4.1. Устройства ввода-вывода

Типы устройств по функциональному назначению;

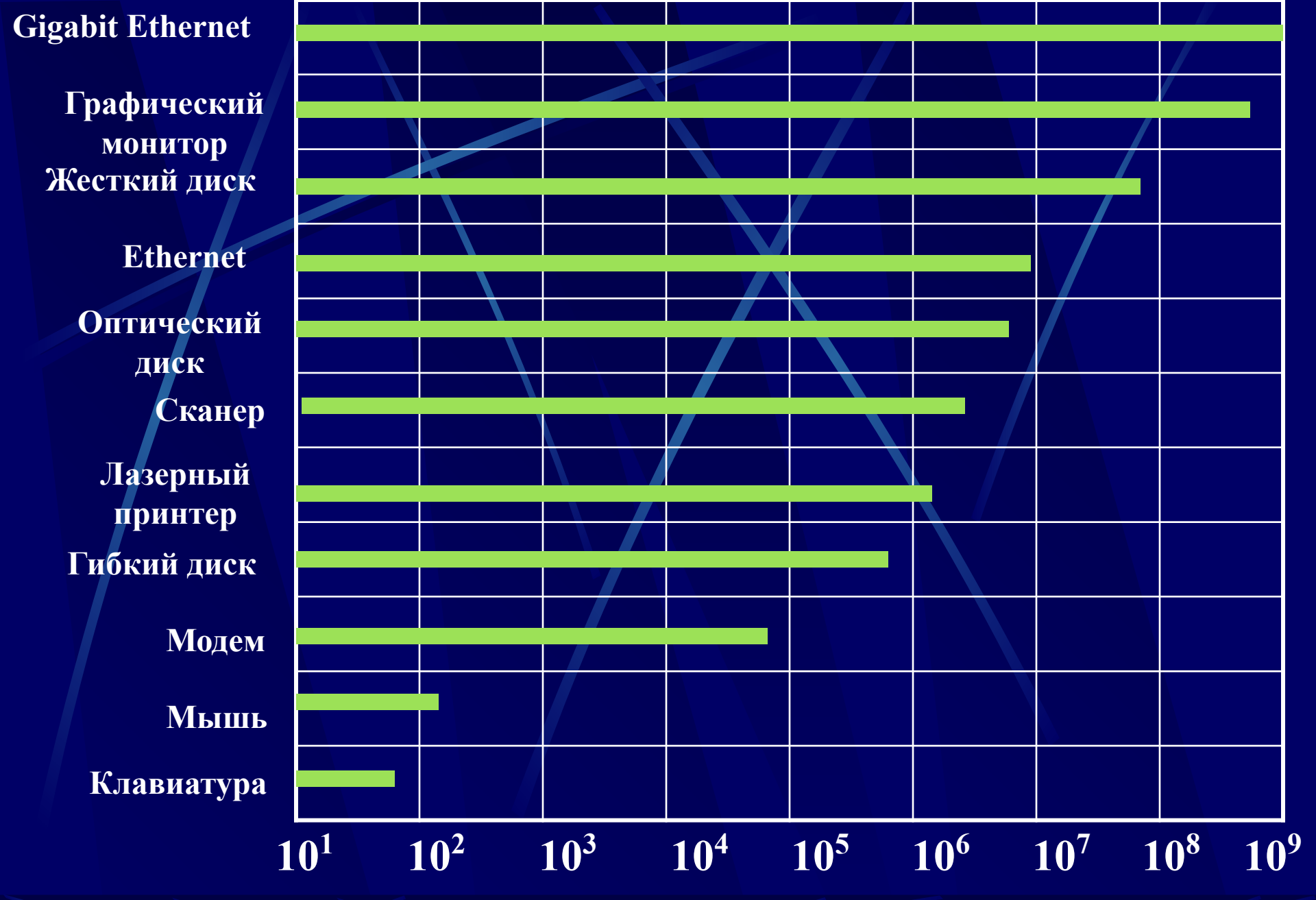
1. Работающие с пользователем. Используются для связи с пользователем компьютера (принтеры, дисплеи, клавиатура, манипуляторы (мышь, джойстик и т. п.).
2. Работающие с компьютером. Используются для связи с электронным оборудованием (диски, магнитные ленты, датчики, контроллеры, преобразователи и т. п.).
3. Коммуникации. Используются для связи с удаленными устройствами (модемы, адаптеры цифровых линий и др.).

Типы устройств по принципам функционирования:

1. Блочные, хранящие информацию в виде адресуемых блоков фиксированного размера и позволяющие работать с каждым блоком независимо от других блоков (дисковые устройства).
2. Символьные, принимающие или предоставляющие поток символов без какой-либо структуры (принтеры, модемы, сетевые карты).

Различия в характеристиках устройств ввода-вывода

- **Скорость передачи данных (на несколько порядков).**
- **Применение. Один и тот же тип устройства может требовать различного ПО и стратегии операционной системы (диск для хранения файлов приложений и файла подкачки, терминал пользователя и администратора).**
- **Сложность управления (для принтера относительно простой интерфейс управления, для диска – намного сложнее).**
- **Единицы передачи данных. Данные могут передаваться блоками или потоком байтов или символов.**
- **Представление данных. Различные устройства используют разные схемы кодирования данных, включая различную кодировку символов и контроль четности.**
- **Условия ошибок. Природа ошибок, способ сообщения о них, возможные ответы резко отличаются от одного устройства к другому.**


Gigabit Ethernet
Графический монитор
Жесткий диск
Ethernet
Оптический диск
Сканер
Лазерный принтер
Гибкий диск
Модем
Мышь
Клавиатура
10^1 10^2 10^3 10^4 10^5 10^6 10^7 10^8 10^9

4.2. Основные функции подсистемы ввода-вывода

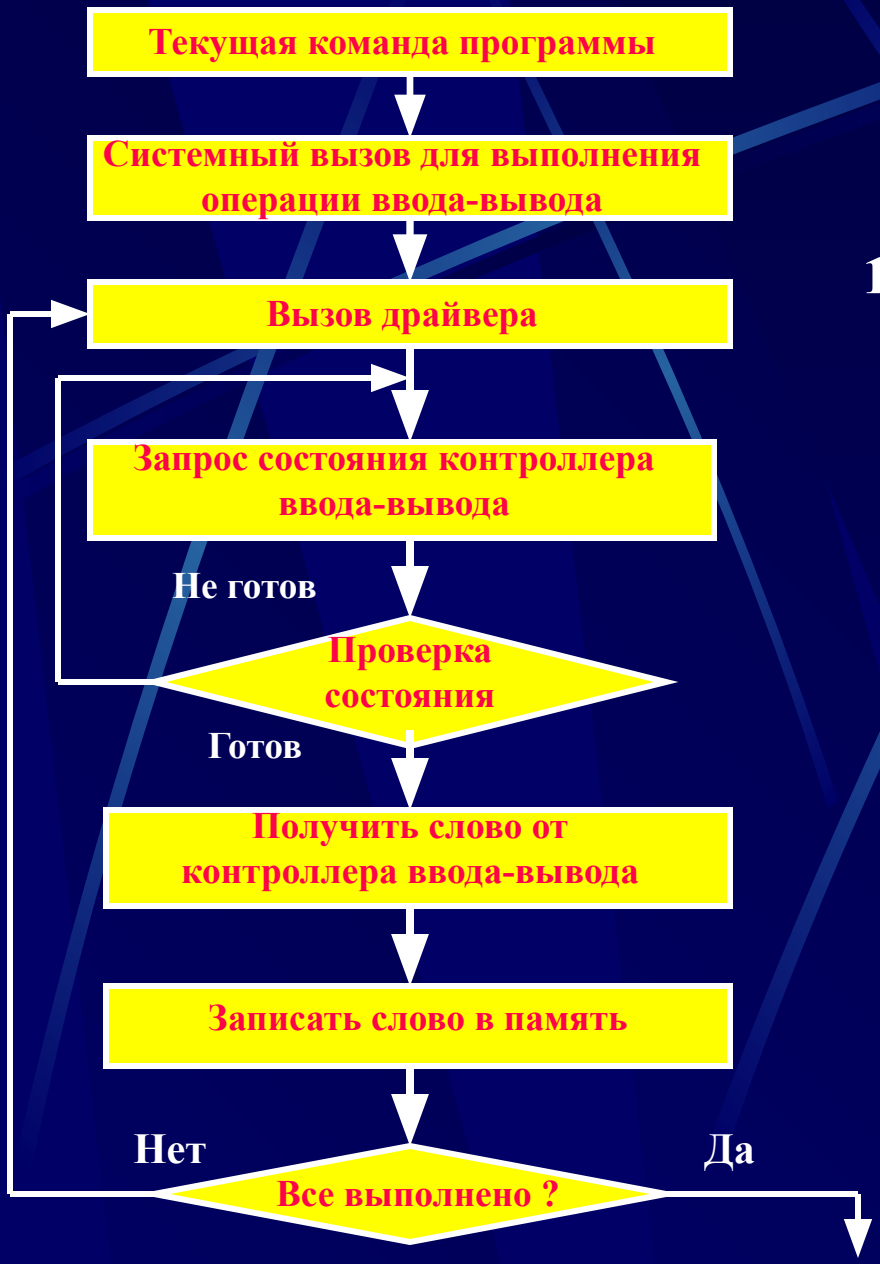
Основные компоненты: драйверы, файловая система, система прерываний

1. Организация параллельной работы устройств ввода-вывода и процессора.
2. Согласование скоростей обмена и кэширование данных.
3. Разделение устройств и данных между процессами.
4. Обеспечение удобного логического интерфейса между устройствами и остальной частью системы.
5. Поддержка широкого спектра драйверов с возможностью простого включения в систему нового драйвера.
6. Динамическая загрузка и выгрузка драйверов.
7. Поддержка нескольких файловых систем.
8. Поддержка синхронных и асинхронных операций ввода-вывода.

4.2.1. Организация параллельной работы устройств ввода-вывода и процессора

Эволюция ввода – вывода

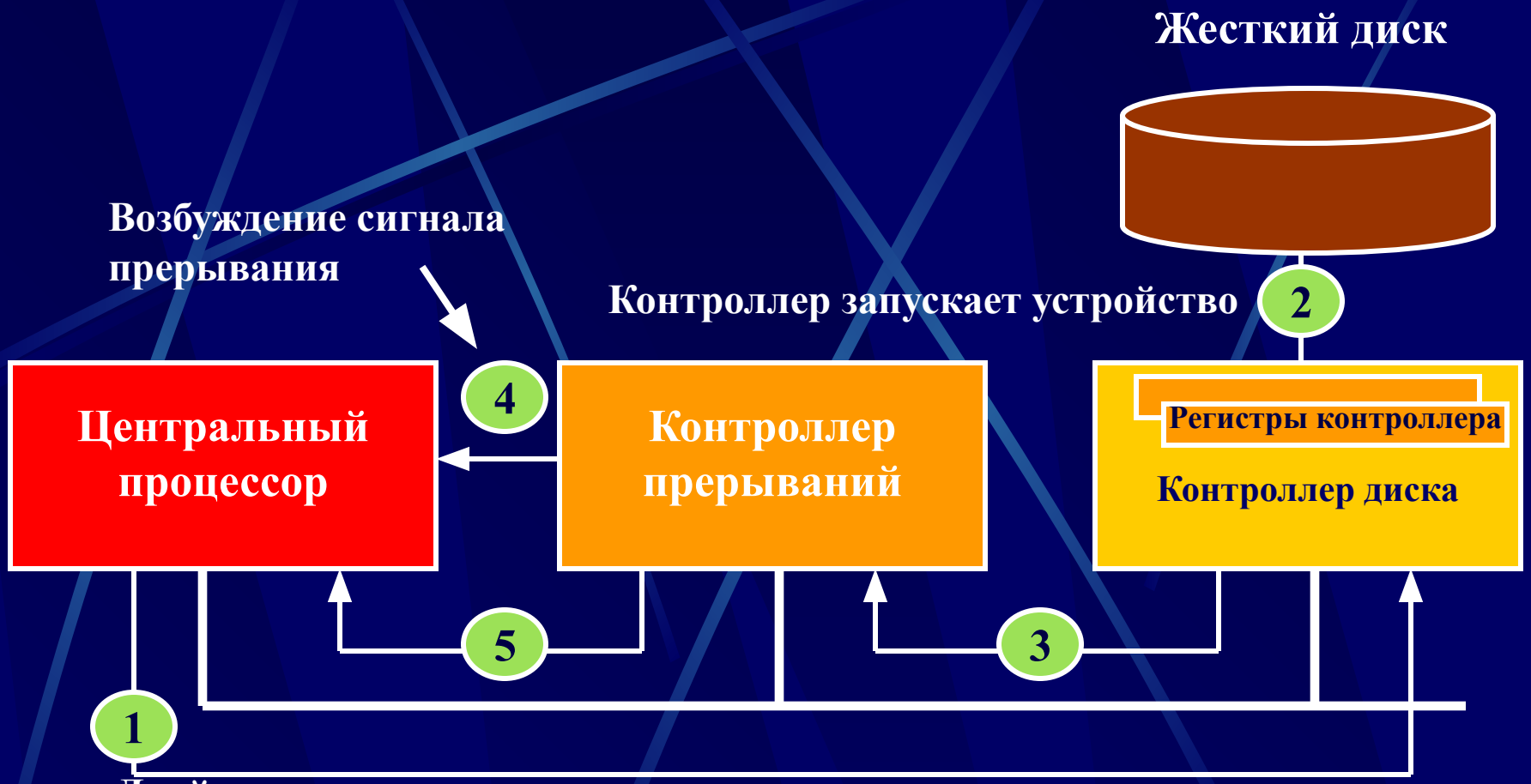
1. Процессор непосредственно управляет периферийным устройством.
2. Устройство управляется контроллером. Процессор использует программируемый ввод - вывод без прерываний (переход к абстракции интерфейса ввода - вывода).
3. Использование контроллера прерываний. Ввод-вывод, управляемый прерываниями.
4. Использование модуля (канала) прямого доступа к памяти. Перемещение данных в память (из нее) без использования процессора.
5. Использование отдельного специализированного процессора ввода-вывода, управляемого центральным процессором.
6. Использование отдельного компьютера для управления устройствами ввода-вывода при минимальном вмешательстве центрального процессора.



1. **Программируемый ввод-вывод без прерываний**

Процессор посылает необходимые команды контроллеру ввода-вывода и переводит процесс в состояние ожидания завершения операции ввода-вывода.

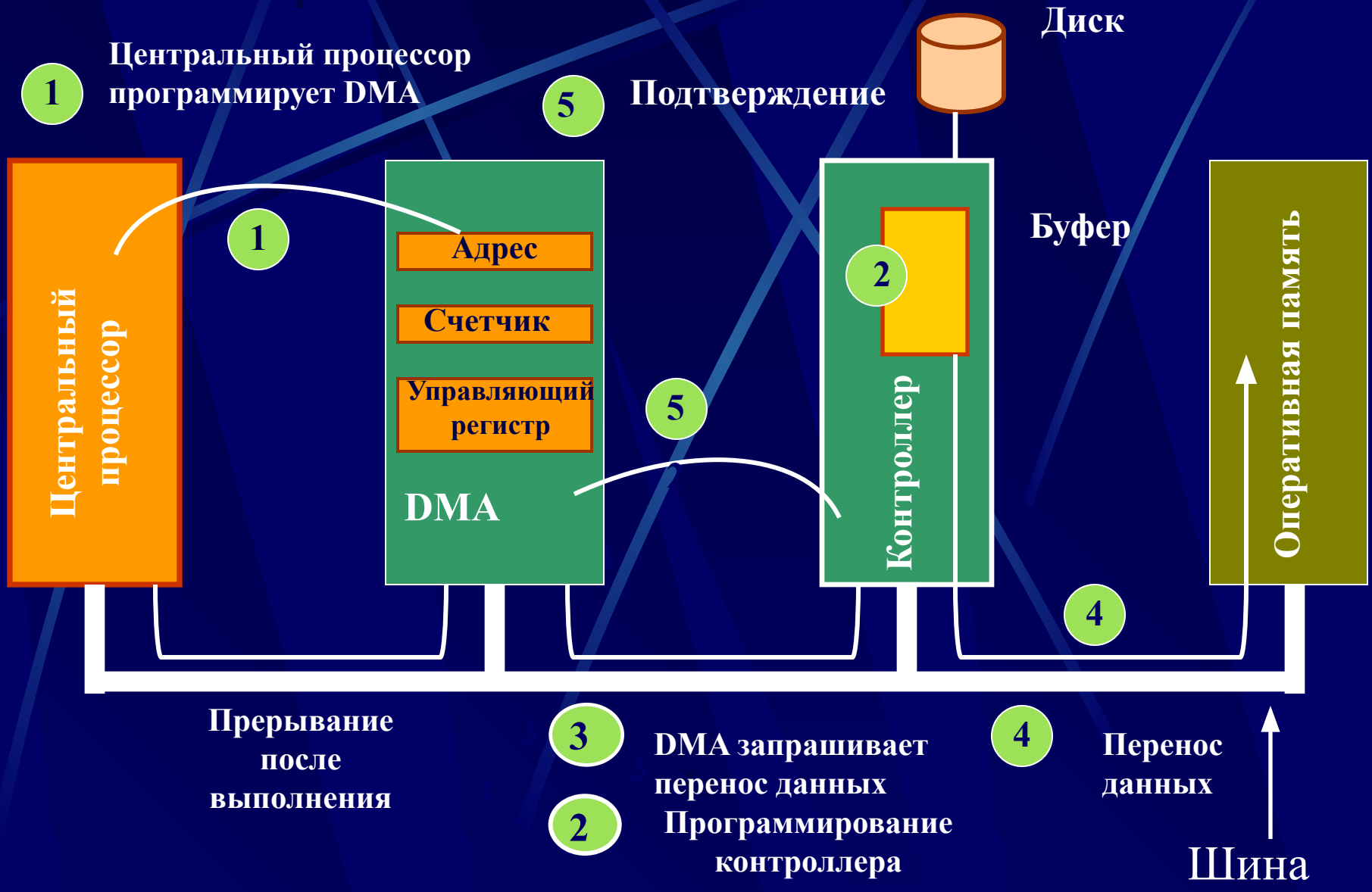
2. Ввод-вывод, управляемый прерываниями. Процессор посылает необходимые команды контроллеру ввода-вывода и продолжает выполнять процесс, если нет необходимости в ожидании выполнения операции. В противном случае процесс приостанавливается до получения прерывания, а процессор переключается на выполнение другого процесса.



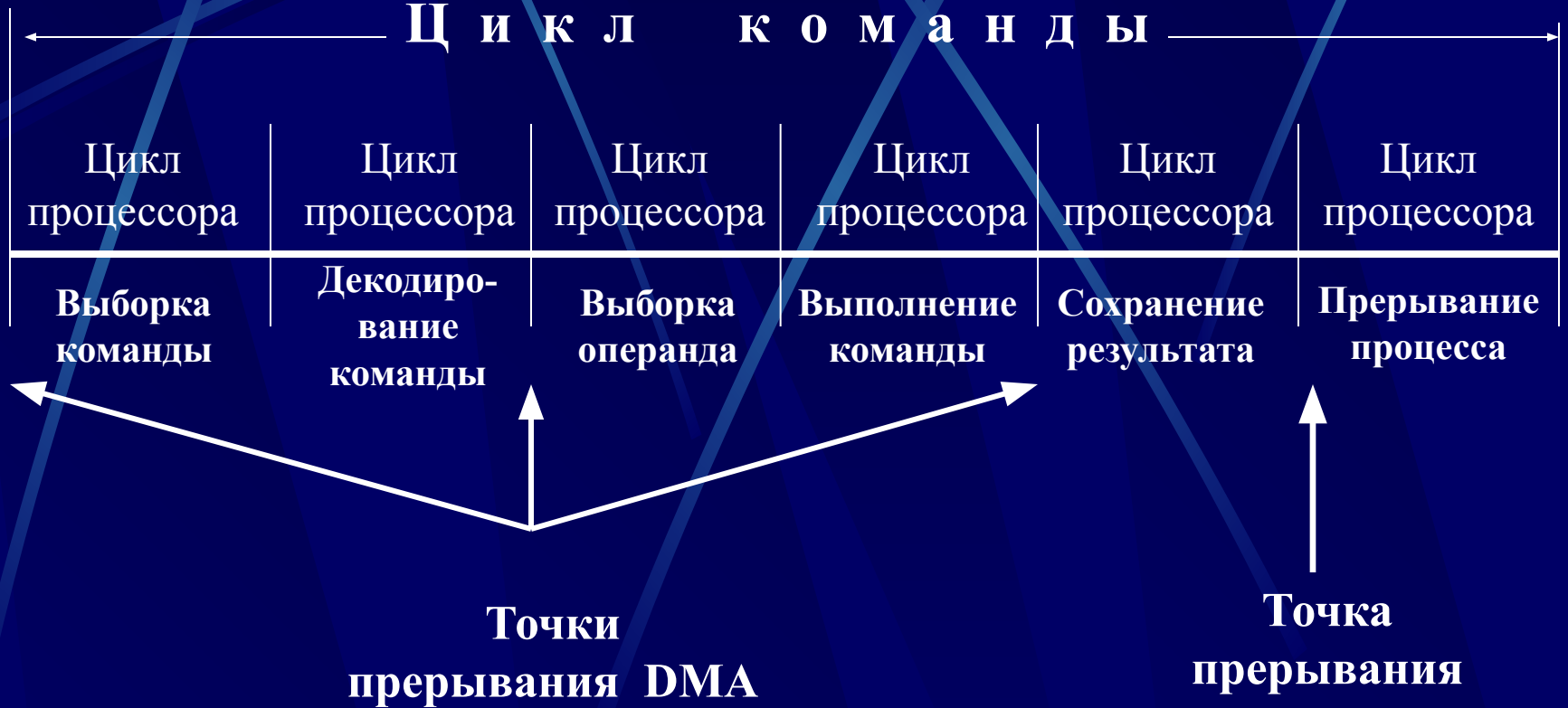
3 Контроллер завершил операцию

5 Обработка прерывания, перемещение данных в область программы, передача управления программе

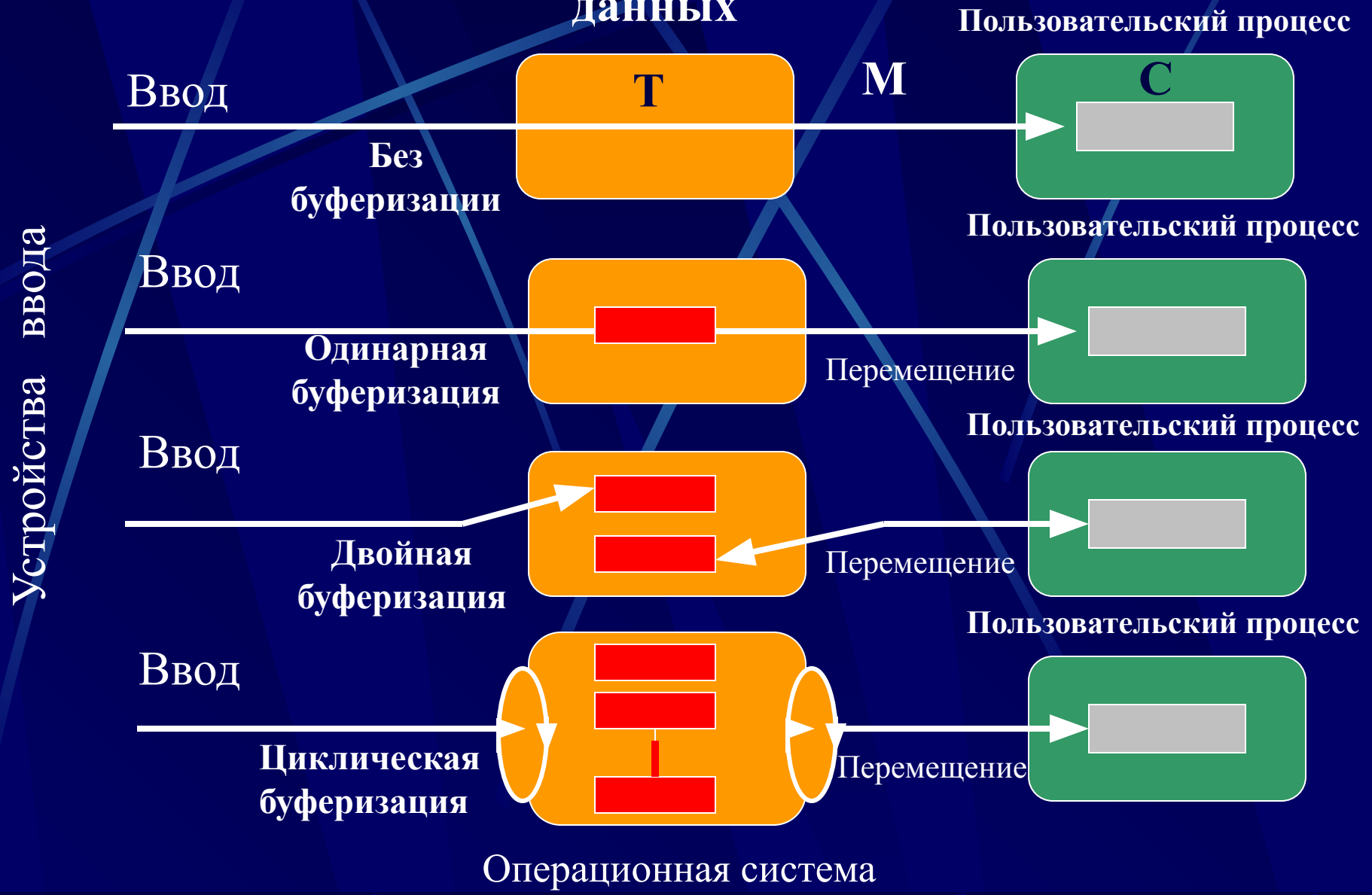
3. Прямой доступ к памяти. Модуль прямого доступа к памяти управляет обменом данных между основной памятью и контроллером ввода-вывода. Процессор посылает запрос на передачу блока данных модулю прямого доступа к памяти, а прерывание происходит только после передачи всего блока данных.



Работа DMA-контроллера


Цикл команды
Цикл процессора
Цикл процессора
Цикл процессора
Цикл процессора
Цикл процессора
Цикл процессора
Выборка команды
Декодирование команды
Выборка операнда
Выполнение команды
Сохранение результата
Прерывание процесса
Точки прерывания DMA
Точка прерывания

4.2.2. Согласование скоростей обмена и кэширование данных



Время обработки блока данных

Без буферизации $T + C$

Одинарная буферизация $\max \{T, C\} + M$

в большинстве случаев $T + C > \max \{T, C\}$

Двойная буферизация $\max \{T, C\}$

если **C <= T,** то блочно-ориентированное устройство может работать с максимальной скоростью;

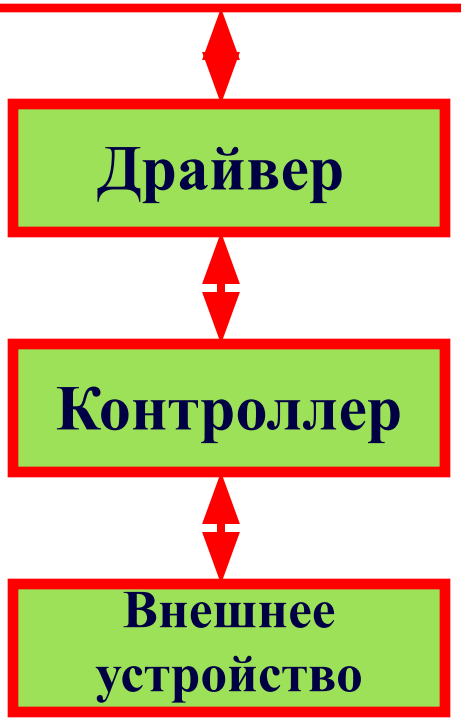
если **C > T,** то процесс избавляется от необходимости ожидания завершения ввода-вывода.

Циклическая буферизация используется при высокой частоте ввода-вывода.

Буферизация данных позволяет сократить количество реальных операций ввода за счет кэширования данных.

4.2.6. Поддержка широкого спектра драйверов

Операционная система



Интерфейс драйвер – ядро (Driver Kernel Interface, DKI)

Интерфейс драйвер – устройство (Driver Device Interface, DDI)

Аппаратный низкоуровневый интерфейс контроллер - устройство

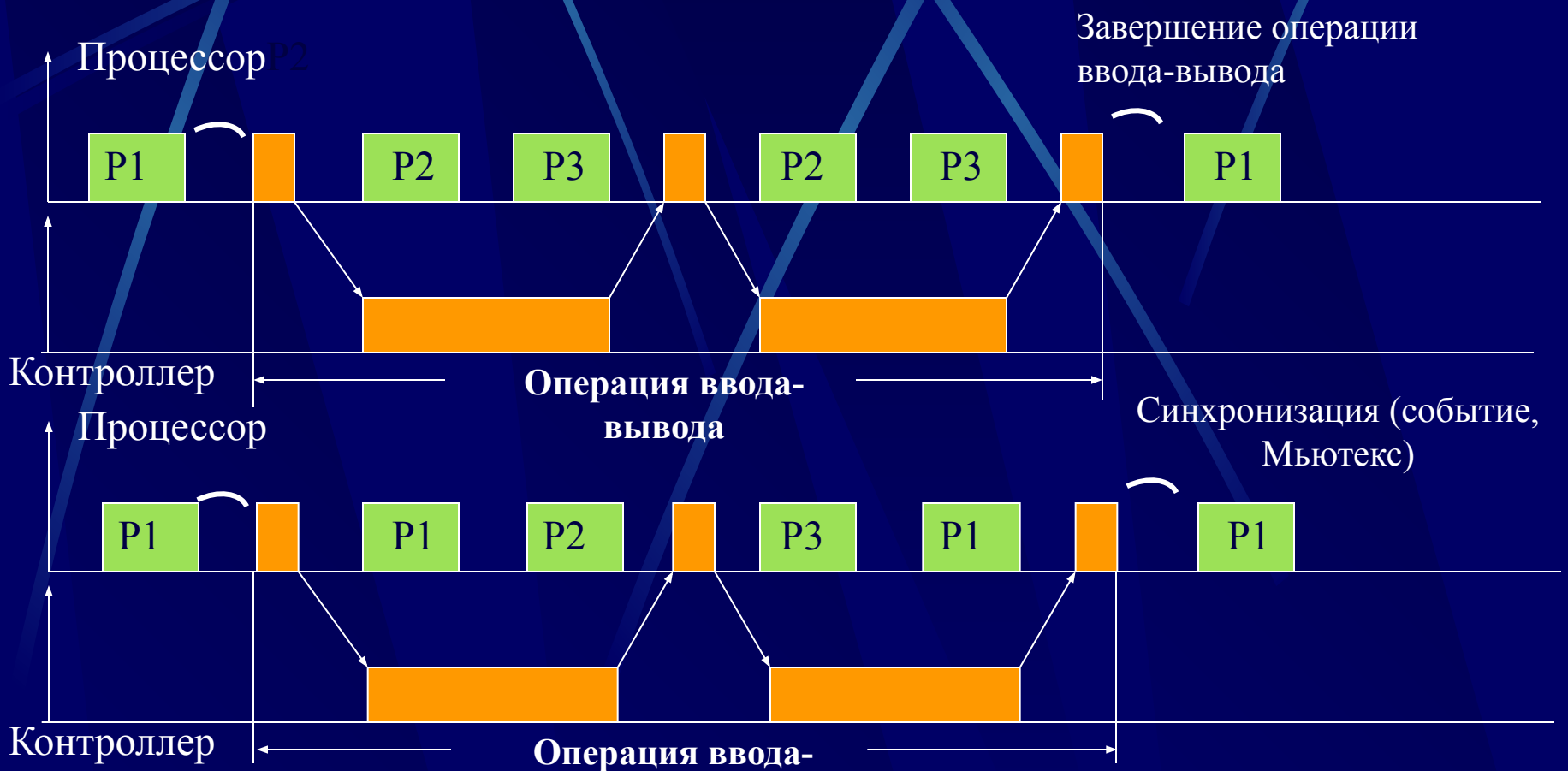
Функции драйвера

1. Обработка запросов записи-чтения от программного обеспечения управления устройствами. Постановка запросов в очередь
2. Проверка входных параметров запросов и обработка ошибок
3. Инициализация устройства и проверка статуса устройства
4. Управление энергопотреблением устройства.
5. Регистрация событий в устройстве
6. Выдача команд устройству и ожидание их выполнения возможно в блокированном состоянии до поступления прерывания от устройства
7. Проверка правильности завершения операции
8. Передача запрошенных данных и статуса завершенной операции
9. Обработка нового запроса при незавершенном предыдущем запросе (для реентерабельных драйверов)

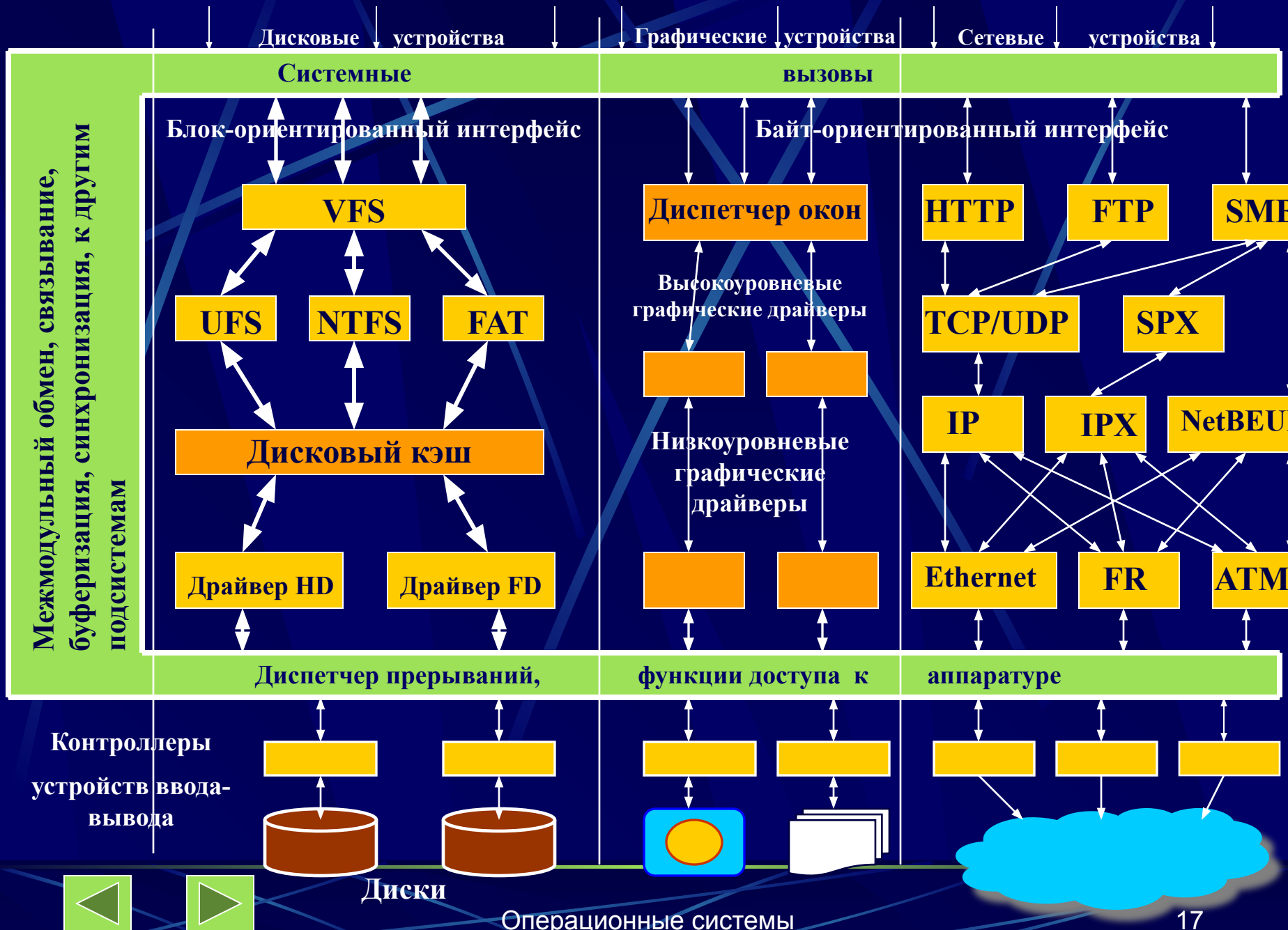
4.2.6. Динамическая выгрузка и загрузка драйверов

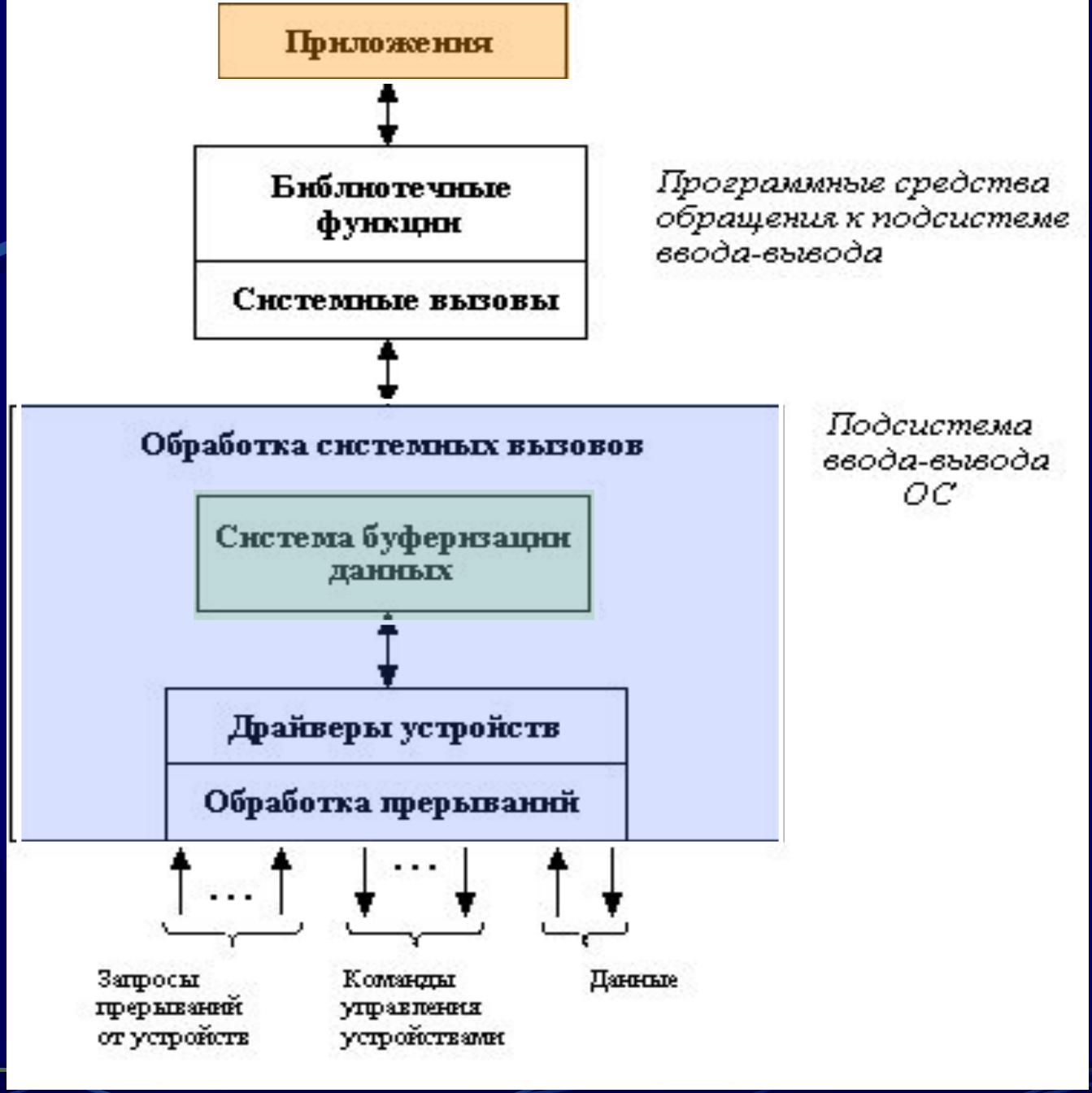
4.2.7. Поддержка нескольких файловых систем

4.2.8. Поддержка синхронных и асинхронных операций ввода-вывода



4.3. Многослойная модель подсистемы ввода-вывода




Приложения
Библиотечные функции
Системные вызовы
Программные средства обращения к подсистеме ввода-вывода
Обработка системных вызовов
Система буферизации данных
Драйверы устройств
Обработка прерываний
Подсистема ввода-вывода ОС
Запросы прерываний от устройств
Команды управления устройствами
Данные

4.4. Файловая система

4.4.1. Основные понятия. Цели и задачи файловой системы

Причины создания файловых систем:

1. Необходимость длительного (иногда вечного) и надежного хранения больших объемов информации.
2. Обеспечение возможности совместного использования информации различными приложениями. Эффективное разделение, защита и восстановление данных.

Решение этих проблем заключается в хранении информации в файлах. Файл – это поименованная совокупность данных, хранящаяся на каком-либо носителе информации.

При рассмотрении файлов используются следующие понятия:

1. Поле (field) – основной элемент данных.
2. Запись (record) – набор связанных полей, которые могут обрабатываться как единое целое.
3. Файл (file) – совокупность однородных записей.
4. База данных (database) – набор связанных данных, представленных совокупностью файлов

Файловая система – это часть операционной системы, включающая:

- совокупность всех файлов на различных носителях информации (магнитные диски, магнитные ленты, CD-ROM и т. п.);
- наборы структур данных, используемых для управления файлами (каталоги и дескрипторы файлов, таблицы распределения свободного и занятого пространства носителей информации);
- комплекс системных программных средств, реализующих различные операции над файлами (создание, чтение, запись, уничтожение, изменение свойств и др.).

Задачи файловой системы

- соответствие требованиям управления данными и требованиям со стороны пользователей, включающим возможности хранения данных и выполнения операций с ними;
- гарантирование корректности данных, содержащихся в файле;
- оптимизация производительности, как с точки зрения системы (пропускная способность), так и с точки зрения пользователя (время отклика);
- поддержка ввода-вывода для различных типов устройств хранения информации;
- минимизация или полное исключение возможных потерь или повреждений данных;
- защита файлов от несанкционированного доступа;
- обеспечение поддержки совместного использования файлов несколькими пользователями (в том числе средства блокировки файла и его частей, исключение тупиков, согласование копий и т. п.);
- обеспечение стандартного набора подпрограмм интерфейса ввода-вывода.

Требования к файловой системе со стороны пользователя диалоговой системы общего назначения

1. Создание, удаление, чтение и изменения файлов.
2. Контролируемый доступ к файлам других пользователей.
3. Управление доступом к своим файлам.
4. Реструктурирование файлов в соответствии с решаемой задачей.
5. Перемещение данных между файлами.
6. Резервирование и восстановление файлов в случае повреждения.
7. Доступ к файлам по символическим именам.


4.4.2. Архитектура файловой системы

Методы доступа

Доступ к записям

Выбор устройства, пла-нирование распределе-ни внешней памяти

Буферизация, обмен блоками

Инициализация, выпол-нение и завершение опе-рации

4.4.3. Организация файлов и доступ к ним

4.4.3.1. Типы, именование и атрибуты файлов

Обычные файлы – содержат информацию, занесенную пользователем, системной или прикладной программой.

Каталоги – системные файлы, поддерживающие структуру файловой системы.

Специальные файлы – фиктивные файлы, ассоциированные с устройствами ввода-вывода и используемые для унификации доступа к последовательным устройствам ввода-вывода.

Именованные конвейеры (каналы) – циклические буферы, позволяющие выходной файл одной программы соединить со входным файлом другой программы.

Отображаемые файлы – обычные файлы, отображаемые на адресное пространство процесса по указанному виртуальному адресу.

Правила именования файлов зависят от операционной системы,

Атрибут	Значение
Тип файла	Обычный, каталог, специальный и т. д.
Владелец файла	Текущий владелец
Создатель файла	Идентификатор пользователя, создавшего файл
Пароль	Пароль для получения доступа к файлу
Время	Создания, последнего доступа, последнего изменения
Текущий размер файла	Количество байтов в записи
Максимальный размер	Количество байтов, до которого можно увеличивать размер файла
Флаг «только чтение»	0 – чтение-запись, 1 – только чтение
Флаг «скрытый»	0 – нормальный, 1 – не показывать в перечне файлов каталога
Флаг «системный»	0 – нормальный, 1 – системный
Флаг «архивный»	0 – заархивирован, 1- требуется архивация
Флаг ASCII/двоичный	0 – ASCII, 1 – двоичный
Флаг произвольного доступа	0 – только последовательный доступ, 1 – произвольный доступ
Флаг «временный»	0 – нормальный, 1 – удаление после окончания работы процесса
Позиция ключа	Смещение до ключа в записи
Длина ключа	Количество байтов в поле ключа

4.4.3.2. Логическая организация файлов

Модель 1. Неструктурированная последовательность байт (ОС UNIX).

Модель 2. Структурированный файл : смешанный, последовательный, индексно-последовательный, индексированный, прямого доступа.

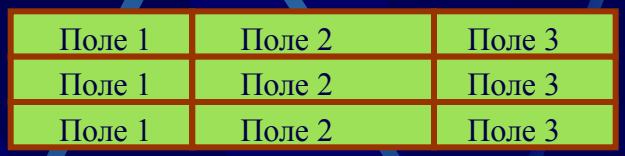
Смешанный файл

Каждое поле описывает само себя (имя, длина, значение). Доступ – полный перебор.

Достоинства: рациональное использование дискового пространства, хорошо подходят для полного перебора

Недостатки: сложность вставки и обновления записей

Последовательный файл

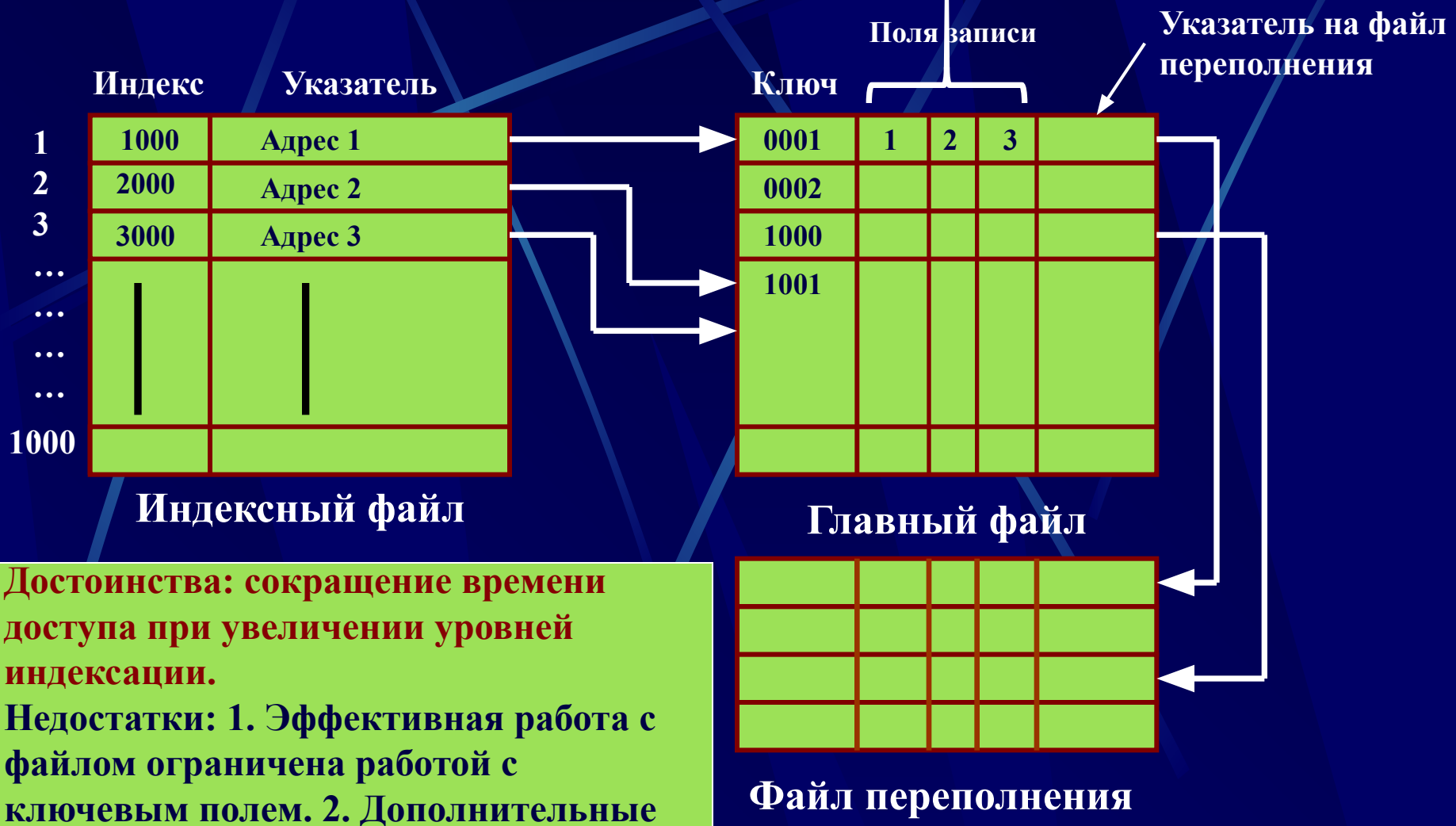


Записи имеют одну длину, одни и те же поля и хранят только значения полей (одно поле – ключевое). Атрибуты файловой структуры: имя и длина каждого поля.

Достоинства: оптимальный вариант для пакетных приложений, записи хранятся в ключевой последовательности, возможно хранение на диске и МЛ. Возможна организация в виде списка, что упрощает вставку новых записей.

Недостатки: малоэффективен для диалоговых приложений

Индексно-последовательный файл



Достоинства: сокращение времени доступа при увеличении уровней индексации.

Недостатки: 1. Эффективная работа с файлом ограничена работой с ключевым полем. 2. Дополнительные затраты времени на периодическое слияние с файлом переполнения.

Индексированный файл

Типы индексов:

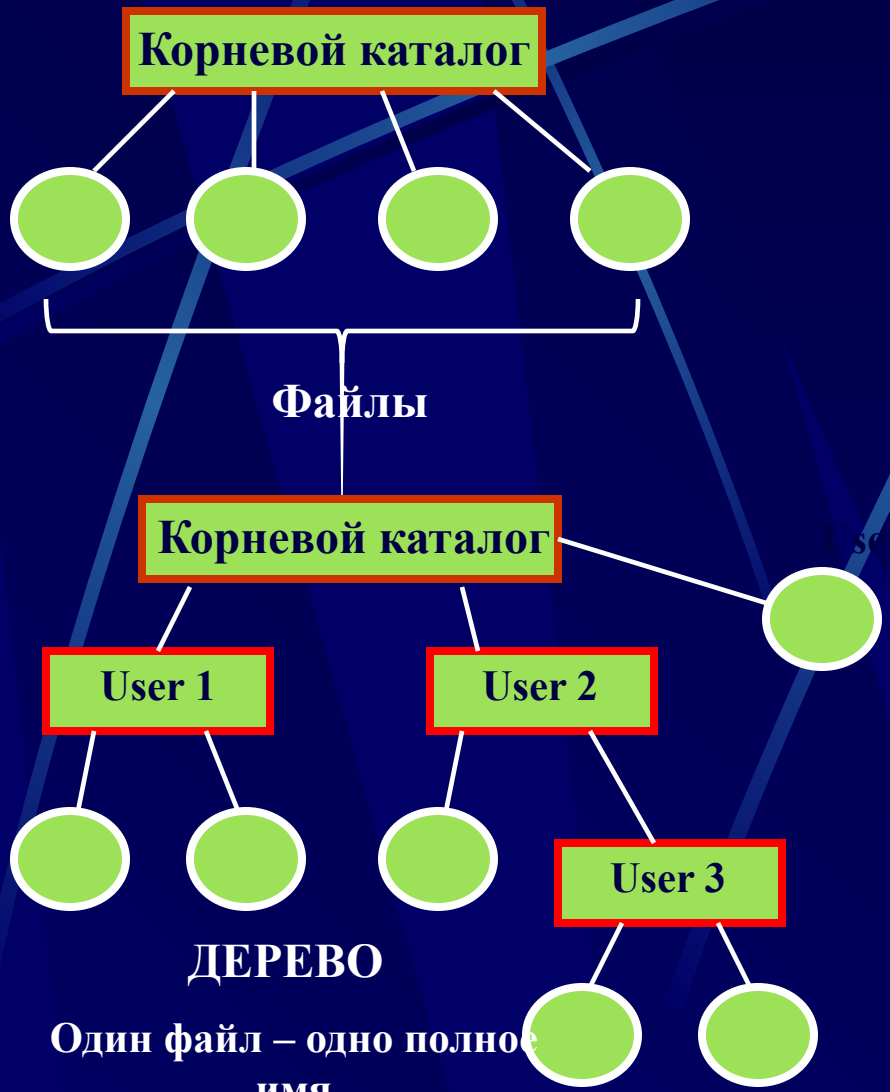
1. Полный индекс – содержит по одному элементу для каждой записи главного файла.
2. Частный индекс содержит элементы для записей, в которых имеется интересующее пользователя поле.
3. При добавлении новой записи в главный файл необходимо обновлять все индексные файлы.
4. Индексы организуются в виде последовательных файлов.

Достоинство: быстрый доступ. **Недостатки**: большая избыточность данных, неэффективность обработки всех записей файла.

Файл прямого доступа

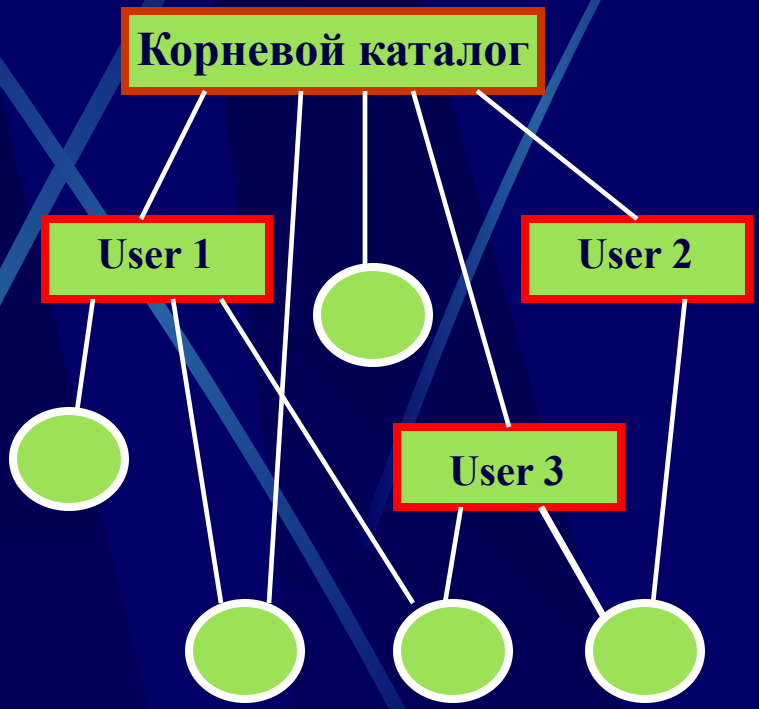
1. Обеспечивает прямой доступ к любой записи фиксированной длины по известному адресу (ключу) при хранении файлов на диске.
2. Достоинства: быстрый доступ к любой записи, простота вставки, удаления и модификации записей.
3. Недостатки: записи фиксированной структуры и длины.

4.4.4. Каталоговые системы



ДЕРЕВО

Один файл – одно полное имя

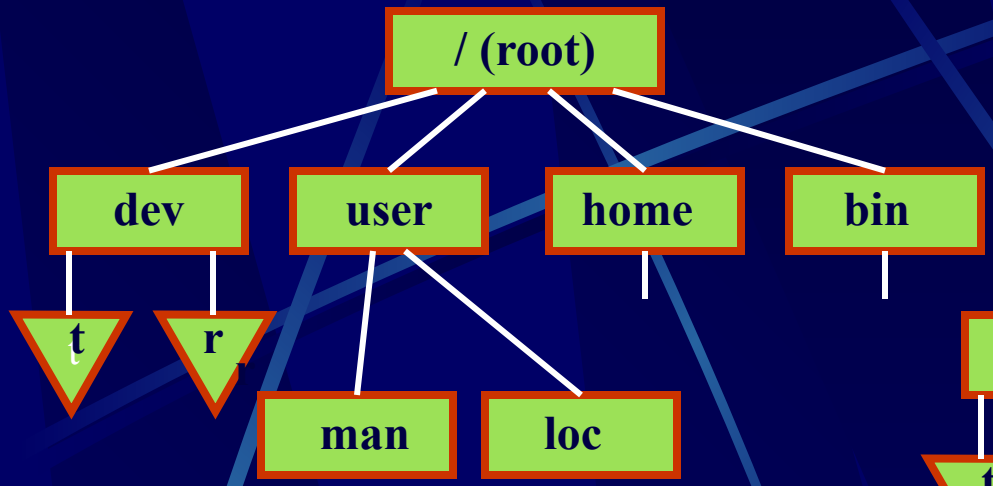


СЕТЬ

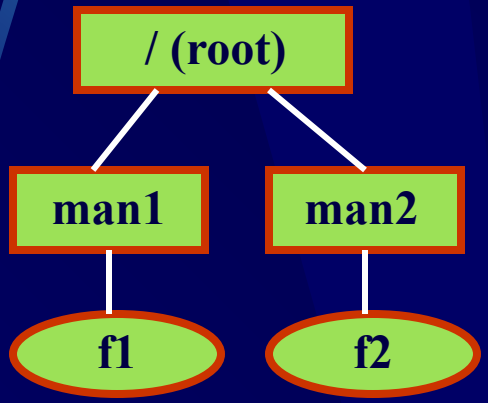
Один файл – много полных имен

Файловый каталог является связующим звеном между системой управления файлами и набором файлов

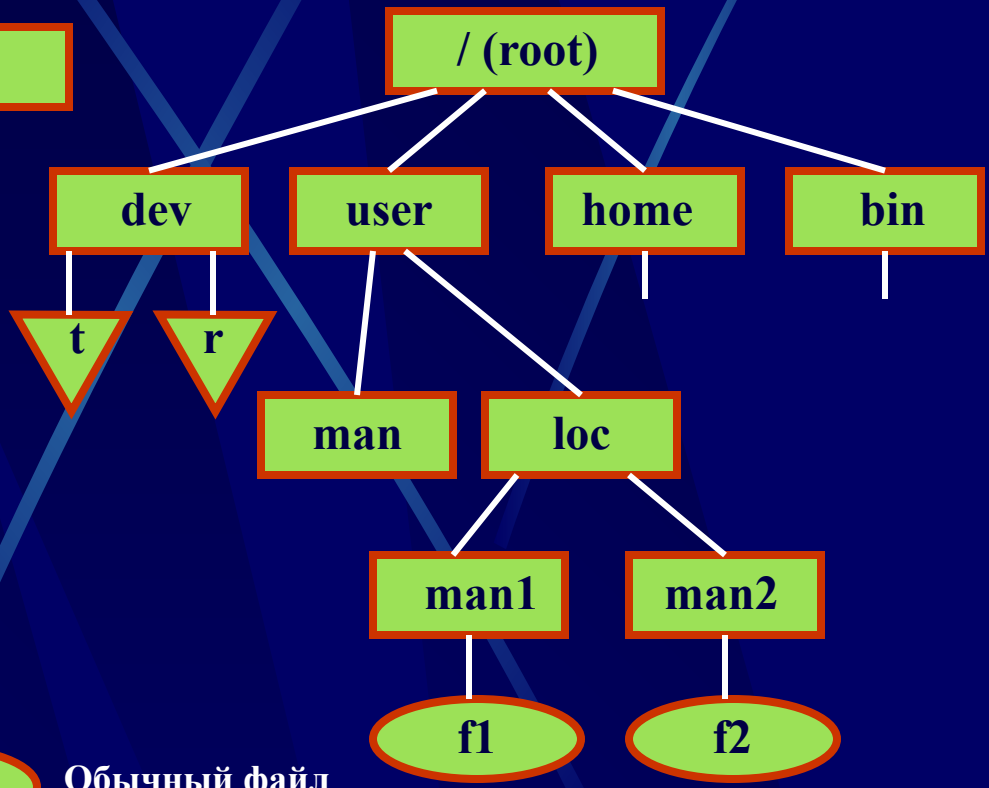
Монтирование



Файловая система 1



Файловая система 2

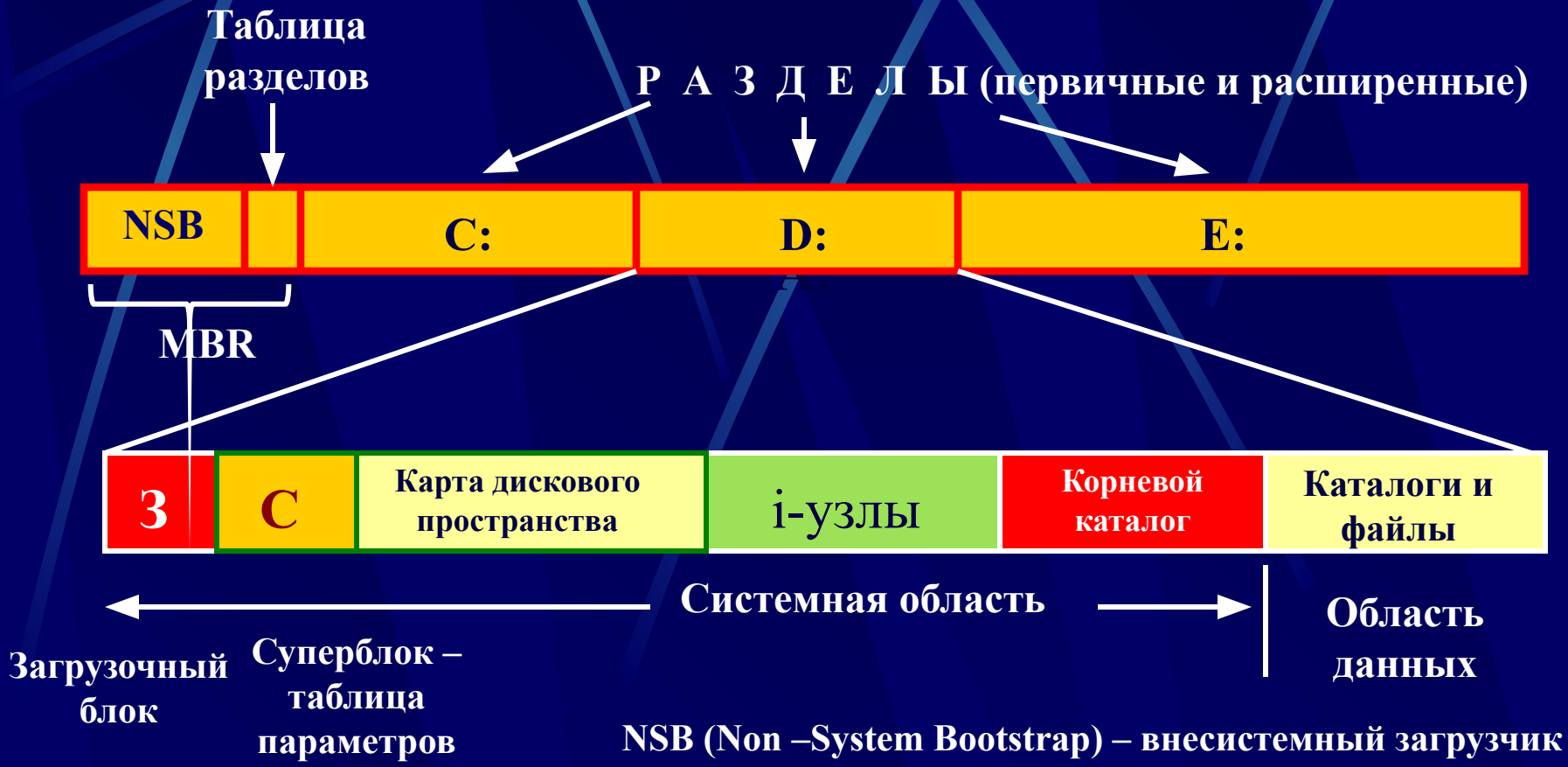


Общая файловая система после монтирования

4.4.3. Физическая организация файловой системы

Структура диска: пластины, дорожки, цилиндры, секторы, кластеры.
Низкоуровневое форматирование – создание дорожек и секторов.

Высокоуровневое форматирование – создание разделов и кластеров для определенной файловой системы или нескольких файловых систем.



Адресация блоков данных диска

1 способ: c – h - s

c – номер цилиндра,

h – номер головки,

s – номер сектора

2 способ: LBA

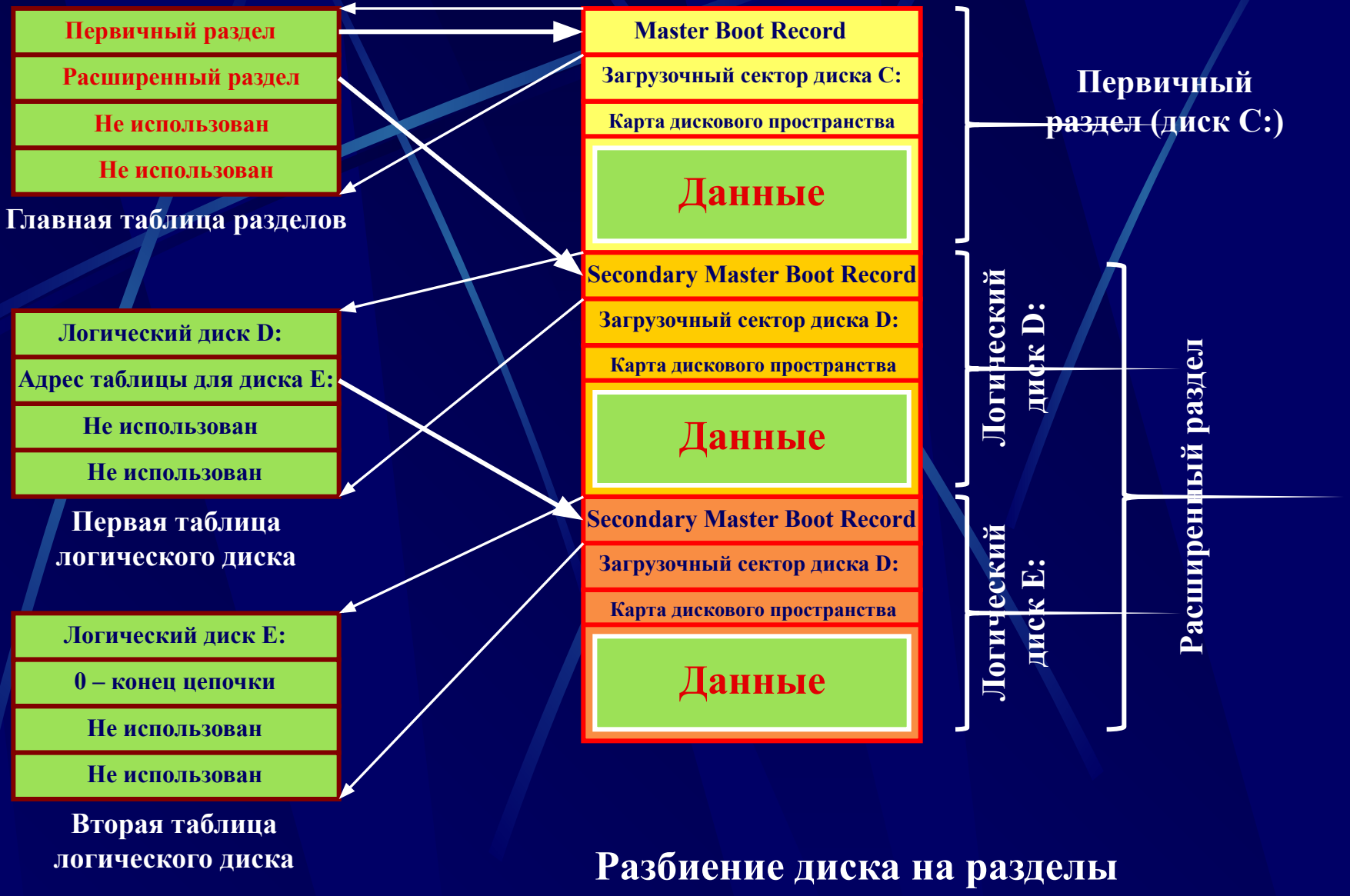
= (c * H + h) * S + s – 1 H

число рабочих поверхностей в цилиндре, S – количество секторов на дорожке

Системные идентификаторы: 06h – FAT16, 07h – NTFS, 0Bh – FAT32

Структура элемента таблицы разделов

N п/п	Назначение	Размер в байтах
1.	Флаг активности раздела (Boot Indicator)	1
2.	Номер головки начала раздела	1
3.	Номер сектора и цилиндра загрузочного сектора раздела	2
4.	Системный идентификатор, показывающий на принадлежность к ОС и ФС	1
5.	Номер головки конца раздела	1
6.	Номер сектора и цилиндра последнего сектора раздела	2
7.	Младшее и старшее двухбайтовые слова относительного номера начального сектора	4
8.	Младшее и старшее двухбайтовые слова размера раздела в секторах	4
9.	Сигнатура-признак MBR и загрузочных секторов – 55AA h (только в конце MBR)	2



Разбиение диска на разделы

Физическая организация и адресация файла

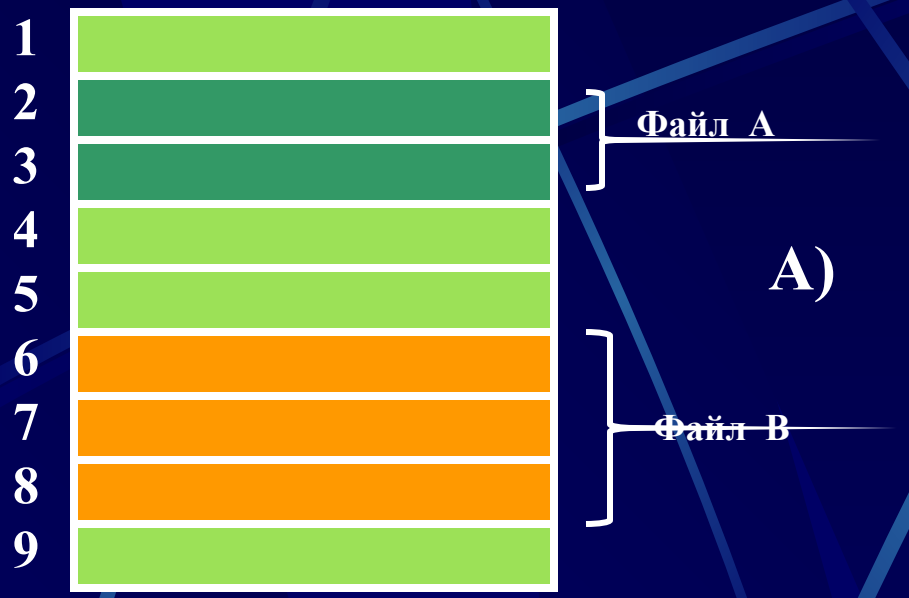
Критерии эффективности физической организации файла:

- ✔ скорость доступа к данным;
- ✔ объем адресной информации файла;
- ✔ степень фрагментированности дискового пространства;
- ✔ максимально возможный размер файла.

Возможные схемы размещения файлов:

- ❑ - непрерывное размещение (непрерывные файлы);
- ❑ - связный список блоков (кластеров) файла;
- ❑ - связный список индексов блоков (кластеров) файла;
- ❑ - перечень номеров блоков (кластеров) файлов;
- ❑ - структуры, называемые I-узлами (index-node – индекс-узел).

Непрерывное размещение

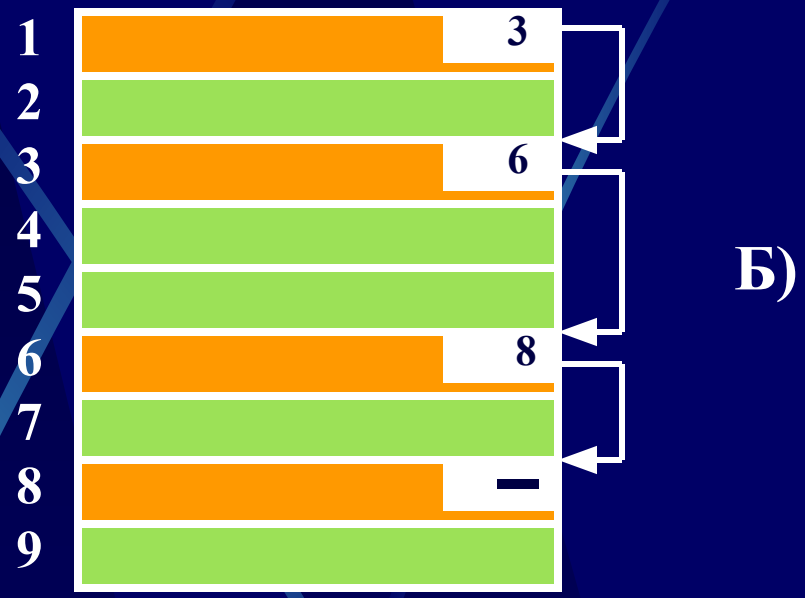


Достоинства: высокая скорость доступа, минимальный объем адресной информации, нет ограничений на размер файла.

Недостатки: нет возможностей для изменения размера файла, высокая степень возможной внешней фрагментации

Область применения – компакт-дики

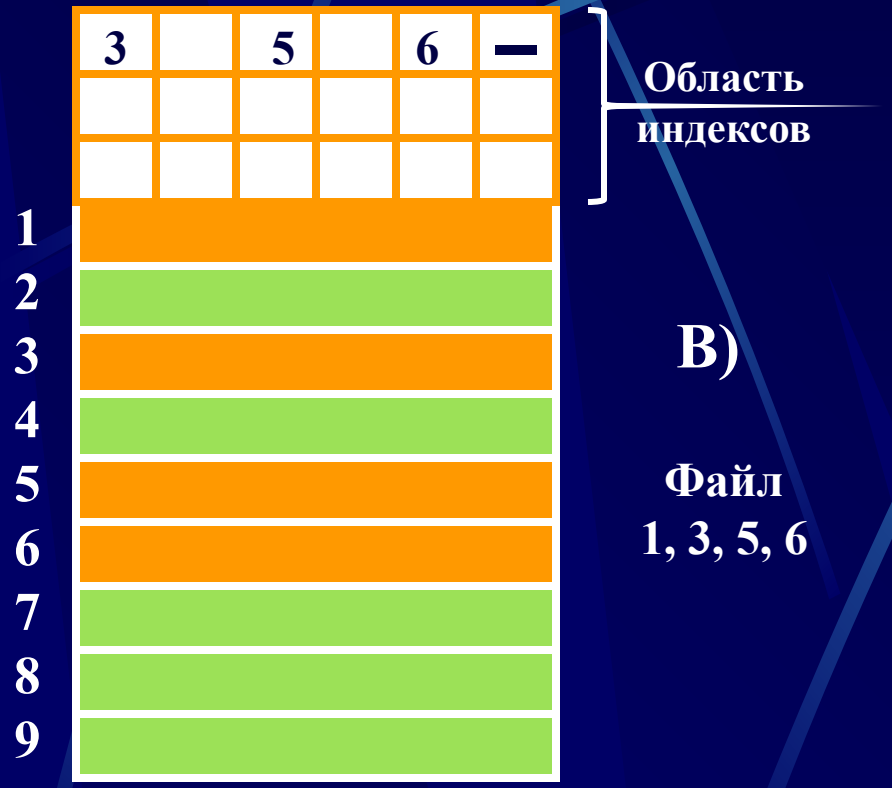
Связный список кластеров



Достоинства: минимальная адресная информация, отсутствие внешней фрагментации, возможность изменения размеров файла.

Недостатки: медленный доступ, сложность доступа к произвольному блоку файла, некратность блока файла степени двойки.

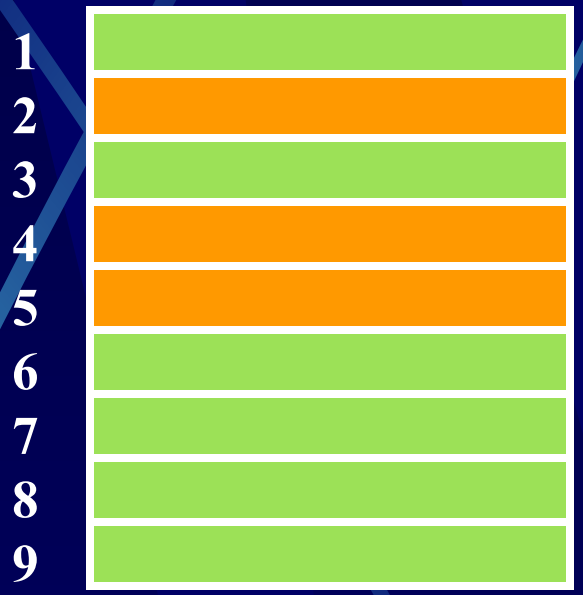
Связный список индексов



В)

Файл 1, 3, 5, 6

Перечень номеров кластеров



Файл 2, 4, 5

Г)

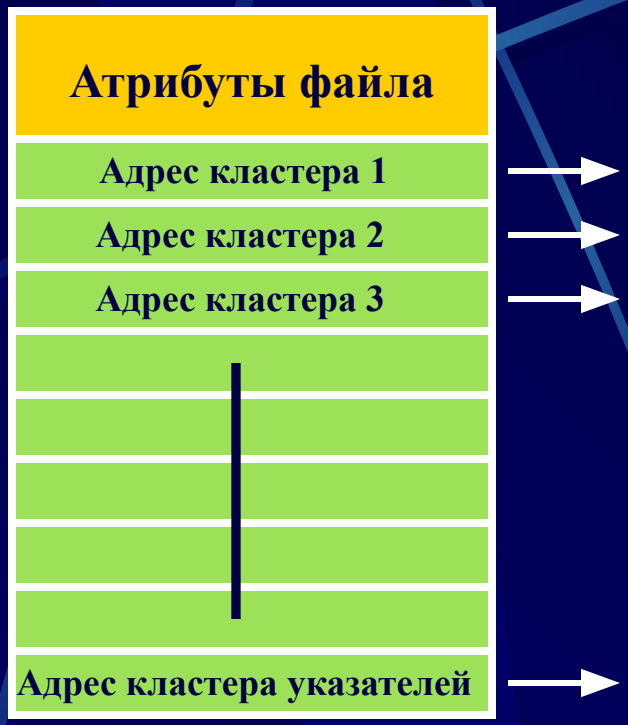
Все достоинства варианта А), быстрый доступ к произвольному кластеру файла, полное заполнение кластера, кратное степени двойки

Недостаток: рост адресной информации с увеличением емкости диска

Достоинства: высокая скорость доступа к произвольному кластеру благодаря прямой адресации, отсутствие внешней фрагментации.

Недостаток: длина адреса зависит от размера файла и может быть значительной.

I- узел (index node)

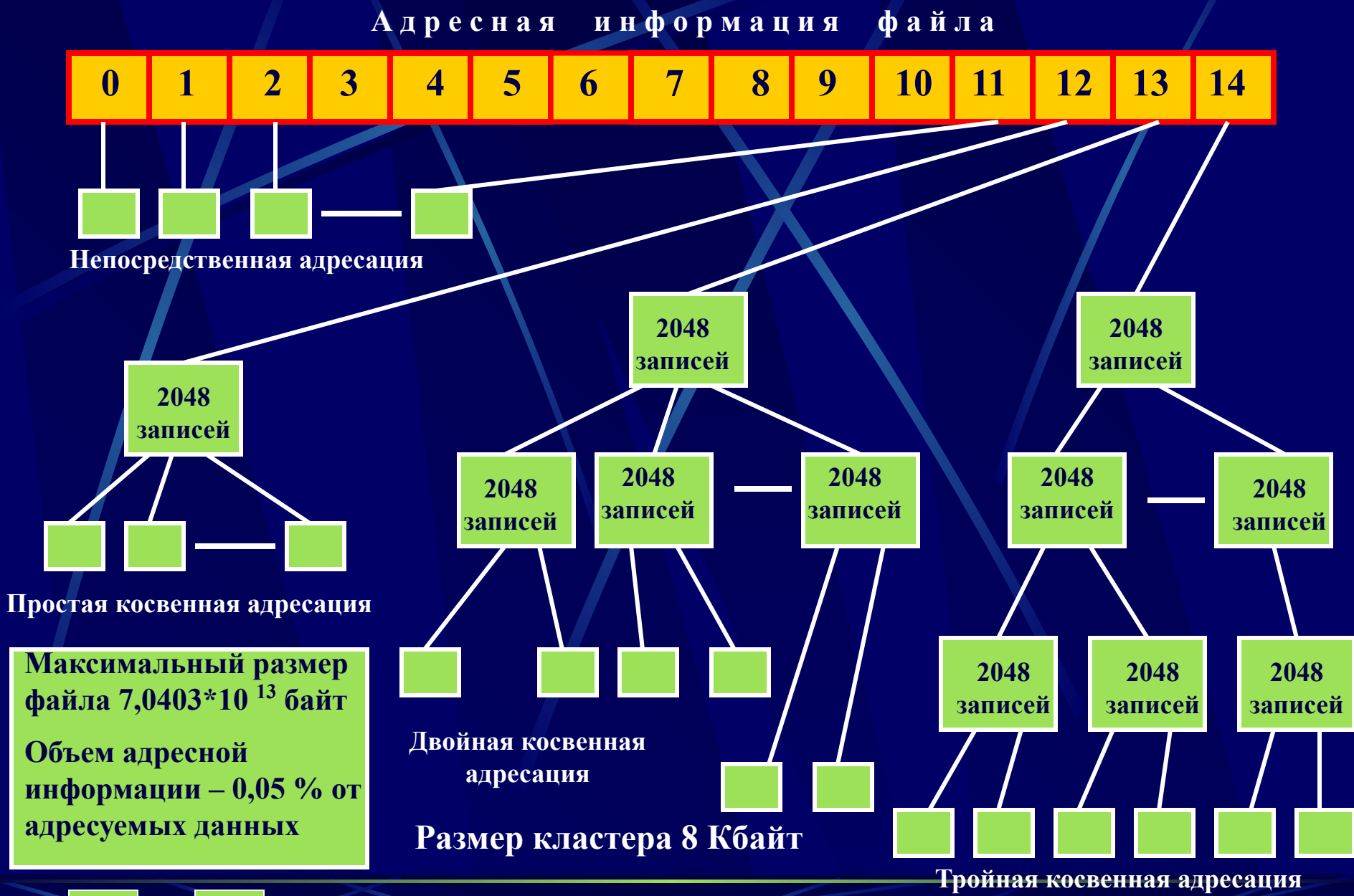


Достоинства: I-узел находится в памяти только при открытии файла, что сокращает объем адресной информации; объем адресной информации не зависит от емкости диска, а лишь от числа открытых файлов; высокая скорость доступа к произвольному кластеру файла благодаря прямой адресации.

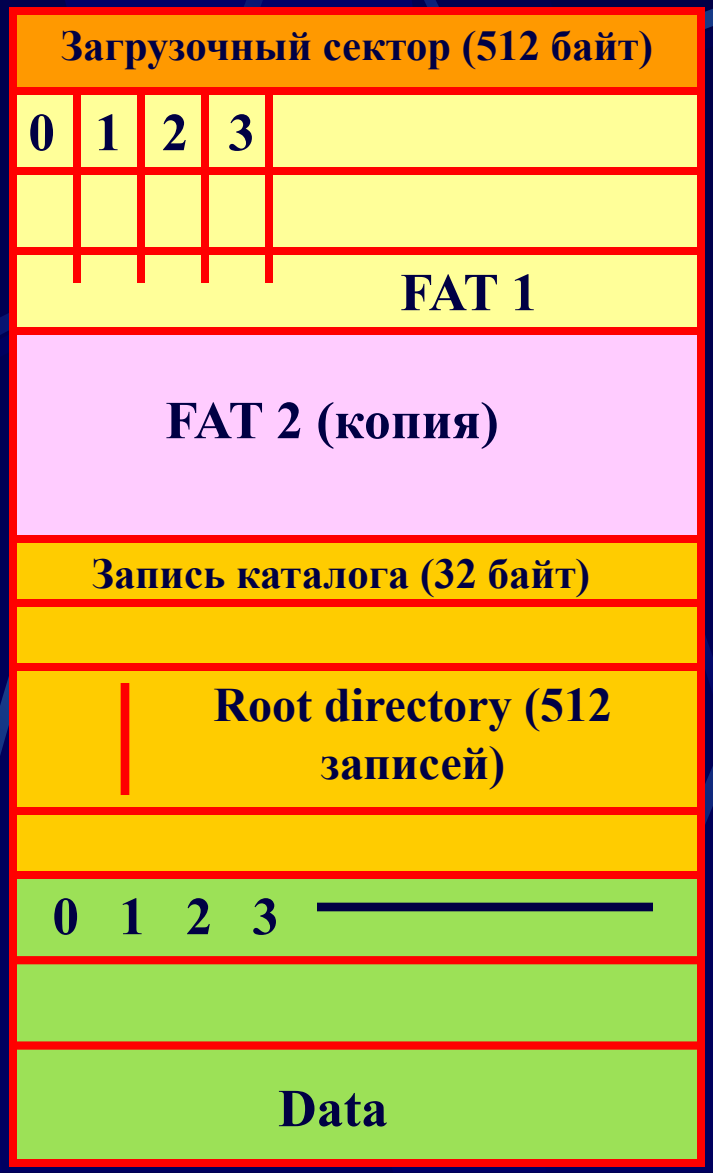
Недостатки: фиксированного количества адресов может оказаться недостаточным для адресации файла, отсюда необходимость сочетания прямой и косвенной адресации

Кластер, содержащий дополнительные дисковые адреса

Файловая система ОС UNIX ufs



Физическая организация FAT



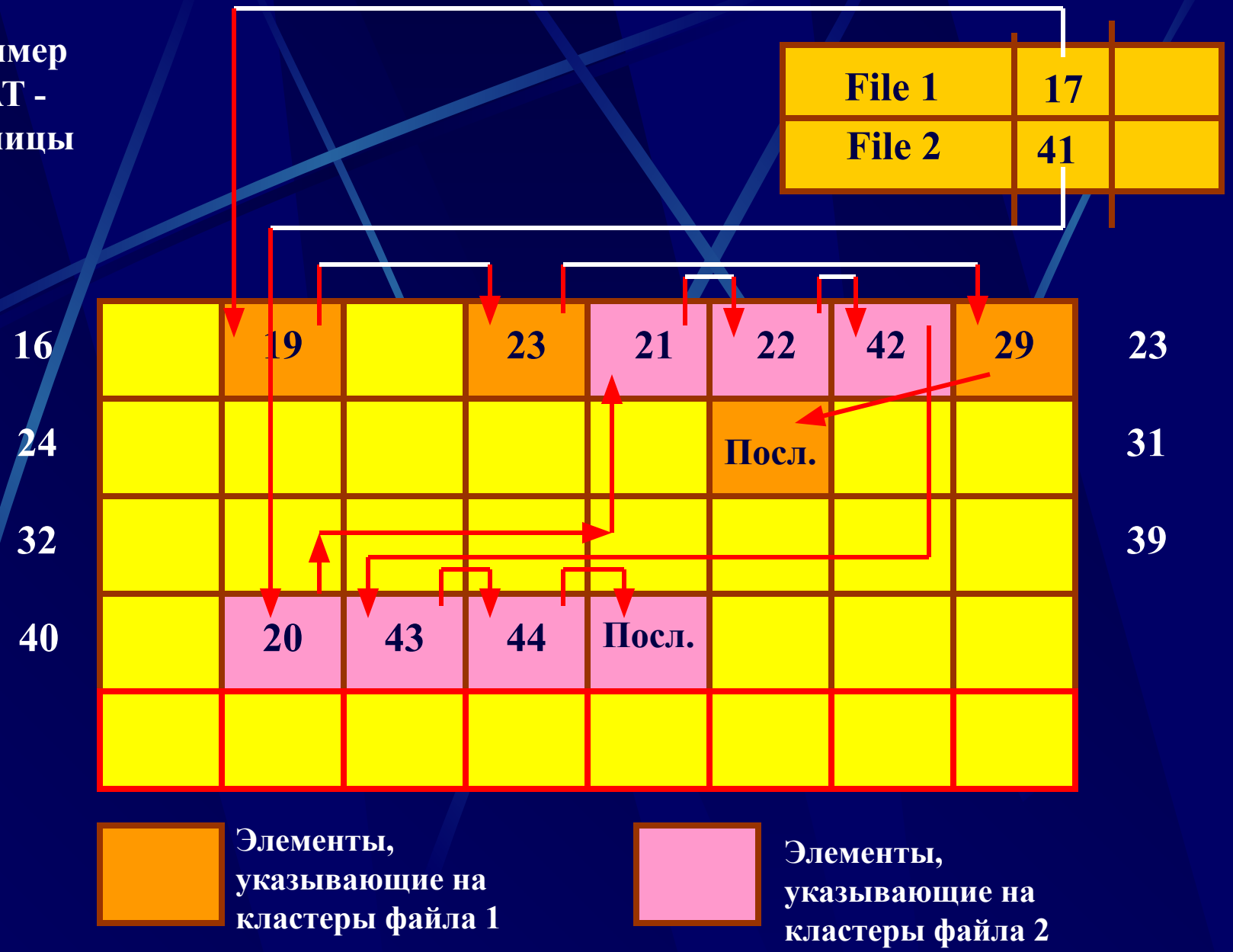
Индексные указатели, связанные с кластерами принимают значения:

кластер свободен (0000h); последний кластер файла (fff8h – ffffh); кластер поврежден (fff7h); резервный кластер (fff0h - fff6h)

Формат каталога

Длина поля	Описание
8 байт	Имя файла
3 байт	Расширение файла
1 байт	Атрибуты файла
1 байт	Зарезервировано
3 байт	Время создания
2 байт	Дата создания
2 байт	Дата последнего доступа
2 байт	Зарезервировано
2 байт	Время последней модификации
2 байт	Дата последней модификации
2 байт	Начальный кластер
4 байт	Размер файла

Пример FAT - таблицы



Основные характеристики файловых систем

FAT	Разрядность указателя	Число кластеров	Максимальный объем кластера	Максимальный размер раздела	Имя файла
FAT12	12	4096	4 Кбайт	16 Мбайт	8.3
FAT16 255.3	16	65536	64 Кбайт	4 Гбайт	8.3
FAT 32	32	4 Г	32 Кбайт	2^{32} по 32 Кбайт	255.3
NTFS	64	2^{64}	4 Кбайт	2^{64} по 4 Кбайт	255.3

Программа Fdisk автоматически определяет размер кластера на основе выбранной файловой системы и размера раздела. Существует недокументированный параметр команды Format, позволяющий явно указать размер кластера:

Format /z:n, где n – размер кластера в байтах, кратный 512.

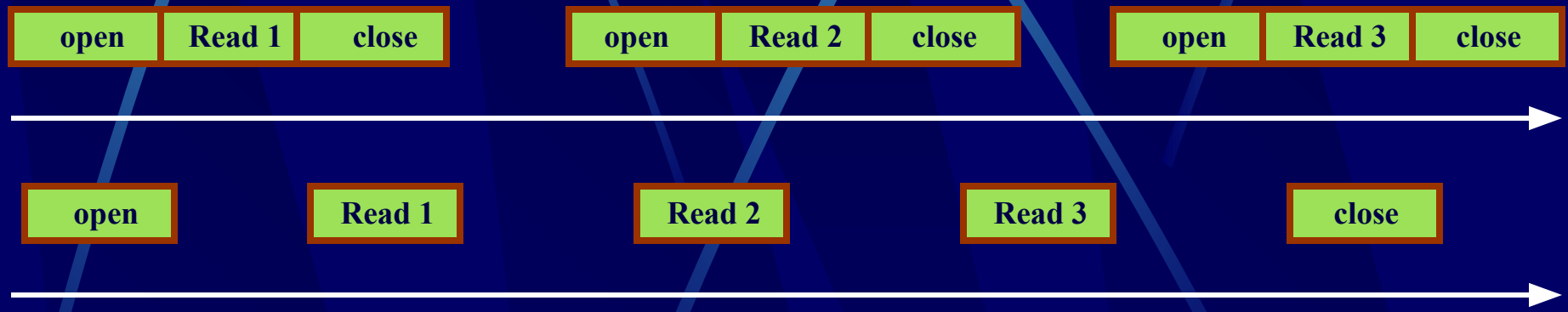
4.4.6. Операции управления каталогами и файловые операции

Win32 API	UNIX	Описание
CreateDirectory	mkdir	Создать новый каталог
RemoveDirectory	rmdir	Удалить пустой каталог
FindFirstFile	opendir	Инициализация для начала чтение записей каталога
FindNextFile	readdir	Прочитать следующую запись каталога
MoveFile	rename	Переместить файл из одного каталога в другой
SetCurrentDirectory	chdir	Изменить текущий рабочий каталог
CreateFile	open	Создать (открыть) файл, вернуть дескриптор файла
DeleteFile	unlink	Удалить существующий файл
CloseHandle	close	Закрыть файл
ReadFile	read	Прочитать данные из файла
WriteFile	write	Записать данные в файл
SetFilePointer	lseek	Уст-вить указатель в файле в определенную позицию
GetFileAttributes	stat	Вернуть атрибуты файла
LockFile	fcntl	Заблокировать файл для взаимного исключения
Unlock File	fcntl	Отменить блокировку области файла

Способы выполнения файловых операций

Последовательность универсальных действий:

1. По символьному имени файла найти его характеристики, которые хранятся в файловой системе на диске.
2. Скопировать характеристики файла в оперативную память, поскольку только в этом случае программный код может их использовать.
3. На основании характеристик файла проверить права пользователя на выполнение запрошенной операции (чтение, запись, удаление и т. п.).
4. Очистить область памяти, отведенную под временное хранение характеристик файла.



Примеры системных вызовов для работы с файлами:

fd = create (“abc”, mode); fd = open (“file”, how);
read (fd, buffer, nbytes); write(fd, buffer, nbytes);

Стандартные файлы ввода – вывода, перенаправление вывода

read (stdin, buffer, nbytes); write(stdout, buffer, nbytes);
< file - перенаправление ввода,
> file – перенаправление вывода на файл

Примеры системных вызовов для работы с файлами

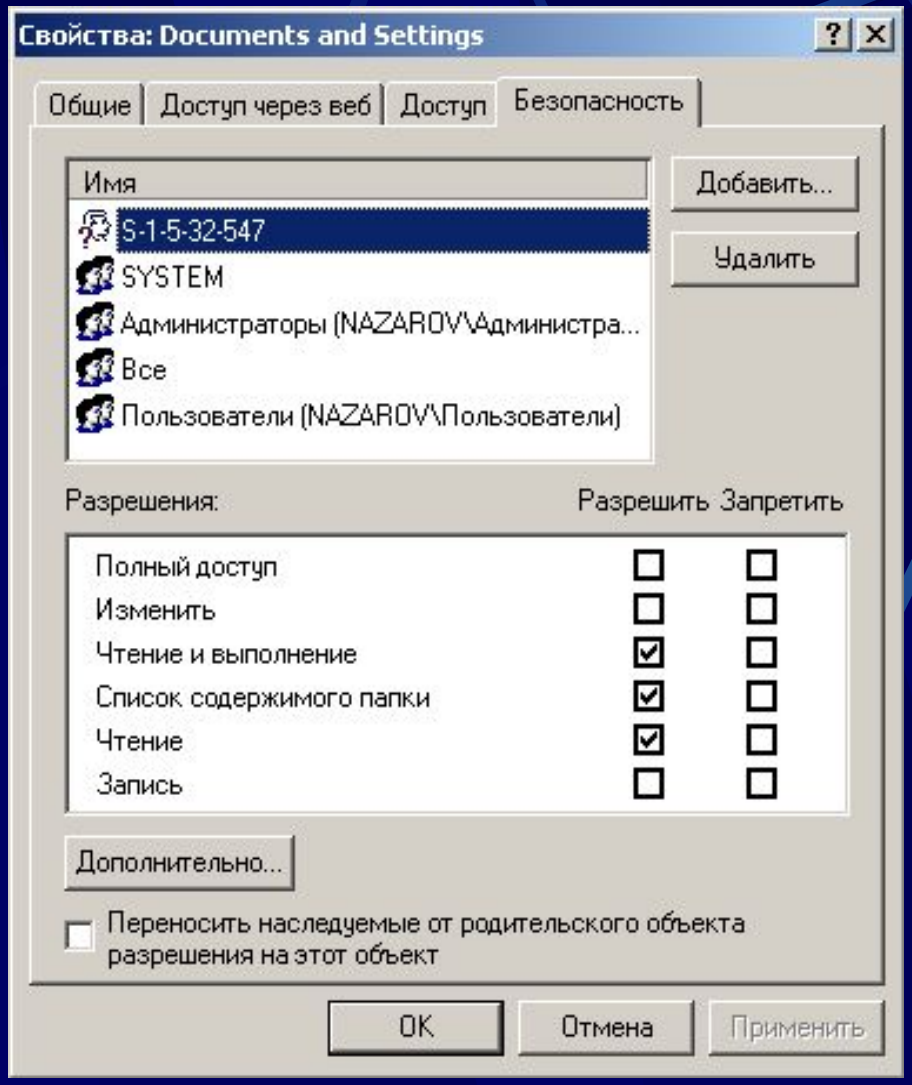
fd = creat (“name”, mode) – файла с заданным режимом защиты;
fd = open (“name”, how) – открыть файл для чтения, записи или и того и другого;
s = close (fd) – закрыть открытый файл;
n = read (fd, buffer, nbytes) – прочитать данные из файла в буфер;
n = write (fd, buffer, nbytes) – записать данные из буфера в файл;
position = lseek (fd, offset, whence) – переместить указатель в файле;
s = fstat | stat (fd | “name”, &buf) - получить информацию о состоянии файла.

При выполнении программы стандартным образом файлы с дескрипторами 0, 1 и 2 уже открыты для стандартного ввода, стандартного вывода и стандартного потока сообщений об ошибках.

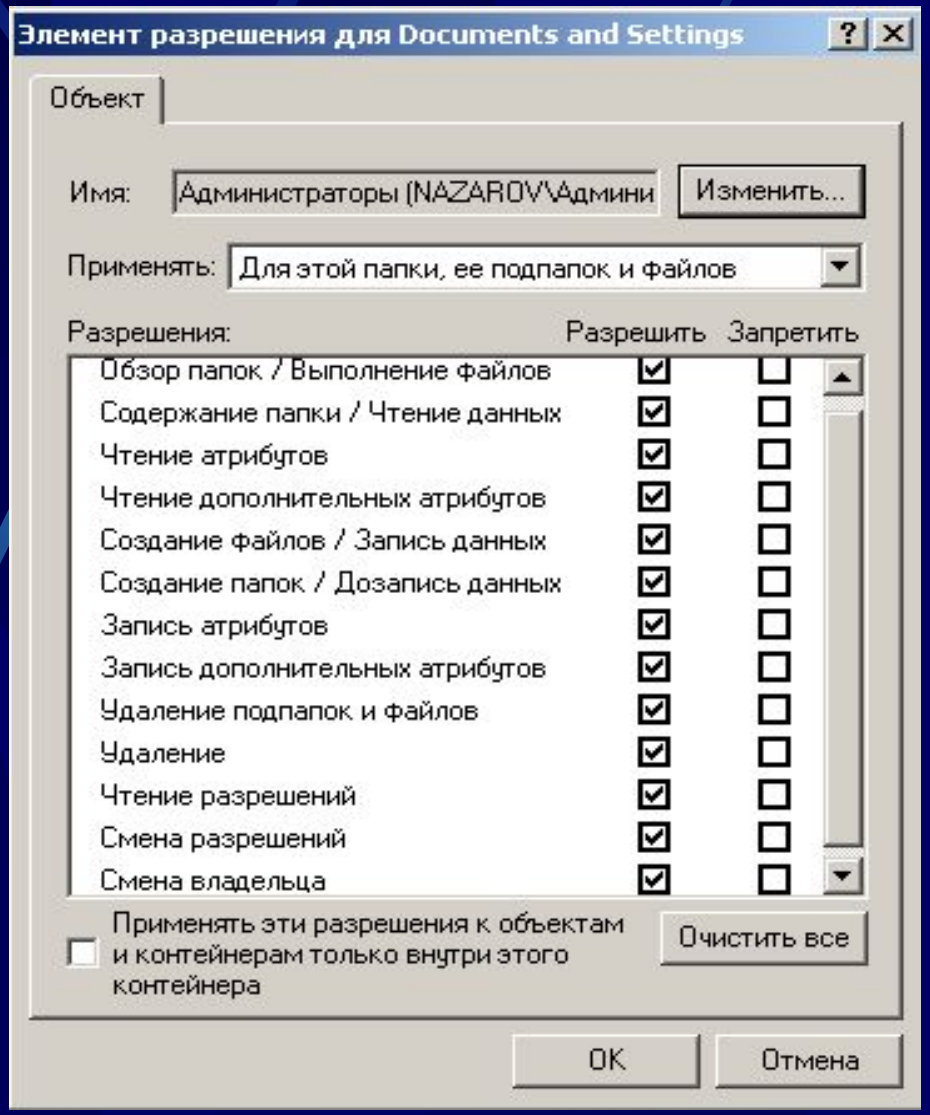
n = read (stdin, buffer, nbytes); n = write (stdout, buffer, nbytes)

stdin = 0; stdout =1; stderr = 2.

Разрешения на доступ к каталогам

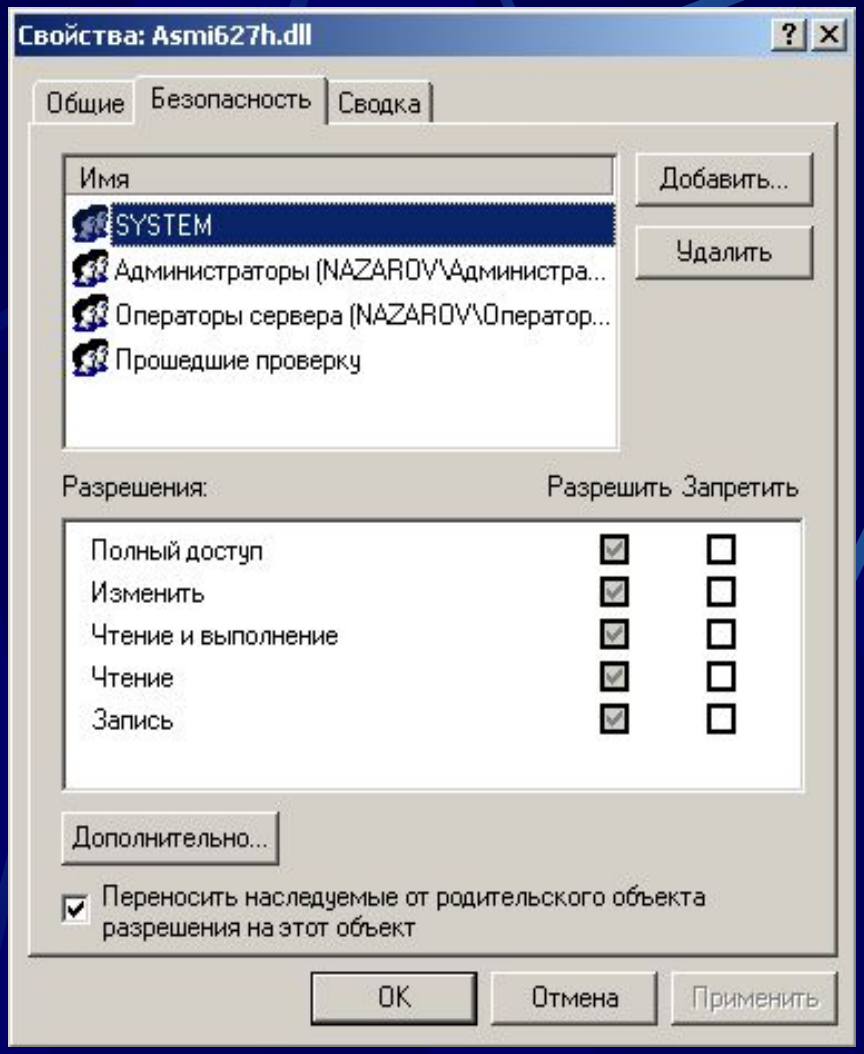


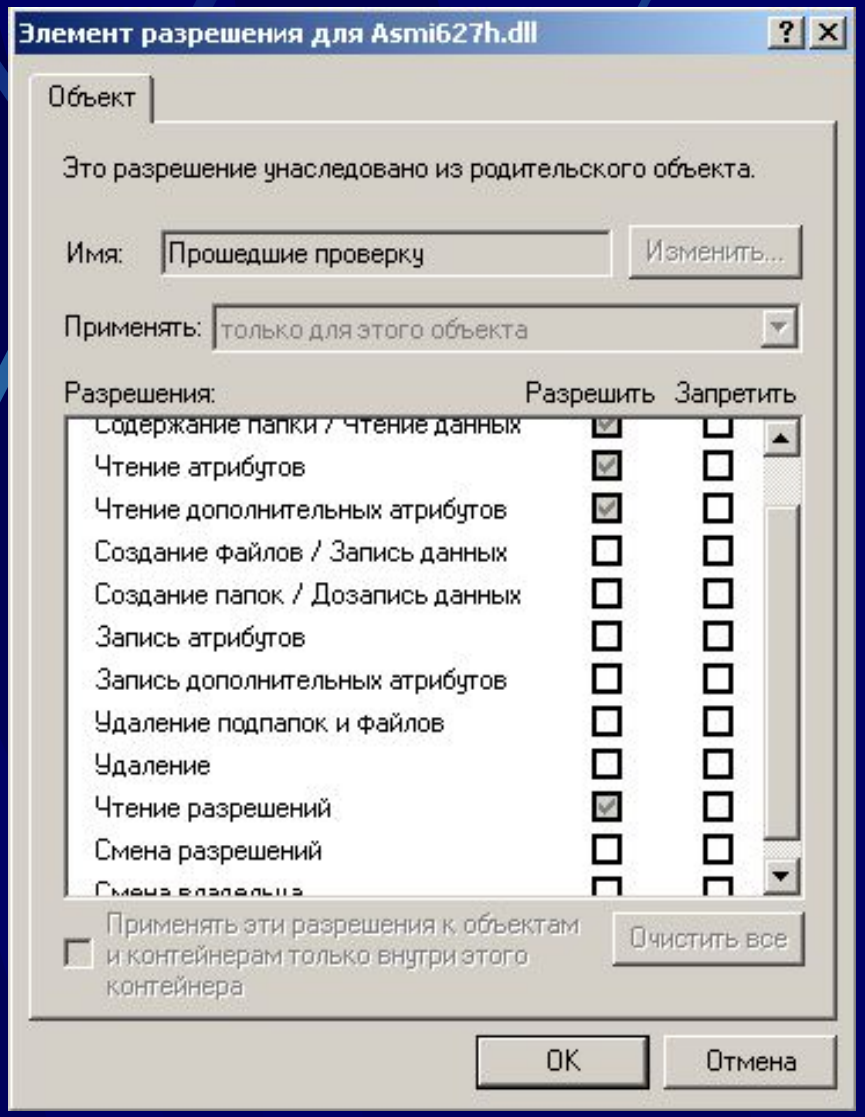
Стандартные разрешения



Специальные разрешения

Разрешения на доступ к файлам





Квоты дискового пространства