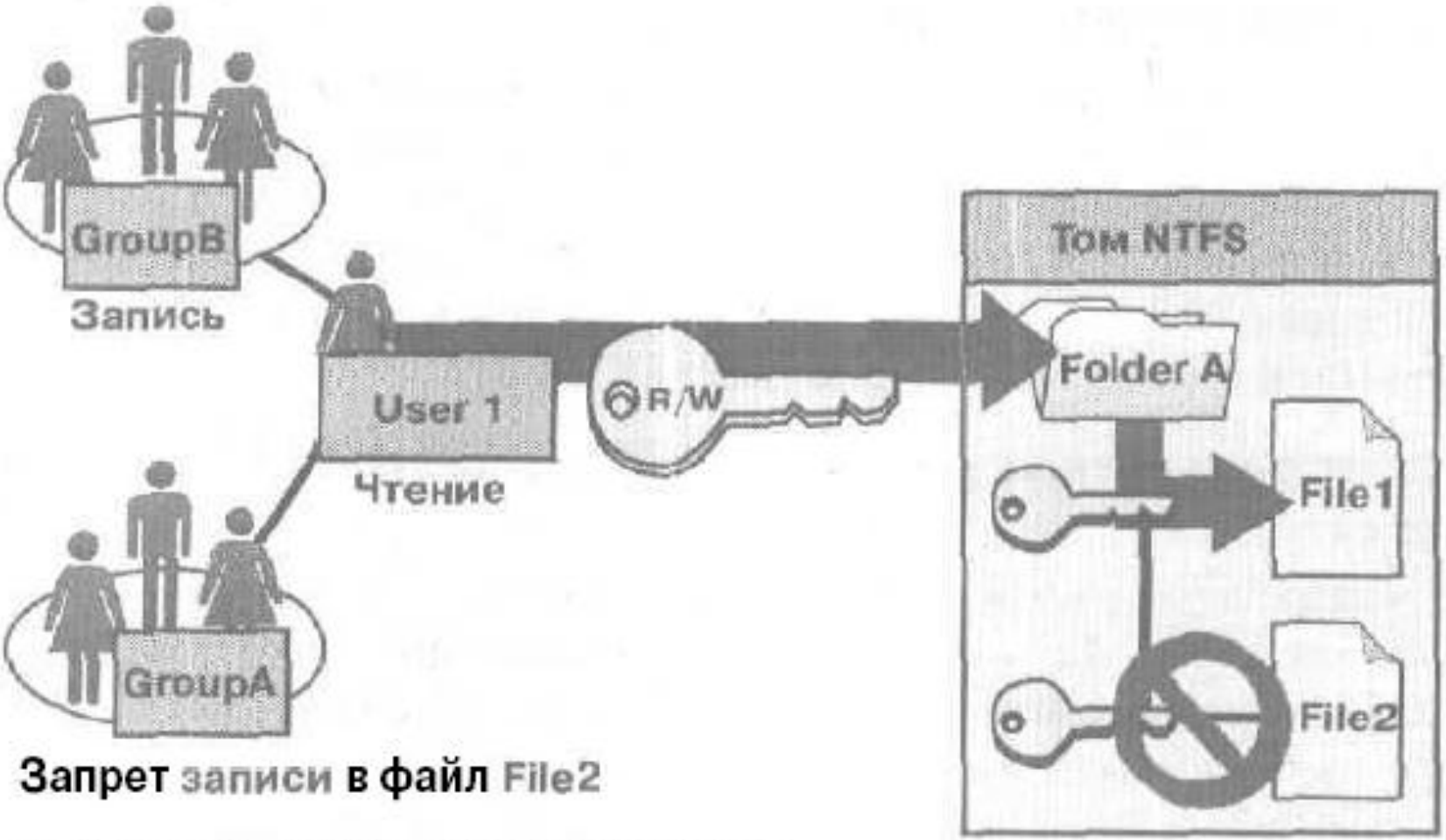
В NTFS разрешения для файлов имеют больший приоритет, чем разрешения для папок. Если у вас есть разрешение на доступ к файлу и **право Обход перекрестной проверки (Bypass Traverse Checking),** то вы сможете воспользоваться доступом к этому файлу, даже если у вас нет доступа к папке, в которой содержится файл.

Множественные разрешения NTFS

2) Приоритет запрещения над разрешениями

Можно запретить доступ к файлу пользователю или группе, хотя этот метод контроля ресурсов нельзя назвать предпочтительным. ***Запрет имеет больший приоритет, чем разрешение на всех уровнях***, на которые он распространяется. Даже если у группы в которую входит пользователь имеется разрешение на доступ к файлу или папке, запрет на доступ для пользователя блокирует все имеющиеся разрешения.

Множественные разрешения NTFS



Запрет записи в файл File2

- Разрешения NTFS суммируются
- Разрешения для файлов перезаписывают разрешения АЛЯ папок
- Запрет имеет более высокий приоритет

Множественные разрешения NTFS

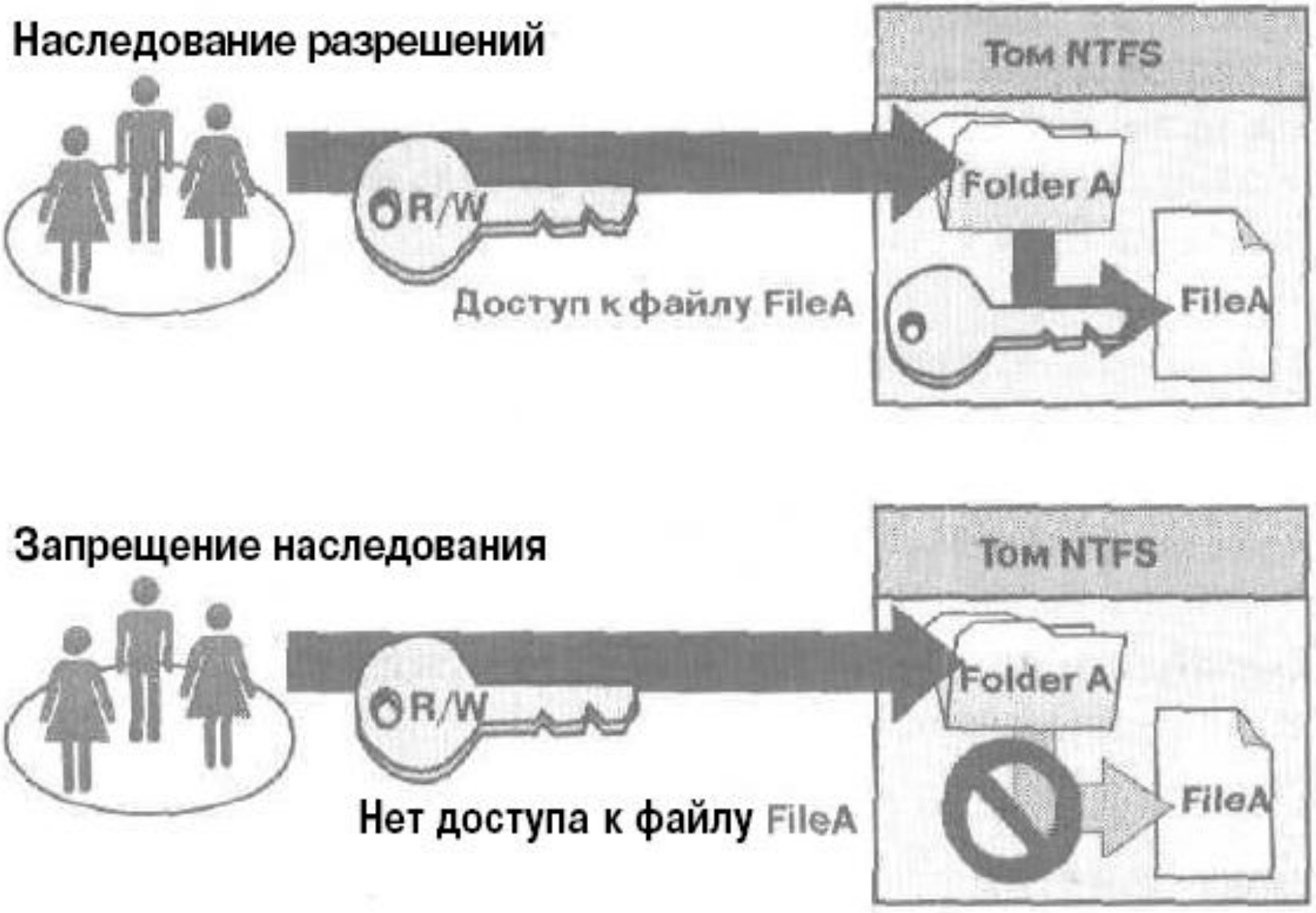
Пользователь может читать и записывать данные в файл Filel, а также читать данные из файла File2, но он не может записывать данные в файл File2, так как он является членом группы Group A, для которой было запрещено разрешение **Запись** (Write) для файла File2.

Множественные разрешения NTFS

3) Наследование разрешений в NTFS

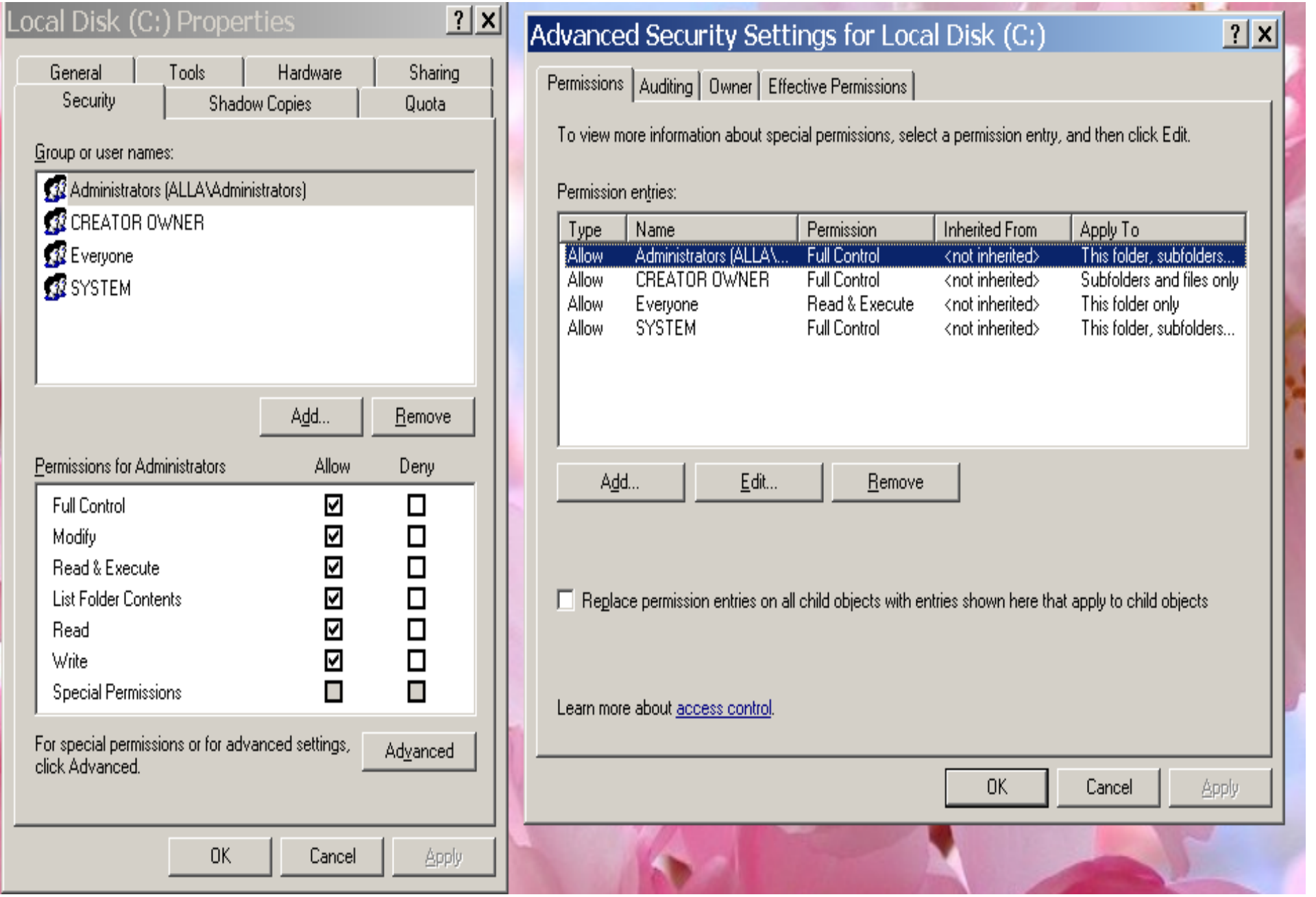
По умолчанию разрешения, назначаемые родительской папке, наследуются и распространяются на подпапки и файлы, содержащиеся в родительской папке. Однако можно предотвратить наследование разрешений.

Наследование



Наследование

Для предотвращения наследования подпапками и файлами разрешений, установленных для родительской папки, снимите флажок **Наследовать от родительского объекта применимые к дочерним объектам разрешения, добавляя их к явно заданным в этом окне** (Inherit From Parent The Permission Entries That Apply To Child) в диалоговом окне **Дополнительные параметры безопасности** (Advanced Security Settings).

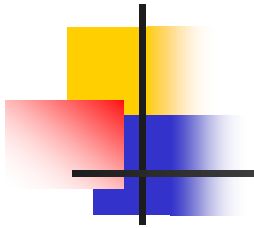

Local Disk (C:) Properties
General
Tools
Hardware
Sharing
Security
Shadow Copies
Quota
Group or user names:
Administrators (ALLA\Administrators)
CREATOR OWNER
Everyone
SYSTEM
Add...
Remove
Permissions for Administrators
Allow
Deny
Full Control
Modify
Read & Execute
List Folder Contents
Read
Write
Special Permissions
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply
Advanced Security Settings for Local Disk (C:)
Permissions
Auditing
Owner
Effective Permissions
To view more information about special permissions, select a permission entry, and then click Edit.
Permission entries:
Type
Name
Permission
Inherited From
Apply To
Allow
Administrators (ALLA\...
Full Control
<not inherited>
This folder, subfolders...
Allow
CREATOR OWNER
Full Control
<not inherited>
Subfolders and files only
Allow
Everyone
Read & Execute
<not inherited>
This folder only
Allow
SYSTEM
Full Control
<not inherited>
This folder, subfolders...
Add...
Edit...
Remove
Replace permission entries on all child objects with entries shown here that apply to child objects
Learn more about access control.
OK
Cancel
Apply

Управление дисковыми квотами

Дисковые квоты используются для управления объемом хранимых данных в распределенных средах.

Дисковые квоты позволяют распределять дисковое пространство между пользователями в зависимости от того, владельцами каких файлов и папок они являются.

Управление дисковыми квотами



Windows XP Professional учитывает дисковые квоты для каждого тома, даже если эти тома расположены на одном и том же жестком диске. ***Поскольку квоты отслеживаются для пользователя, дисковое пространство, занятое файлами пользователя, вычисляется независимо от того, в каких папках пользователь хранит эти файлы.***

Характеристики дисковых квот и их описание

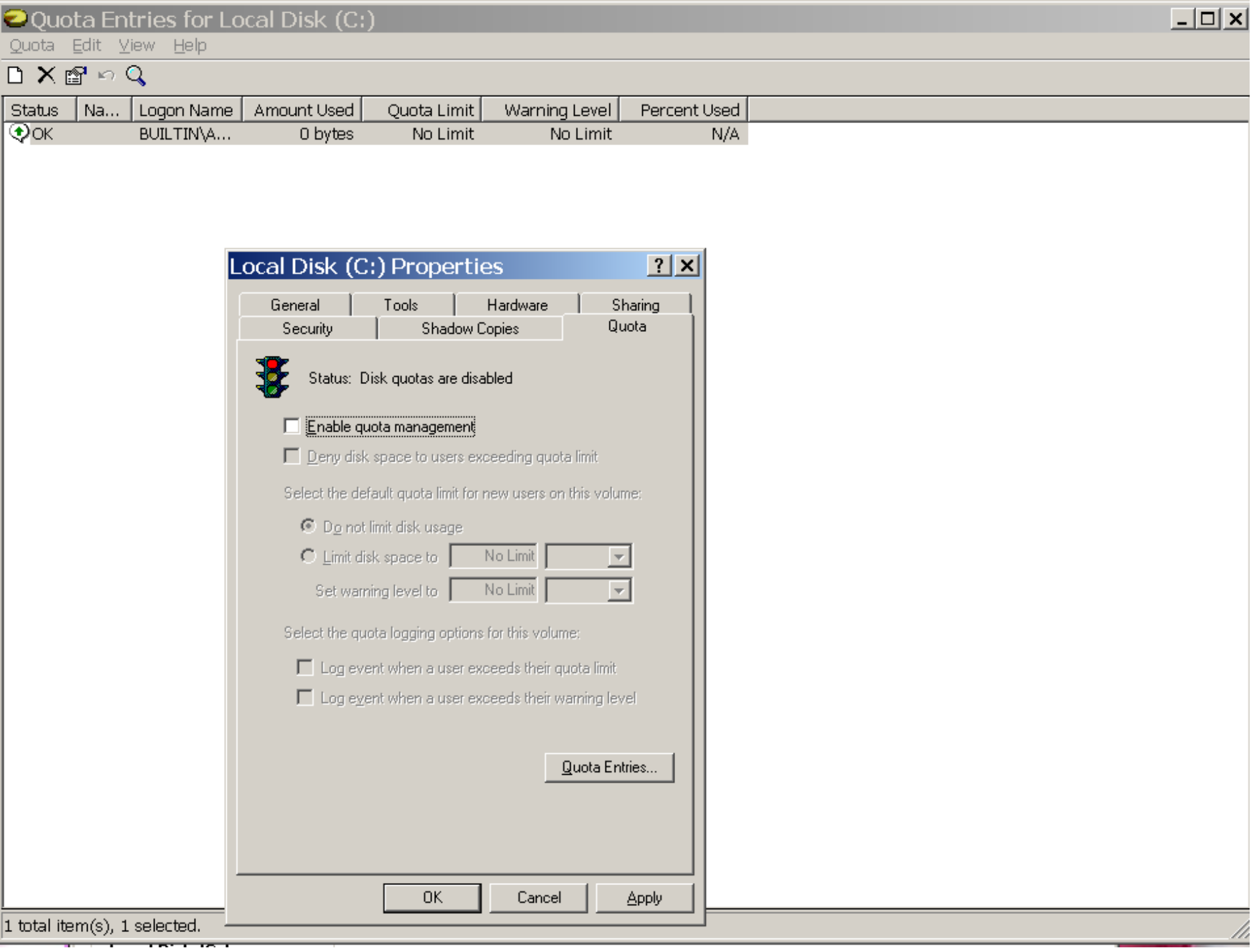
1) Используемый объем диска вычисляется на основе владения файлами - Когда пользователь копирует или сохраняет файл на томе NTFS либо становится владельцем файла на томе NTFS, Windows **XP** Professional берет необходимое для размещения файла пространство из выделенной квоты

Характеристики дисковых квот и их описание

2) Дисковые квоты не учитывают сжатие - Пользователь отвечает за каждый распакованный байт независимо от того, какой объем дискового пространства используется в действительности. Одна из причин этого заключается в том, что степень сжатия зависит от типа файлов.

Характеристики дисковых квот и их описание

3) ***Свободное пространство для приложений зависит от предела квоты*** – выделяется остаток дисковой квоты.

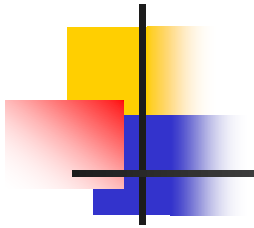

Quota Entries for Local Disk (C:)
Quota Edit View Help
Status Na... Logon Name Amount Used Quota Limit Warning Level Percent Used
OK BUILTIN\A... 0 bytes No Limit No Limit N/A
Local Disk (C:) Properties
General Tools Hardware Sharing
Security Shadow Copies Quota
Status: Disk quotas are disabled
Enable quota management
Deny disk space to users exceeding quota limit
Select the default quota limit for new users on this volume:
Do not limit disk usage
Limit disk space to No Limit
Set warning level to No Limit
Select the quota logging options for this volume:
Log event when a user exceeds their quota limit
Log event when a user exceeds their warning level
Quota Entries...
OK Cancel Apply
1 total item(s), 1 selected.

Контроль дисковых квот

С помощью диалогового окна *Записи квот (Quota Entries For)* можно просмотреть следующую информацию:

- ***объем дискового пространства, используемый каждым из пользователей;***
- ***значения порога предупреждения и предела квоты для каждого пользователя;***
- **пользователи, превысившие порог предупреждения;**
- ***пользователи, превысившие предел квоты*.**

Повышение безопасности с помощью EFS



Файловая система Microsoft Encrypting File System (EFS) предоставляет возможность шифрации данных, хранящихся на дисках NTFS.

Повышение безопасности с помощью EFS

Encrypting File System (EFS) –не поддерживается в "домашних" версиях (Windows XP Home Edition, Windows Vista Basic и Windows Vista Home Premium).

Повышение безопасности с помощью EFS

EFS использует симметричное шифрование для защиты файлов, а также ассиметричное шифрование для защиты случайно-сгенерированного ключа шифрования для каждого файла.

Повышение безопасности с помощью EFS

1) Каждый файл шифруется с помощью симметричного алгоритма шифрования (более быстрый способ шифрования).

   Используется случайно-сгенерированный ключ для каждого файла, называемый File Encryption Key (FEK),

Повышение безопасности с помощью EFS

2) FEK - (случайный для каждого файла ключ симметричного шифрования) защищается путём ассиметричного шифрования на базе алгоритма RSA.

Повышение безопасности с помощью EFS

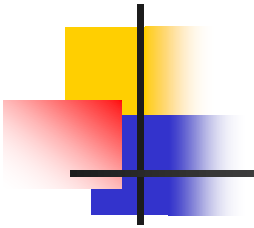
EFS не шифрует файлы, передаваемые по сети, поэтому для защиты передаваемых данных необходимо использовать другие протоколы защиты данных (IPSec или WebDAV

Повышение безопасности с помощью EFS

Алгоритмы шифрования используемые EFS:

Windows 2000	- DESX (none)
Windows XP RTM	- DESX (3DES)
Windows XP SP1	- AES (3DES, DESX)
Windows Server 2003	- AES (3DES, DESX)
Windows Vista	- AES (3DES, DESX)
Windows Server 2008	- AES (3DES, DESX)

Алгоритм Triple DES



Triple DES (3DES) — симметричный блочный шифр, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES.

Размер ключа: 112 (2TDES) или 168 bits (3TDES)

Размер блока: 64 бит

Число раундов: 48DES-эквивалентных раундов

Тип: Сеть Фейстеля

Схема Triple DES

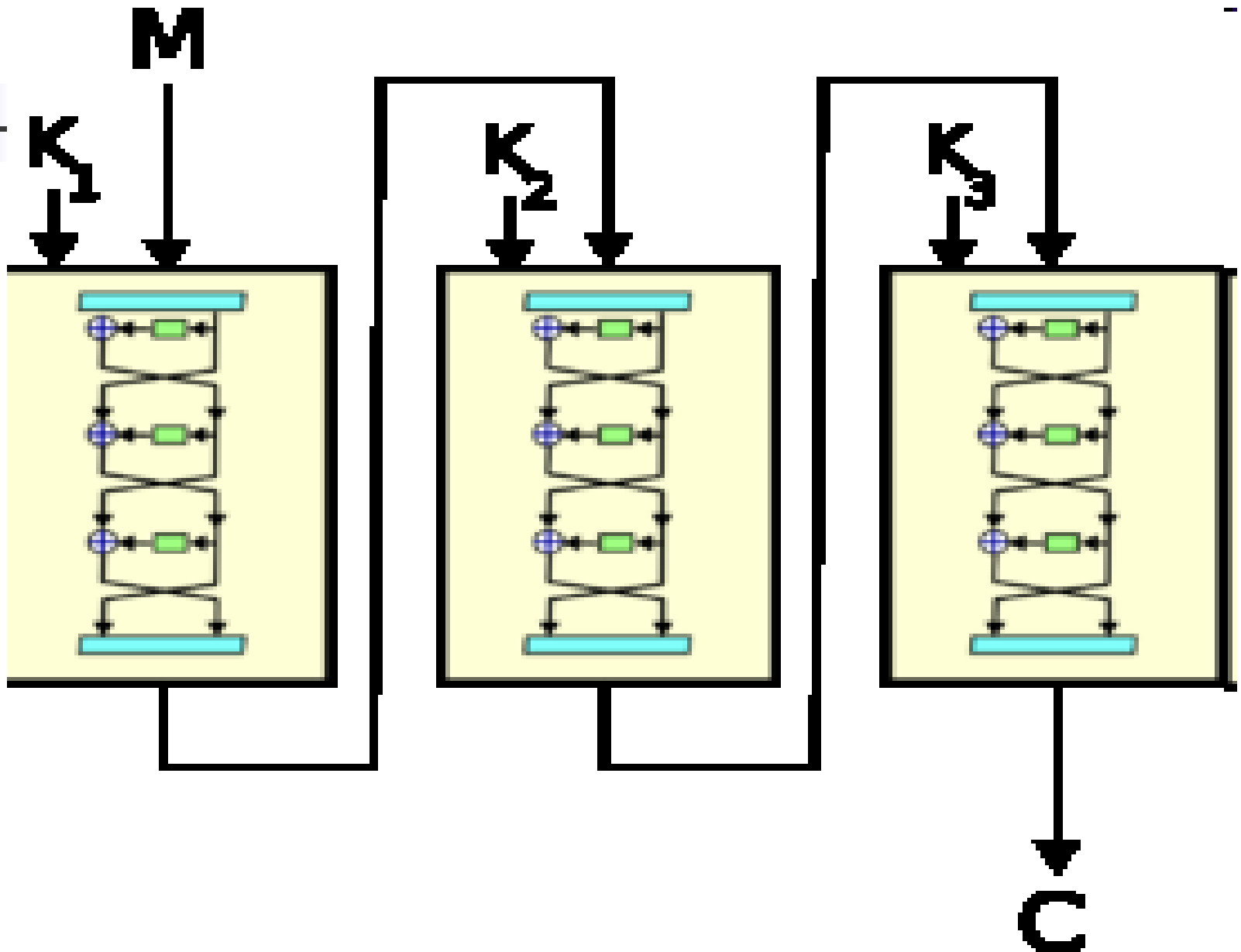


Схема Triple DES

где k1, k2, k3 — ключи для каждого DES-шага,

M — входные данные, которые нужно шифровать.

C- зашифрованные данные

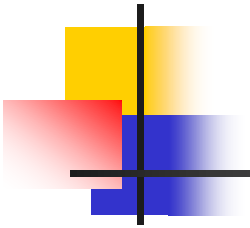
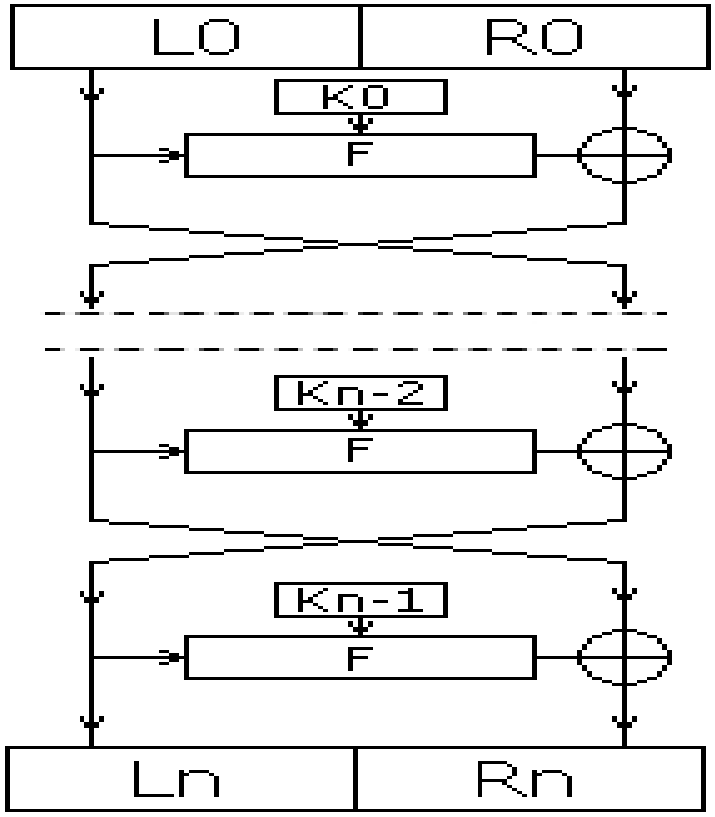


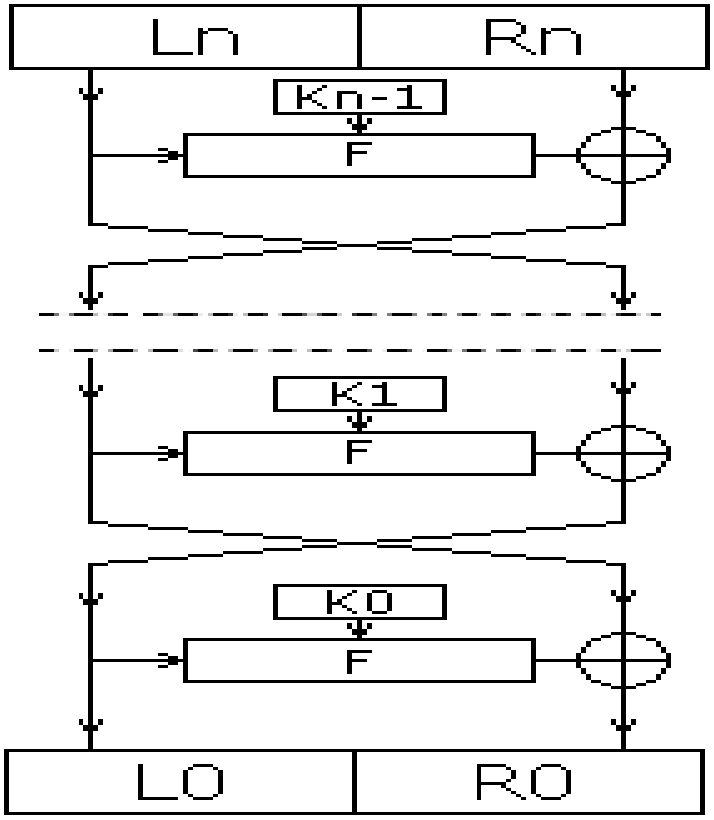
Схема Triple DES

При выполнении алгоритма 3DES ключи могут быть выбраны так:

1) k1, k2, k3 независимы.
2) k1, k2 независимы, а k1 = k3
3) k1 = k2 = k3

Сеть Фейстеля
слева – шифрование,
справа - расшифрование





Сеть Фейстеля

Шифрование:

Вся информация разбивается на блоки фиксированной длины. В случае, если длина входного блока меньше, чем размер, который шифруется заданным алгоритмом, то блок удлиняется каким-либо способом. Как правило длина блока является степенью двойки, например: 64 бита, 128 бит.

Выбранный блок делится на два равных подблока — «левый» (L0) и «правый» (R0).

Сеть Фейстеля

«Левый подблок» L0 видоизменяется функцией f(L0,K0) в зависимости от раундового ключа K0, после чего он складывается по модулю 2 с «правым подблоком» R0.

Результат сложения присваивается новому левому подблоку L1, который будет половиной входных данных для следующего раунда, а «левый подблок» L0 присваивается без изменений новому правому подблокуR1 (см. схему), который будет другой половиной.

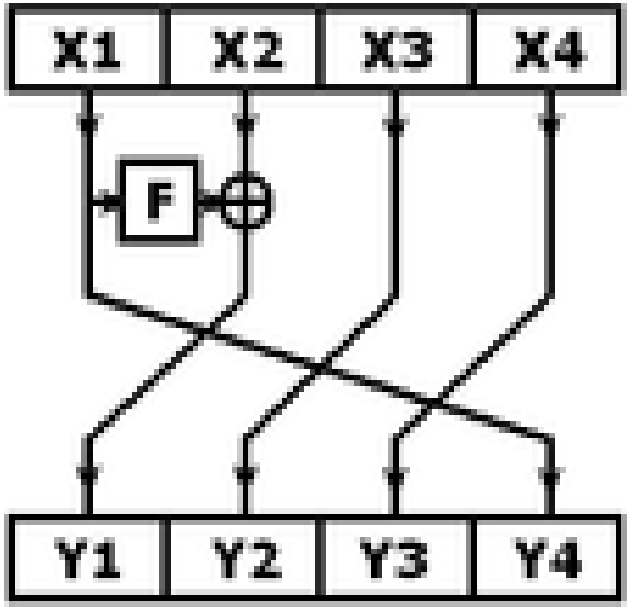
После чего операция повторяется N-1 раз, при этом при переходе от одного этапа к другому меняются раундовые ключи (K0 на K1 и т. д.) по какому-либо математическому правилу, где N — количество раундов в заданном алгоритме.

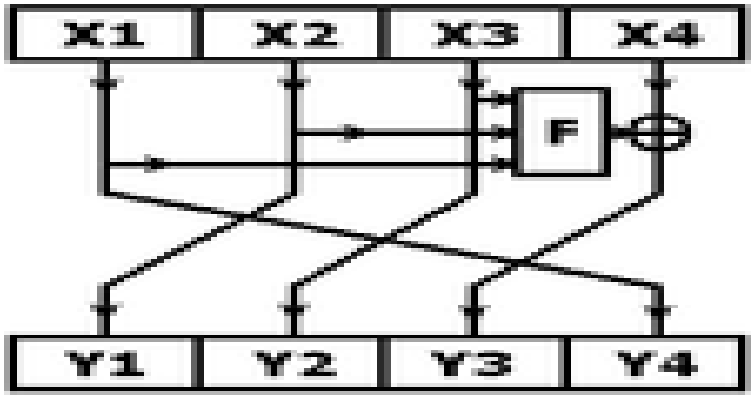
Сеть Фейстеля

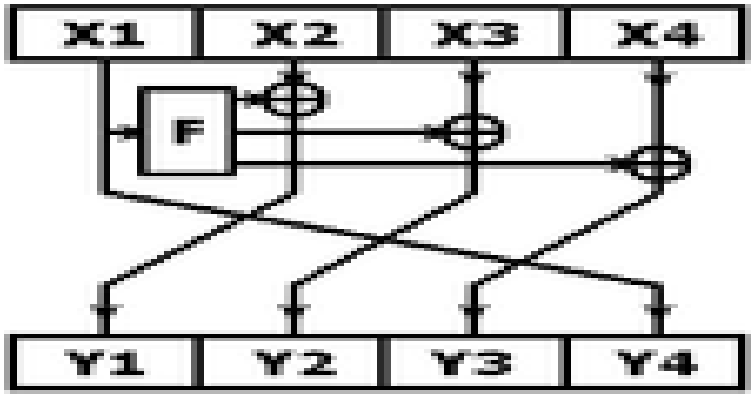
Расшифрование

Расшифровка информации происходит так же, как и шифрование, с тем лишь исключением, что ключи идут в обратном порядке, то есть не от первого к N-ному, а от N-го к первому.

Модификации сети Фейстеля



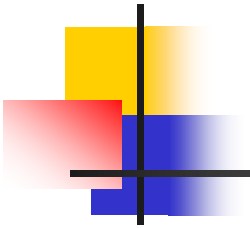




Advanced Encryption Standard (AES)

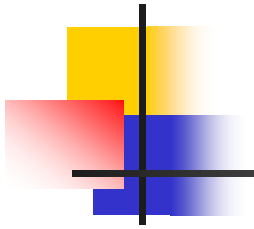
Advanced Encryption Standard (AES), также известный как Rijndael — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES.

Advanced Encryption Standard (AES)

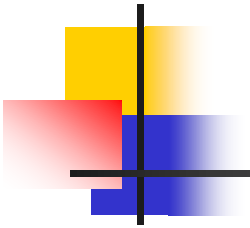


26 мая 2002 года AES был объявлен стандартом шифрования. По состоянию на 2006 год AES является одним из самых распространённых алгоритмов симметричного шифрования.

Advanced Encryption Standard (AES)

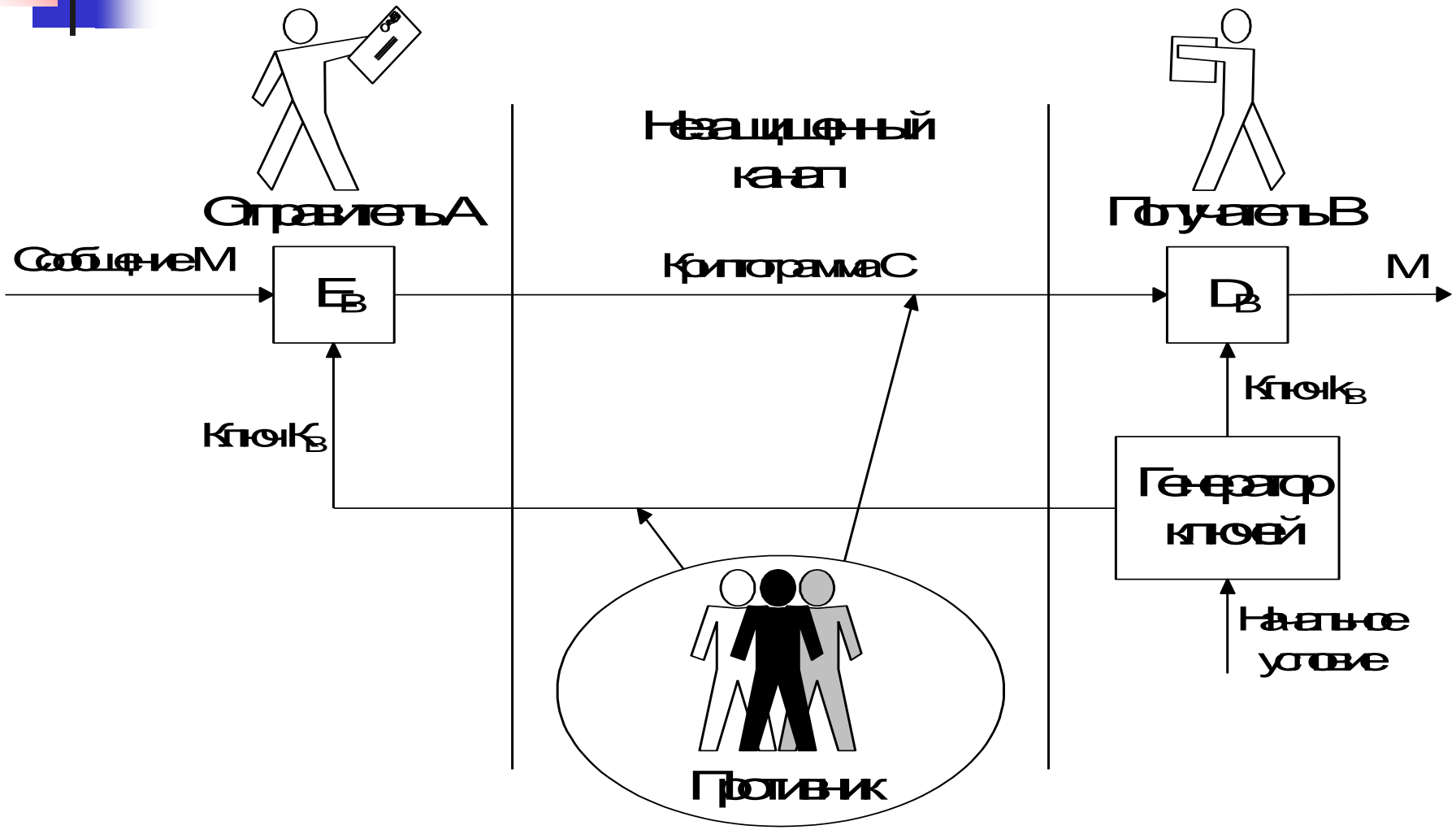


Использование ассиметричных алгоритмов для шифрования ключей в EFS



Поставщики услуг криптографии поддерживают два алгоритма: RSA Base и RSA Enhanced - для создания сертификатов EFS и для шифрования симметричных ключей шифрования.

Обобщенная схема асимметричной криптосистемы с открытым ключом



Обобщенная схема асимметричной криптосистемы с открытым ключом

В этой криптосистеме применяют два различных ключа:

КВ – открытый ключ получателя В, используемый отправителем А:

kB – секретный ключ получателя В.

Генератор ключей целесообразно располагать на стороне получателя В (чтобы не пересылать секретный ключ kB по незащищенному каналу).

Обобщенная схема асимметричной криптосистемы с открытым ключом

Процесс передачи зашифрованной информации в асимметричной криптосистеме осуществляется следующим образом:

1. Подготовительный этап:

   - абонент В генерирует пару ключей: секретный ключ kB и открытый ключ КВ ;

   - открытый ключ КВ посылает абоненту А и остальным абонентам (или делается доступным, например, на разделяемом ресурсе).

Обобщенная схема асимметричной криптосистемы с открытым ключом

2. Использование – обмен информацией между абонентами А и В:

- абонент А зашифровывает сообщение с помощью открытого ключа КВ абонента В и отправляет шифртекст абоненту В;

- абонент В расшифровывает сообщение с помощью своего секретного ключа kB , Никто другой (в том числе абонент А) не может расшифровать данное сообщение, так как не имеет секретного ключа абонента В. Защита информации в асимметричной криптосистеме основана на секретности ключа kB получателя сообщения.

Обобщенная схема асимметричной криптосистемы с открытым ключом

Характерные особенности асимметричных криптосистем:

- открытый ключ и криптограмма могут быть отправлены по незащищенным каналам;

- алгоритмы шифрования и расшифрования являются открытыми.

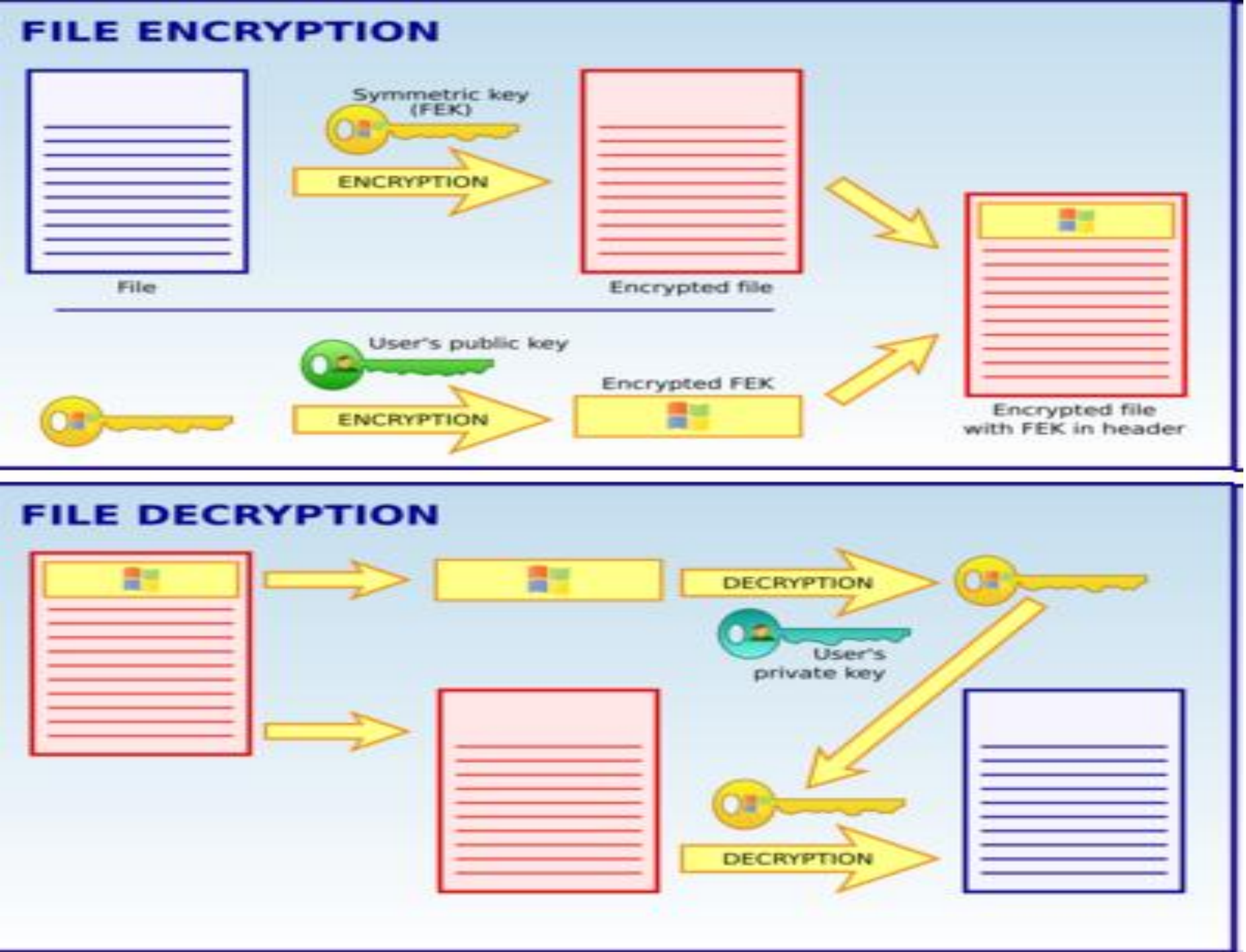
Обобщенная схема асимметричной криптосистемы с открытым ключом

Примеры ассиметричных алгоритмов шифрования: алгоритм RSA

- Система RSA используется для защиты программного обеспечения и в схемах цифровой подписи.

- Также она используется в открытой системе шифрования PGP и иных системах шифрования в сочетании с симметричными алгоритмами.

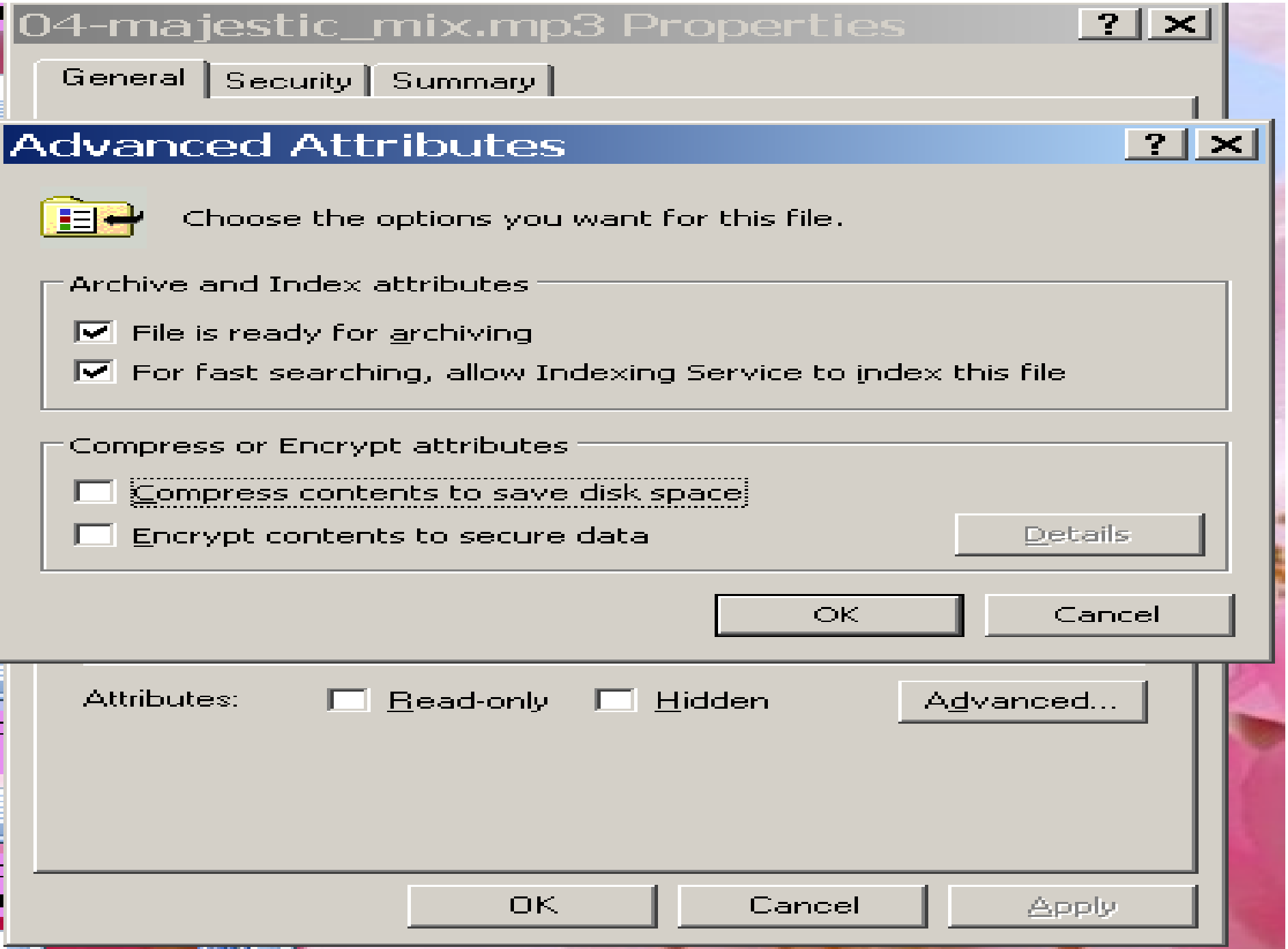
- Из-за низкой скорости шифрования (около 30 кбит/с при 512 битном ключе на процессоре 2 ГГц), сообщения обычно шифруют с помощью более производительных симметричных алгоритмов со случайным ключом (сеансовый ключ), а с помощью RSA шифруют лишь этот ключ. (в данное время широкое применение находят AES, IDEA, Serpent, Twofish).


FILE ENCRYPTION
Symmetric key
(FEK)
ENCRYPTION
File
Encrypted file
User's public key
ENCRYPTION
Encrypted FEK
Encrypted file
with FEK in header
FILE DECRYPTION
DECRYPTION
User's
private key
DECRYPTION

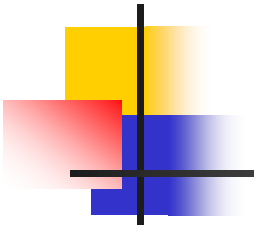
Повышение безопасности с помощью EFS

Windows XP Professional включает также ***команду Cipher***, предоставляющую возможности шифрации и дешифрации файлов из командной строки.

Чтобы восстановить зашифрованный файл в случае утраты его владельцем секретного ключа, в Windows XP Professional ***предусмотрен агент восстановления.***


04-majestic_mix.mp3 Properties
General
Security
Summary
Advanced Attributes
Choose the options you want for this file.
Archive and Index attributes
File is ready for archiving
For fast searching, allow Indexing Service to index this file
Compress or Encrypt attributes
Compress contents to save disk space
Encrypt contents to secure data
Details
OK
Cancel
Attributes:
Read-only
Hidden
Advanced...
OK
Cancel
Apply

Что разрешается шифровать?



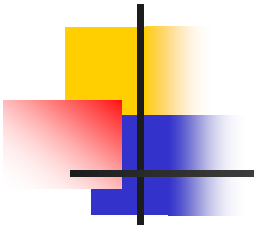
При установленном атрибуте шифрования папки EFS автоматически шифрует:

- все новые файлы, создаваемые в папке;
- все незашифрованные файлы, скопированные или перемещенные в папку;
- все вложенные файлы и подпапки (по особому требованию);
- автономные файлы.

Хранилища сертификатов с открытыми ключами

Сертификаты пользователя расположены в папке Documents and Settings\<имя_пользователя>\ApplicationData\Microsoft\SystemCertificates\My\Certificates профиля пользователя. Эти сертификаты записываются в локальном реестре при каждом входе в систему компьютера. Для перемещаемых профилей сертификаты обычно хранятся в определенном месте (не на компьютере) и "следуют" за пользователем при его входе в систему любого компьютера в домене.

Хранение закрытых ключей



Поставщики услуг криптографии (cryptographic service provider, CSP) - как Base CSP, так и Enhanced CSP, хранят закрытые ключи в профиле пользователя в папке:

%SystemRoot%\Documents and Settings\<имя_пользователя>\

Application Data\Microsoft\Crypto\RSA.

В перемещаемых профилях пользователей закрытый ключ располагается в папке RSA на контроллере домена и загружается на компьютер только на время его работы.