**Ответ на вопрос 1: Общие сведения об операционных системах, цели применения ОС, структура ОС**

**Общие сведения об операционных системах**

Операционная система (ОС) представляет собой комплекс программ, предназначенных для управления вычислительной системой, включая планирование задач, распределение ресурсов, управление вводом-выводом информации и данными. Она является связующим компонентом между пользователем, прикладным программным обеспечением и аппаратным обеспечением компьютера (Рис. 1)



Основные функции операционных систем:

* Управление вычислениями.
* Планирование и распределение ресурсов.
* Управление вводом-выводом и данными.

ОС включает ядро, прикладные программные интерфейсы (API) и системные утилиты, которые могут загружаться по мере необходимости

**Цели применения ОС**

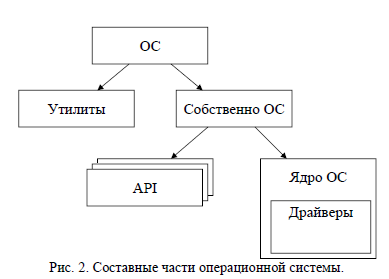
1. Увеличение пропускной способности компьютера (количество задач в единицу времени).
2. Уменьшение времени реакции системы.
3. Контроль работоспособности аппаратных и программных компонентов.
4. Обеспечение пользовательского интерфейса.
5. Управление программами и данными в процессе вычислений.
6. Адаптивность к различным аппаратным средствам.
7. Предоставление интерфейса для прикладных программ (API)

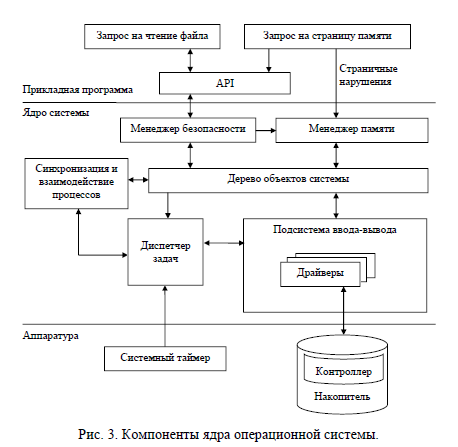
**Структура операционной системы**

ОС состоит из следующих компонентов:

* **Ядро операционной системы**, включающее:
  + Менеджер безопасности.
  + Менеджер памяти.
  + Подсистему синхронизации и взаимодействия процессов.
  + Дерево объектов системы.
  + Диспетчер задач.
  + Подсистему ввода-вывода.

Все компоненты ядра взаимодействуют между собой для обеспечения работоспособности системы (Рис. 2 и Рис. 3)





### Ответ на вопрос 2: Модель взаимодействия компонентов операционной системы

Модель взаимодействия компонентов операционной системы определяется их функциями и взаимодействием через ядро системы. Ключевые компоненты ядра ОС включают:

1. **Менеджер безопасности** — проверяет доступность ресурсов и прав.
2. **Менеджер памяти** — управляет виртуальной и физической памятью.
3. **Подсистема синхронизации и взаимодействия процессов** — обеспечивает совместное использование ресурсов.
4. **Дерево объектов системы** — организует хранение и доступ к объектам ОС.
5. **Диспетчер задач** — планирует выполнение процессов.
6. **Подсистема ввода-вывода** — обрабатывает запросы на доступ к внешним устройствам.

Эти компоненты взаимодействуют между собой для обеспечения согласованной работы системы (Рис. 3 на стр. 11)

#### Последовательность взаимодействия

1. **Инициирование действий**: Процессы взаимодействуют с ОС через системные вызовы.
2. **Обработка запросов**:
   * Диспетчер задач переключает процессор между процессами, основываясь на приоритетах.
   * Менеджер безопасности проверяет параметры вызова и права доступа.
   * Подсистема ввода-вывода передает запросы драйверам.
3. **Завершение обработки**:
   * Драйверы взаимодействуют с аппаратурой, а затем возвращают управление системе.
   * Менеджер памяти выделяет дополнительные ресурсы, если это требуется

### Ответ на вопрос 3: Классификация операционных систем

Операционные системы (ОС) классифицируются по различным признакам:

1. **Общее и специальное назначение**:
   * **Общего назначения**: предназначены для широкого круга задач.
   * **Специального назначения**: для переносных устройств, встроенных систем, баз данных, систем реального времени и других приложений.
2. **Режим обработки задач**:
   * **Однопрограммные**: выполнение одной задачи в один момент времени.
   * **Мультипрограммные**: создается видимость одновременного выполнения нескольких задач, что особенно полезно для однопроцессорных систем.
3. **Организация работы в диалоговом режиме**:
   * **Однотерминальные**: работа с одним пользователем.
   * **Мультитерминальные**: поддерживают работу нескольких пользователей одновременно.
4. **Ограничение времени реакции**:
   * **Общего назначения**: нет строгих временных ограничений.
   * **Системы реального времени**:
     + Мягкое реальное время (допустимы небольшие задержки, например, до 10 мс).
     + Жесткое реальное время (требуется мгновенная реакция).
5. **Способ организации ядра**:
   * **Монолитное ядро**: включает все основные функции в одном компоненте (например, Windows).
   * **Микроядро**: минимальное ядро с переносом большинства функций в пользовательское пространство (например, QNX).
   * **Гибридное ядро**: сочетает элементы монолитного и микроядра, включая возможность загрузки и выгрузки драйверов

### Ответ на вопрос 4: Теоретические основы параллельного программирования (понятия: процесс, задача, пропускная способность системы)

#### Процесс (process)

Процесс представляет собой выполнение программы с её данными на последовательном процессоре. Это базовая единица выполнения, которая может включать в себя одну или несколько задач. Процессы могут быть независимыми или взаимодействующими в зависимости от их доступа к общим переменным и ресурсам

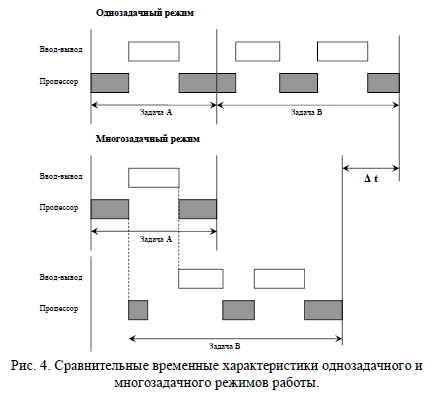
#### Задача (task)

Задача определяется как совокупность программных модулей и данных, требующая ресурсов для выполнения. Этот термин впервые был введён специалистами IBM в рамках разработки многозадачного режима работы компьютера

#### Пропускная способность системы

Пропускная способность системы измеряется количеством стандартных задач, которые система может выполнить за единицу времени. Переход от однозадачного к многозадачному режиму работы позволяет увеличить пропускную способность системы за счёт эффективного использования процессора и минимизации времени простоя

Для улучшения понимания различий между режимами работы системы можно ознакомиться с рисунком 4



### Ответ на вопрос 5: Теоретические основы параллельного программирования: ресурс, концепция виртуальных ресурсов, нить, контекст нити

#### Ресурс

Ресурсом называется любой объект, который может распределяться внутри системы с течением времени.

Ресурсы бывают:

* **Разделяемыми**: используются несколькими процессами одновременно.
* **Неделимыми**: процессы используют их по очереди.

Примеры ресурсов включают процессорное время, память, устройства ввода-вывода, а также системные компоненты, такие как семафоры или таймеры

#### Концепция виртуальных ресурсов

Концепция виртуальных ресурсов позволяет унифицировать разработку программ и повысить эффективность использования ресурсов. Операционная система создаёт виртуальные ресурсы, моделируя:

* Виртуальные процессоры на основе реальных CPU.
* Виртуальную память, организованную с помощью динамического преобразования адресов.
* Виртуальные устройства ввода-вывода, работающие параллельно

#### Нить

Нить — это виртуальный процессор, который функционирует в адресном пространстве процесса и выполняет свою задачу. Особенности нитей:

* Они разделяют адресное пространство, глобальные переменные, а также доступ к файлам и устройствам ввода-вывода.
* Реальная параллельность возможна только при наличии нескольких процессорных ядер.

#### Контекст нити (различающиеся параметры)

Контекст нити включает:

1. Регистры процессора (включая программный счётчик).
2. Стек.
3. Текущее состояние нити.
4. Соотношение родитель-потомок

### Ответ на вопрос 6: Теоретические основы параллельного программирования: классификация программных модулей, реентерабельность, повторная входимость

#### Классификация программных модулей

Программные модули делятся на:

1. **Однократно используемые** — разрушают себя в процессе выполнения и не подходят для параллельного программирования, обычно используются при загрузке системы.
2. **Многократно используемые**:
   * **Привилегированные модули** — блокируют ресурсы процессора, отключая прерывания. Используются в ядре ОС.
   * **Непривилегированные модули** — могут быть прерваны. В свою очередь, делятся на:
     + **Нереентерабельные** — используют глобальные переменные. Их повторный вызов приводит к сбоям.
     + **Реентерабельные** — используют только локальные переменные, что позволяет их безопасно вызывать параллельно

#### Понятие реентерабельности

Реентерабельные модули не зависят от глобальных данных. Примером является функция abs() для вычисления модуля числа. Реентерабельные модули обеспечивают более высокий уровень безопасности при параллельном программировании, но их реализация сложнее. Практически невозможно создать на практике

#### Повторная входимость

Повторно-входимые модули содержат критические секции, которые блокируют выполнение другими нитями на определённых участках. Это наиболее часто используемые модули в системах. Примером является функция fopen() из библиотеки POSIX

### Ответ на вопрос 7: Понятие прерывания. Дисциплины прерываний. Механизм обработки прерываний однозадачной и многозадачной ОС (страницы 18-19)

#### Понятие прерывания

Прерывание — это механизм, который позволяет координировать параллельную работу устройств вычислительной системы и реагировать на особые состояния. При прерывании процессор передаёт управление от текущей программы к специальному обработчику прерываний, что позволяет реализовать асинхронный режим работы и параллельность функционирования системы.

Обработка прерывания включает:

1. Установление факта прерывания.
2. Сохранение состояния прерванного процесса (аппаратного контекста).
3. Передачу управления обработчику прерываний.
4. Восстановление состояния прерванного процесса.

#### Дисциплины обработки прерываний

1. **С относительными приоритетами** — текущая обработка прерывания не прерывается даже более приоритетным запросом.
2. **С абсолютными приоритетами** — всегда обслуживаются прерывания с наивысшим приоритетом.
3. **По принципу стека (LCFS)** — обслуживается последнее поступившее прерывание.

#### Механизм обработки в однозадачной ОС

Обработка выполняется последовательно, без переключения задач. Прерывания в основном управляют взаимодействием процессора с периферией.

#### Механизм обработки в многозадачной ОС

Прерывания используются не только для работы с аппаратурой, но и для управления задачами, включая:

* Сохранение контекста задачи.
* Диспетчеризацию задач.
* Переключение задач до завершения обработки прерывания

### Ответ на вопрос 8: Диспетчер задач: дисциплины диспетчеризации (страницы 18-24)

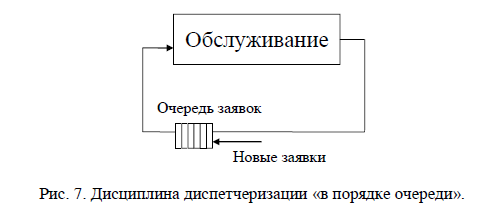
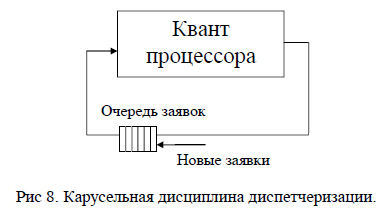
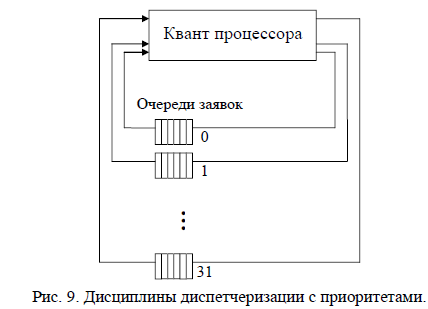
#### Диспетчер задач

Диспетчер задач является компонентом ядра операционной системы, который обеспечивает переключение процессора между задачами. От реализации диспетчера задач зависят:

* Время реакции системы.
* Скорость выполнения приоритетных нитей.
* Общая эффективность операционной системы.

#### Дисциплины диспетчеризации

Иерархия дисциплин диспетчеризации представлена на рисунке 6 (страница 20). Основное деление:

1. **Бесприоритетные**:
   * Линейные (в порядке очереди, случайные)  
     
   * Циклические (карусельная - ни одна заявка не может привести к блокировке, так как обрабатывается не более чем заданный интервал времени, который называется квантом (20-30 мс))  
     
2. **Приоритетные**:
   * С фиксированными приоритетами (с относительным, абсолютным приоритетом, адаптивное обслуживание)   
     
   * С динамическими приоритетами (с зависимостью от времени ожидания, с зависимостью от времени обслуживания)

### Ответ на вопрос 9: Аппаратные и программные приоритеты в операционной системе Windows (страницы 18-24)

#### Аппаратные приоритеты

Аппаратные приоритеты в Windows разделены на уровни прерываний IRQL (Interrupt Request Levels), которые используются ядром ОС для обработки аппаратных событий:

* Более приоритетный интервал отвечает за работу ядра системы, в нем используется дисциплина с абсолютными приоритетами и невытесняющей многозадачностью
* Менее приоритетный интервал предназначен для прикладных процессов. В нем используется карусельная дисциплина диспетчеризации с приоритетами

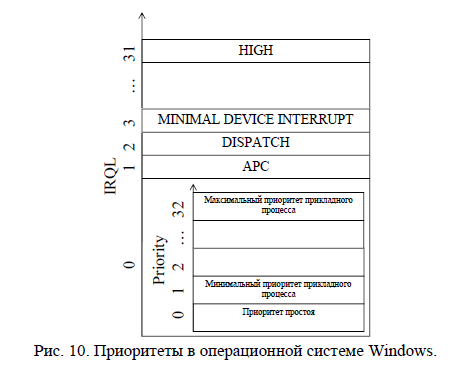
#### Программные приоритеты

Программные приоритеты в Windows разделены на 32 уровня:

* Уровни с 1 до 15 предназначены для обычных процессов.
* Уровни с 16 до 31 зарезервированы для процессов реального времени.
* Приоритет простоя имеет уровень 0.

#### Особенности диспетчеризации в Windows

* **Динамические приоритеты**:
  + Изменяются в зависимости от состояния нити (например, после длительного ожидания или выполнения активного окна).
  + Служат для улучшения реакции системы на пользовательские действия.
* **Адаптивность обслуживания**:
  + Нити с активным окном получают повышение приоритета.
  + Если нить долго выполняется, её приоритет постепенно уменьшается.



### Ответ на вопрос 10: Теоретические основы параллельного программирования: независимые и взаимодействующие процессы, понятия ресурса и критического ресурса (страницы 27-30)

#### Независимые и взаимодействующие процессы

* **Независимые процессы**: не имеют общих переменных или состояния, их выполнение не влияет друг на друга.
* **Взаимодействующие процессы**: имеют доступ к общим переменным, могут обмениваться данными. Взаимодействие может быть организовано в двух формах:
  + **Конкуренция**: Процессы конкурируют за общий ресурс.
  + **Сотрудничество**: Один процесс передаёт результаты другому (схема "поставщик-потребитель").

#### Понятие ресурса

Ресурс — объект, который может быть использован процессом. Он может быть:

* **Разделяемым**: Одновременный доступ несколькими процессами.
* **Критическим**: используется только одним процессом в каждый момент времени. Остальные ожидают его освобождения**. Критический ресурс** требует синхронизации для предотвращения конфликтов. **Критическая секция** — часть программы, где организован доступ к критическому ресурсу. Некорректная работа с критической секцией приводит к ошибкам и сбоям программ

**Ответ на вопрос 11: Условия функционирования конкурирующих процессов, понятие тупика, задача об обедающих философах (страницы 28-30)**

**Условия функционирования конкурирующих процессов**

Для корректной работы конкурирующих процессов необходимо соблюдение следующих условий:

1. **Взаимное исключение** — каждый процесс может быть в критической секции один.
2. **Отсутствие блокировки** — процесс не должен бесконечно находиться в критической секции.
3. **Возможность входа.** Ни один процесс не должен ждать бесконечно долго входа в критическую секцию (в том числе, решение, какой из процессов должен занять КС, не должно откладываться бесконечно долго).
4. **Вход по завершению процесса.** Если процесс, захвативший КС, завершается, в том числе аварийно, то КС должна освободиться

**Понятие тупика**

Тупик (deadlock) возникает, если процессы блокируют друг друга, ожидая освобождения ресурсов. Классический пример тупика:

* Один процесс захватывает ресурс A и ожидает освобождения ресурса B.
* Другой процесс захватывает ресурс B и ожидает ресурс A.

Для предотвращения тупиков используются стратегии:

* Последовательное захватывание ресурсов.
* Избежание круговой зависимости в запросах ресурсов.

**Задача об обедающих философах**

Задача иллюстрирует проблему синхронизации доступа к критическим ресурсам:

* Пять философов сидят за столом, между ними расположены вилки (критические ресурсы).
* Для принятия пищи философу требуются две вилки.
* Если философы начнут захватывать вилки одновременно, может возникнуть тупик.

Решения задачи:

* Изменение порядка захвата вилок (например, всегда начинать с вилки слева).
* Использование одного из философов для контроля доступности вилок (централизованный подход).

### Ответ на вопрос 12: Принцип организации критических секций в процессорах Intel и ARM (страницы 30-31)

#### Принцип организации критических секций

Критическая секция — это часть кода, обеспечивающая синхронизированный доступ к критическому ресурсу, чтобы избежать ошибок, вызванных параллельным выполнением процессов.

1. **В процессорах Intel**:
   * Применяется инструкция XCHG, которая выполняет атомарный обмен значениями между регистром и ячейкой памяти.
   * Используется специальный сигнал #LOCK, блокирующий доступ к оперативной памяти для других процессоров или устройств во время выполнения операции.
   * Пример использования (на языке ассемблера):

mov eax, 1

l1: xchg eax, Status

cmp eax, 1

je l1

// Критическая секция

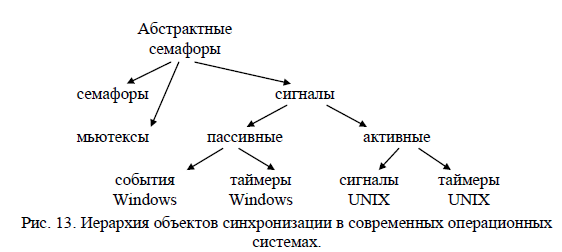
mov Status, 0

1. **В процессорах ARM**:
   * Требуется больше инструкций для организации критических секций, так как здесь используются механизмы управления кешированием данных и барьеры памяти.
   * Процессоры ARM, начиная с платформы ARMv7, поддерживают аналогичные атомарные операции для синхронизации, такие как LDREX и STREX.
2. **Отличия между платформами**:
   * Intel предлагает более простую и эффективную реализацию за счёт инструкции XCHG и встроенной поддержки блокировки на шине управления.
   * ARM использует барьеры памяти и специализированные инструкции для обеспечения согласованности данных.

### Ответ на вопрос 13: Классификация видов взаимодействия процессов (страницы 34-38)

#### Виды взаимодействия процессов:

1. **События**:
   * Представляют собой двоичные семафоры.
   * В Windows возможны два состояния: сигнальное (SIGNALED) и несигнальное (NONSIGNALED).
   * Создаются функцией CreateEvent. Бывают:
     + С ручным сбросом: остаются в сигнальном состоянии после захвата.
     + Автоматические: сбрасываются после захвата
2. **Семафоры**:
   * Могут принимать более двух значений (в отличие от событий).
   * Реализуют счетчик для ограничения доступа к ресурсу.
   * Создаются функцией CreateSemaphore. Функция ReleaseSemaphore увеличивает значение семафора
3. **Мьютексы**:
   * Обеспечивают исключительное владение ресурсом для одного процесса или нити.
   * Мьютекс переходит в сигнальное состояние, если владеющая нить завершает работу.
   * Важное свойство — поддержка повторного захвата одним владельцем
4. **Таймеры ожидания**:
   * Переходят в сигнальное состояние по истечении времени.
   * Создаются функцией CreateWaitableTimer. Настраиваются через SetWaitableTimer. Могут быть периодическими
5. **CRITICAL\_SECTION**:
   * Локальный объект синхронизации для приложений. Не требует обращения к ядру ОС.
   * Позволяет использовать активную блокировку с экономией ресурсов.
   * Функции работы: EnterCriticalSection, LeaveCriticalSection, TryEnterCriticalSection
6. **Процессы и нити**:
   * Синхронизируются через различные объекты (например, события и мьютексы).
   * Могут находиться в сигнальном и несигнальном состояниях



### Ответ на вопрос 14: Классификация видов взаимодействия процессов — барьеры, мониторы, портфель задач, читатели и писатели (страницы 38-39)

#### Барьеры

* Представляют собой объект синхронизации, который переходит в сигнальное состояние только при достижении порогового числа ожидающих процессов.
* Реализация может использовать семафоры и счётчик.
* Пример: при организации через Interlocked-функции счётчик увеличивается каждым процессом, а последний процесс переводит барьер в сигнальное состояние, обнуляя счётчик

#### Мониторы

* Это абстракция, где критический ресурс может обрабатываться через несколько критических секций, организованных в виде методов класса.
* Для синхронизации используются мьютексы.
* Пример задачи: "спящий парикмахер", где потоки ожидают освобождения ресурса или выполнения действия

#### Портфель задач (Thread Pool)

* Предназначен для выполнения параллельных задач.
* Задача передаётся в портфель, откуда выделенная нить выполняет её.
* Применяется в ядре ОС Windows, где задачи портфеля называются WorkItem.

#### Читатели и писатели

* Используются для организации доступа к данным с различной гранулярностью:
  + Одновременное чтение разрешено для нескольких процессов.
  + Модификация доступна только одному процессу.
* Повышение пропускной способности достигается за счёт параллельных операций чтения.
* Реализован в ОС Windows как объект RESOURCE

### Ответ на вопрос 15: Прямая передача данных между процессами: разделяемая память, прямое чтение-запись памяти процесса (страницы 44-45)

#### Разделяемая память

* Является самым быстрым механизмом передачи данных между процессами, достигая скорости в несколько гигабайт в секунду.
* **Организация в Windows**:
  + **Создание области памяти**: CreateFileMapping.
  + **Подключение области памяти к процессу**: MapViewOfFile.
  + **Отключение области памяти**: UnmapViewOfFile.
  + **Закрытие описателя**: CloseHandle.
* Разделяемая память представляет собой набор страниц, которые могут быть отображены в разных процессах. Доступные режимы:
  + READONLY (только чтение).
  + READWRITE (чтение и запись).
  + WRITECOPY (чтение и запись с копированием при модификации).

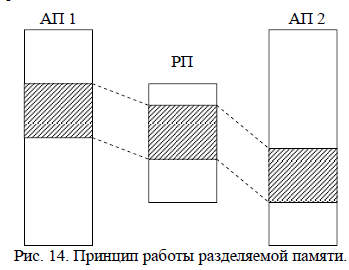
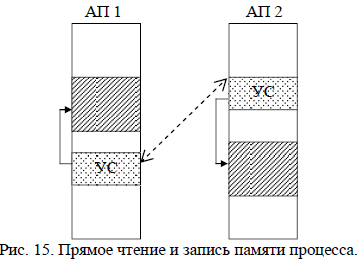
Пример синхронизации доступа к разделяемой памяти — использование функции InterlockedExchange или других объектов синхронизации (рис. 14)

#### Прямое чтение-запись памяти процесса

* Позволяет одному процессу читать или записывать данные в память другого.
* **Организация**:
  + Функции: ReadProcessMemory (чтение) и WriteProcessMemory (запись).
  + Процессы должны обладать необходимыми полномочиями, например, запускаться от одного пользователя.
* Этот метод отличается высокой безопасностью передачи данных и эффективностью, однако накладывает на программиста задачу структурирования информации.

Пример алгоритма работы:

1. Обмен адресами управляющих структур между процессами.
2. Чтение и запись данных на основе этих адресов (рис. 15)

### Ответ на вопрос 16: Опосредованная передача данных между процессами: почтовые ящики, файлы, конвейеры (страницы 47-49)

#### Почтовые ящики

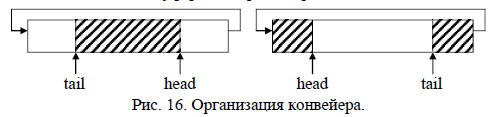
* Механизм асинхронного обмена сообщениями между процессами.
* Реализация включает очереди сообщений и служебные данные.
* Почтовые ящики можно разделить на классы с точки зрения доступа к почтовому ящику:
  1. К процессу-отправителю.
  2. К процессу-получателю.
  3. К паре процессов.
  4. К имени.
* Достоинства метода:
  1. Процессу не нужно знать о существовании других процессов, пока он не получит от них сообщение;
  2. Два процесса могут за один раз обменяться более чем одним сообщением;
  3. Операционная система может гарантировать приватность сообщений. Высокий уровень безопасности достигается особенно при использовании привязки почтового ящика к паре взаимодействующих процессов.
  4. Очередь буферов позволяет отправителю продолжать работу, не ожидая получателя (возможность асинхронной передачи данных).
* Недостатки:
  1. Возможность переполнения буферов сообщений.
* Операции:
  1. **SEND\_MESSAGE**: Отправка сообщения получателю
  2. **WAIT\_MESSAGE**: Получение ответа получателем.
  3. **SEND\_ANSWER**: отправка уведомления о доставке от получателя отправителю.
  4. **WAIT\_ANSWER**: получение уведомления о доставке со стороны отправителя.

#### Файлы

* Используются для передачи больших объемов данных.
* Информация может передаваться:
  + Через содержимое файла.
  + Через атрибуты файла.
  + Через факт доступности файла на запись.
* Достоинства:
  + Объем передаваемой информации ограничен только размером носителя.
  + Надежность передачи данных.
* Недостатки:
  + Низкая скорость.
  + Необходимость очистки данных (обнуление размера файла)

#### Конвейеры

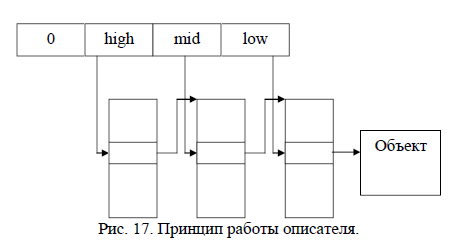
* Решают задачу "производитель-потребитель".
* Организация:
  + В UNIX создаётся функцией pipe, работающей через кольцевой буфер.
  + Процессы получают два дескриптора: для записи и для чтения.
* Особенности:
  + Буфер позволяет избежать необходимости очистки данных.
  + Используются стандартные функции чтения (read) и записи (write).
* Пример: однонаправленный конвейер с использованием указателей головы и хвоста буфера (рис. 16)



### Ответ на вопрос 17: Организация подсистемы безопасности в ОС Windows: понятие описателя, атрибуты безопасности, структура описателя безопасности (SID, ACE, DACL, SACL) (страницы 50-51)

#### Понятие описателя

* Описатель (HANDLE) представляет собой абстрактное число, используемое процессами для доступа к объектам системы.
* С точки зрения операционной системы, описатель является индексом, который хранит:
  1. Указатель на объект.
  2. Права доступа к объекту.
* Примером может служить ситуация, когда процесс пытается записать данные в файл, открытый только для чтения. Если права на запись отсутствуют, менеджер безопасности отклоняет операцию



#### Атрибуты безопасности

Атрибуты безопасности описывают доступность объектов для процессов:

* Определяются через структуру SECURITY\_ATTRIBUTES, которая включает:
  1. Размер структуры.
  2. Указатель на описатель безопасности.
  3. Флаг наследуемости описателя.
* Используются в функциях создания объектов, таких как CreateFile, CreatePipe, CreateProcess【6, стр. 51】.

#### Структура описателя безопасности

* **SID (Security Identifier)**: уникальный идентификатор, обозначающий объект (пользователя, группу или процесс).
* **ACE (Access Control Entry)**: записи, определяющие права доступа. Могут быть разрешающими или запрещающими.
* **DACL (Discretionary Access Control List)**: список, содержащий правила доступа (ACE) для пользователей и групп.
* **SACL (System Access Control List)**: используется для настройки аудита операций над объектом, таких как чтение или запись

### Ответ на вопрос 18: Организация подсистемы безопасности в ОС Windows: права доступа унифицированные и точно определённые, маркеры, олицетворение (страницы 54-55)

#### Унифицированные и точно определённые права доступа

* **Точно определённые права**:
  + Они задаются для каждого типа объектов отдельно, например:
    - Для процессов: PROCESS\_CREATE\_THREAD, PROCESS\_VM\_READ, PROCESS\_VM\_WRITE.
    - Для файлов: FILE\_READ\_DATA, FILE\_WRITE\_DATA.
  + Точно определённые права обеспечивают гибкость управления доступом, но требуют явного указания для каждого объекта.
* **Унифицированные права**:
  + Объединяют несколько точно определённых прав в один набор:
    - GENERIC\_READ: включает чтение и выполнение.
    - GENERIC\_WRITE: запись данных.
    - GENERIC\_EXECUTE: выполнение.
    - GENERIC\_ALL: полный доступ.
  + Для трансформации используются структуры GENERIC\_MAPPING, которые описывают соответствие унифицированных прав точно определённым

#### Маркеры безопасности

* Маркер представляет собой структуру, описывающую права доступа пользователя:
  + Имя пользователя.
  + Группы, в которые он входит.
  + Привилегии.
* Создаётся при входе пользователя в систему и ассоциируется с каждым запущенным процессом.
* Для работы с маркерами используются функции:
  + OpenProcessToken — для получения маркера процесса.
  + GetTokenInformation — для получения информации о маркере

#### Олицетворение

* **Олицетворение** (Impersonation) позволяет выполнять действия от имени другого пользователя.
* Уровни олицетворения:
  + SecurityAnonymous — минимальный доступ.
  + SecurityIdentification — доступ к информации о пользователе, но без права выполнять действия.
  + SecurityImpersonation — вложенное олицетворение на локальной машине.
  + SecurityDelegation — вложенное олицетворение на удалённом сервере.
* Используемые функции:
  + ImpersonateLoggedOnUser — для использования прав другого пользователя.
  + RevertToSelf — для возврата к исходным правам процесса.

### Ответ на вопрос 19: Способы управления памятью: простое непрерывное распределение, оверлейное распределение, распределение статическими разделами (страницы 88-90)

#### Простое непрерывное распределение

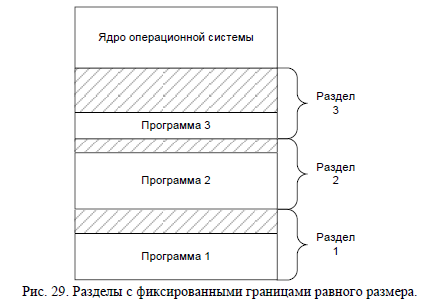
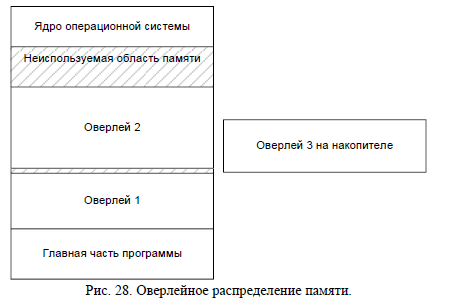
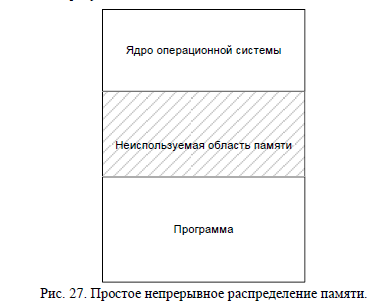
* Вся физическая память делится на три области:
  1. Ядро операционной системы.
  2. Область программы.
  3. Свободная память.
* Достоинства:
  1. Простота реализации.
  2. Низкая стоимость.
* Недостатки:
  1. Однозадачный режим работы.
  2. Низкая эффективность использования памяти, так как не вся программа её полностью использует.
* Пример: ранние версии MS-DOS (рис. 27 на стр. 88).

#### Оверлейное распределение

* Программа разбивается на части (оверлеи), которые перекрываются в адресном пространстве.
* Одновременно в памяти находится основная часть программы и один или несколько оверлеев.
* Достоинства:
  + Программа может быть больше объёма оперативной памяти.
* Недостатки:
  + Затраты на управление оверлеями.
* Пример: Borland C 3.1 для MS-DOS (рис. 28 на стр. 89).

#### Распределение статическими разделами

* Память делится на разделы фиксированного размера, в каждый из которых загружается одна программа.
* Особенности:
  + Разделы могут быть одинакового или разного размера.
  + При неиспользовании памяти в одном из разделов она теряется.
* Достоинства:
  + Мультипрограммный режим работы.
* Недостатки:
  + Фрагментация памяти.
  + Отсутствие защиты между программами и ядром ОС.
* Пример: первые мультипрограммные ОС (рис. 29 на стр. 90).



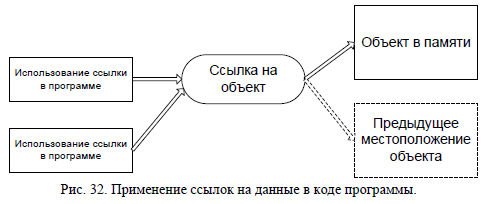
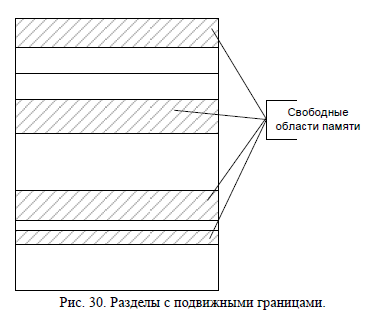
### Ответ на вопрос 20: Способы управления памятью: распределение динамическими разделами, методы устранения фрагментации памяти (страницы 91-94)

#### Распределение динамическими разделами

* **Принцип работы**: память выделяется процессам в виде разделов с подвижными границами. Объем каждого раздела соответствует потребностям программы.
* **Особенности**:
  + Память выделяется побайтно или блоками, кратными степени двойки (например, 4 КБ).
  + Свободные области памяти фиксируются в таблицах, которые используются для поиска подходящих по размеру участков.
* **Методы выделения памяти**:
  + Первый подходящий участок (быстрая обработка, но сохраняется фрагментация).
  + Оптимально подходящий участок (более длительная обработка, снижает фрагментацию).
  + Самый большой участок (уменьшает вероятность фрагментации, но менее эффективно при больших объёмах)

#### Методы устранения фрагментации памяти

1. **Таблицы градаций размеров**:
   * Разделение памяти на группы с фиксированными размерами (например, 4, 16, 64 КБ).
   * Свободные области помещаются в соответствующую таблицу.
   * Преимущества: минимальная фрагментация, поддержка упорядоченного управления памятью.
   * Пример представлен на рисунке 31
2. **Использование ссылок**:
   * Ссылки (указатели) используются для работы с объектами в памяти.
   * Перемещение объекта в памяти не нарушает его доступности, так как ссылки обновляются автоматически.
   * Метод используется в современных исполняющих системах, таких как JVM и .NET【6, стр. 94】.



**Ответ на вопрос 21: Способы управления памятью: сегментная организация памяти (страницы 95-96)**

**Сегментная организация памяти**

Сегментная организация памяти — это метод управления памятью, при котором адресное пространство процесса разделяется на логические сегменты, каждый из которых соответствует определённой функциональности. Например:

1. **Сегмент кода** — содержит инструкции программы.
2. **Сегмент данных** — хранит переменные.
3. **Сегмент стека** — используется для работы с временными данными, такими как параметры функций или локальные переменные.

**Основные характеристики**

1. **Логическая структура:** Память представляется в виде набора сегментов, причём каждый сегмент имеет свою длину и базовый адрес.
2. **Защита памяти:** Каждый сегмент может быть настроен с определённым уровнем доступа (только чтение, чтение-запись, выполнение).
3. **Гибкость:** Сегменты могут изменять свои размеры динамически (например, сегмент стека растёт вниз по мере необходимости).

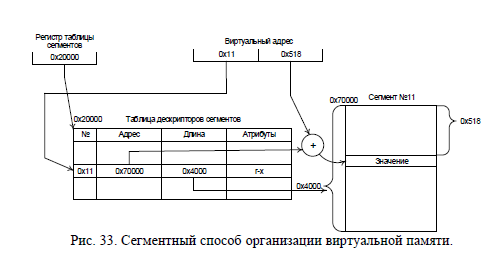
**Преимущества**

* **Простота управления:** Логическое разделение памяти облегчает разработку и отладку программ.
* **Изоляция:** Ошибки в одном сегменте (например, выход за границы массива) не затрагивают другие сегменты.

**Недостатки**

* **Фрагментация:** со временем может возникать ситуация, когда доступная память разбивается на несвязанные небольшие блоки.
* **Сложность реализации:** требуются дополнительные механизмы для преобразования логических адресов в физические.

Для реализации сегментной организации используются таблицы сегментов, которые содержат базовый адрес и размер сегмента. Этот подход активно применялся в процессорах семейства Intel x86, начиная с модели 80286.



**Ответ на вопрос 22: Способы управления памятью: страничная организация памяти (страницы 97-98)**

**Страничная организация памяти**

Страничная организация памяти — это способ управления памятью, при котором адресное пространство процесса делится на равные по размеру блоки, называемые страницами. Физическая память также разбивается на блоки того же размера, называемые страницами памяти (page frames).

**Основные компоненты**

1. **Таблица страниц:** хранит соответствие между виртуальными страницами и страницами памяти.
2. **Страничный регистр:** используется для быстрого доступа к таблице страниц.
3. **Механизм преобразования адресов:** преобразует виртуальный адрес в физический с помощью таблицы страниц.

**Основные принципы работы**

1. Виртуальное адресное пространство процесса делится на страницы фиксированного размера (например, 4 КБ).
2. Таблица страниц обеспечивает преобразование адресов:
   * Если страница загружена в оперативную память, возвращается физический адрес.
   * Если страницы нет в памяти, возникает исключение (page fault), и она загружается из файла подкачки.
3. Системы используют кэш (Translation Lookaside Buffer, TLB) для ускорения преобразования адресов.

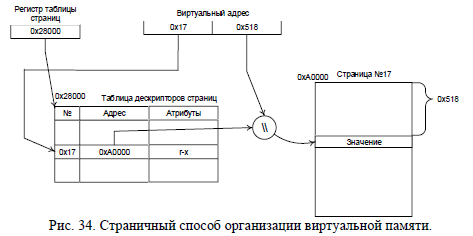
**Преимущества**

* **Исключение фрагментации:** Память выделяется блоками фиксированного размера.
* **Эффективное использование памяти:** Неиспользуемые страницы могут быть выгружены на диск.
* **Изоляция процессов:** Каждый процесс имеет своё виртуальное адресное пространство.

**Недостатки**

* **Затраты на управление:** Поддержание таблицы страниц требует дополнительной памяти.
* **Замедление:** Преобразование адресов может замедлять доступ к памяти.

Этот метод широко используется в современных операционных системах, включая Windows и Linux. Для повышения производительности в современных процессорах применяется многоуровневая организация таблиц страниц.



**Ответ на вопрос 23: Способы управления памятью: сегментно-страничная организация памяти (страницы 99-100)**

**Сегментно-страничная организация памяти**

Сегментно-страничная организация памяти сочетает преимущества сегментной и страничной систем управления памятью. В этой модели адресное пространство процесса делится на сегменты, а каждый сегмент, в свою очередь, разбивается на страницы фиксированного размера.

**Основные принципы работы**

1. Виртуальный адрес делится на три компонента:
   * **Номер сегмента** — указывает на конкретный сегмент.
   * **Номер страницы** — определяет страницу внутри сегмента.
   * **Смещение внутри страницы** — определяет точное местоположение данных в странице.
2. Для преобразования адресов используются:
   * **Таблица сегментов:** хранит базовые адреса и размеры сегментов.
   * **Таблица страниц:** хранит отображение страниц каждого сегмента на физическую память.

**Преимущества**

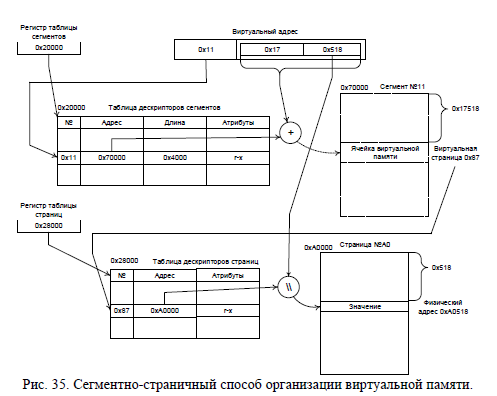
* **Гибкость:** Логическое разделение на сегменты позволяет изолировать функциональные части программы (код, данные, стек).
* **Устранение фрагментации:** благодаря страничному подходу исключается проблема внешней фрагментации.
* **Управление доступом:** Каждому сегменту можно задать индивидуальные права доступа.

**Недостатки**

* **Сложность реализации:** требуются дополнительные таблицы и механизмы для преобразования адресов.
* **Затраты памяти:** Хранение таблиц сегментов и страниц увеличивает требования к оперативной памяти.

**Пример**

Сегментно-страничная организация активно использовалась в архитектуре процессоров Intel x86 (начиная с модели 80386). Она позволяла объединить удобство сегментной модели с эффективностью страничного управления памятью.



**Ответ на вопрос 24: Организация физической памяти компьютера архитектуры Intel. Спецификации ACPI, UEFI (страницы 101-104)**

**Организация физической памяти в архитектуре Intel**

Физическая память компьютеров на архитектуре Intel организована таким образом, чтобы обеспечить:

1. Эффективное использование оперативной памяти.
2. Возможность динамического управления памятью.
3. Поддержку работы с большими объёмами данных.

**Основные элементы физической памяти:**

1. **Первый мегабайт памяти:**
   * Относится к историческому наследию первых ПК.
   * Адресуется напрямую в реальном режиме процессора.
   * Используется для BIOS, таблиц прерываний (IVT) и программ начальной загрузки.
2. **Память за пределами первого мегабайта:**
   * Доступна в защищённом режиме процессора.
   * Может использоваться как физическая или виртуальная память.

**Таблица физической памяти**

Операционные системы используют таблицы для управления физической памятью:

* **RAM:** Основная оперативная память.
* **ROM:** Постоянное запоминающее устройство, содержащее микропрограммы (BIOS или UEFI).
* **Зарезервированная память:** используется для системных нужд или периферийных устройств.

**Спецификация ACPI**

**ACPI** (Advanced Configuration and Power Interface) — стандарт, который обеспечивает:

1. Управление энергопотреблением компонентов компьютера.
2. Информацию о топологии аппаратных устройств.
3. Интерфейс для программной настройки оборудования.

**Основные функции ACPI:**

* Управление питанием процессора, жёстких дисков, видеокарт и других компонентов.
* Поддержка "спящих" режимов (S1, S2, S3, S4, S5).
* Динамическое конфигурирование оборудования в зависимости от нагрузки.

**Спецификация UEFI**

**UEFI** (Unified Extensible Firmware Interface) — современная замена BIOS. Основные преимущества:

1. **Графический интерфейс:** Поддержка экранов высокого разрешения и курсора мыши.
2. **Поддержка GPT:** обеспечивает работу с дисками более 2 ТБ.
3. **Быстрая загрузка:** снижает время инициализации оборудования.
4. **Безопасность:** поддерживает технологию Secure Boot для предотвращения загрузки неподписанного ПО.

**Отличия UEFI от BIOS:**

* Возможность работы с файловыми системами.
* Расширенные опции конфигурации оборудования.
* Поддержка 64-битных режимов работы.

**Ответ на вопрос 25: Распределение виртуального адресного пространства в ОС Windows, управление распределением памяти с помощью функций группы Virtual, с использованием куч (страницы 107-109)**

**Распределение виртуального адресного пространства в ОС Windows**

Вся виртуальная память в ОС Windows делится на три класса (Каждая страница может принадлежать в один момент времени к одному классу):

* **Доступная память** – это вся память процесса (4 Гб), то есть та память, которую он может адресовать.
* **Зарезервированная память** – это область адресного пространства, затребованная процессом, но не используемая для хранения данных. При резервировании памяти вы не выделяете память, а выделяете участок адресного пространства как единицу, с которой вы намерены в дальнейшем работать. Физическая память при этом не выделяется. Эта память может быть затребована в любой момент времени.
* **Выделенная память** – это память, находящаяся в работе. Она может быть на диске или в ОЗУ, но она хранит в себе какие-либо данные.

**Управление памятью с помощью функций группы Virtual**

Windows предоставляет набор функций для управления виртуальной памятью:

1. **VirtualAlloc:** выделяет блок виртуальной памяти.
   * Позволяет задавать размер и свойства памяти (чтение, запись, исполнение).
2. **VirtualFree:** освобождает ранее выделенный блок виртуальной памяти.
3. **VirtualProtect:** изменяет атрибуты доступа к памяти (например, делает её только для чтения).
4. **VirtualQuery:** получает информацию о состоянии памяти (занята, свободна, защищена).

**Использование куч (Heap)**

Куча — это специальная область памяти, предназначенная для динамического управления памятью в приложениях.

**Основные функции:**

1. **HeapCreate:** создаёт новую кучу с заданными параметрами.
2. **HeapAlloc:** выделяет блок памяти из кучи.
3. **HeapFree:** освобождает блок памяти, ранее выделенный с помощью HeapAlloc.
4. **HeapDestroy:** уничтожает кучу, освобождая все связанные с ней ресурсы.

**Преимущества куч:**

* Эффективное управление памятью, так как куча работает в рамках виртуального адресного пространства процесса.
* Возможность создавать несколько куч с разными параметрами для различных нужд программы.

**Ответ на вопрос 26: Подсистема ввода-вывода в современных ОС. Основные понятия (страницы 74-76)**

**Подсистема ввода-вывода**

Подсистема ввода-вывода — это компонент операционной системы, обеспечивающий взаимодействие между приложениями и внешними устройствами (жёсткими дисками, принтерами, сетевыми адаптерами и т. д.). Она выполняет функции управления потоками данных и абстрагирования оборудования от приложений.

**Основные компоненты подсистемы ввода-вывода**

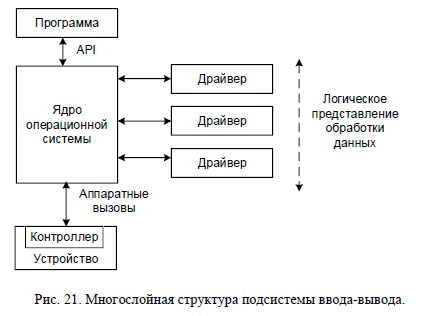
1. **Драйверы устройств:** Программные модули, обеспечивающие прямое управление оборудованием.
2. **Контроллер**: специализированный блок управления, которым снабжено каждое устройство, взаимодействует с драйвером
3. **Задание ввода**-**вывода**

**Режимы работы подсистемы ввода-вывода**

1. **Синхронный режим:** Приложение ожидает завершения операции ввода-вывода перед выполнением следующих инструкций.
2. **Асинхронный режим:** Операции ввода-вывода выполняются в фоновом режиме, что позволяет приложениям продолжать работу.

**Ответ на вопрос 27: Многослойная модель подсистемы ввода-вывода. Организация ввода-вывода в ОС Windows (страницы 76-77)**

Многослойная модель подсистемы ввода-вывода используется в современных операционных системах и предоставляет драйверам возможность многократной последовательной обработки одного задания ввода-вывода в виде, так называемых, слоёв. При использовании многослойной модели подсистемы ввода-вывода дополнительно повышается гибкость и расширяемость функций по управлению устройством. Так драйвер файловой системы может обрабатывать запросы для различных видов накопителей, например, подключенных по интерфейсам SATA и USB.



**В Windows:**

Драйвер вызывает спец. Функцию регистрации в цепочке ввода-вывода. Все запросы идут последовательно по цепочке драйверов посредством ОС. Таким образом один запрос обрабатывается несколькими слоями драйверов. Каждый драйвер получает 3 запроса: запрос к устройству -> работа устройства -> завершение работы устройства

### Ответ на вопрос 28: Система управления файлами. Структура памяти накопителей. Понятия MBR, GPT, LBA (страницы 44-45, 78-79)

#### Система управления файлами

* **Основная задача**: обеспечивает взаимодействие программ с файлами и каталогами.
* Включает возможности:
  + Создание, удаление, переименование файлов.
  + Обеспечение защиты от несанкционированного доступа.
  + Взаимодействие с файловыми системами и подсистемой ввода-вывода.

#### Структура памяти накопителей

* **Логический том**:
  + Состоит из секторов (обычно 512 байт), которые объединяются в кластеры для работы с большими объемами данных.
  + Типичная структура:
    - Сектор загрузчика (зарезервированная область).
    - Таблицы размещения файлов (FAT, NTFS, Ext).
    - Данные【6, стр. 78】.

#### Понятия

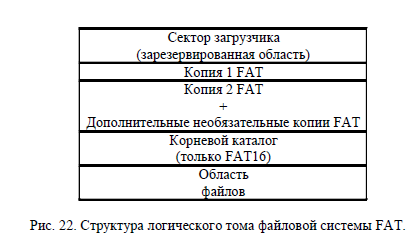
1. **MBR (Master Boot Record)**:
   * Расположен в первом секторе диска.
   * Содержит загрузчик и таблицу разделов.
   * Ограничение: поддержка дисков до 2 ТБ и до 4 разделов.
2. **GPT (GUID Partition Table)**:
   * Современный стандарт, заменяющий MBR.
   * Поддерживает диски объемом до 9.4 зеттабайт.
   * Позволяет создавать до 128 разделов.
   * Использует уникальные идентификаторы для разделов.
3. **LBA (Logical Block Addressing)**:
   * Метод адресации секторов диска через линейный порядок, заменяя CHS (Cylinder-Head-Sector).
   * Упрощает работу с большими объемами памяти【6, стр. 44-45】.

### Ответ на вопрос 29: Файловая система FAT (страницы 78-80)

#### Основные характеристики FAT

Файловая система FAT (File Allocation Table) включает:

1. **Области на диске**:
   * Зарезервированная область (сектор загрузчика).
   * Область таблицы размещения файлов (FAT).
   * Корневой каталог.
   * Область файлов.



1. **Структура FAT**:
   * Содержит записи, каждая из которых соответствует кластеру.
   * Форматы записи:
     + 12-битный FAT для гибких дисков.
     + 16-битный FAT16 для малых жестких дисков.
     + 32-битный FAT32 для больших дисков.
2. **Кластеры**:
   * Блоки секторов, кратные степени двойки.
   * Размер определяется делением объема диска на 64 Кбайт (например, 32 Кбайт для 1.2 ГБ диска).

#### Достоинства и недостатки

* **Достоинства**:
  + Простота реализации.
  + Совместимость с разными ОС.
* **Недостатки**:
  + Низкая производительность на больших дисках.
  + Ограничение на размер файла (до 4 ГБ в FAT32).
  + Отсутствие встроенной защиты данных

### Ответ на вопрос 30: Файловая система NTFS (страницы 81-83)

#### Основные характеристики NTFS

* NTFS (New Technology File System) делит весь диск на кластеры — минимальные блоки данных. Размер кластера может варьироваться от 512 байт до 64 КБ, стандартный размер — 4 КБ.
* Каждая запись на диске управляется через **Master File Table (MFT)** — центральный каталог файлов. MFT представляет собой таблицу фиксированных записей, где каждая запись соответствует одному файлу или каталогу.

#### Структура диска NTFS

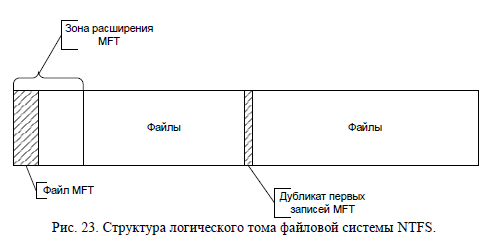
* **MFT-зона**: первые 12% объема диска зарезервированы для роста MFT, чтобы минимизировать её фрагментацию.
* Остальная часть диска используется для хранения файлов. Свободное место может быть перераспределено между MFT-зоной и файловым пространством при необходимости.

#### Основные преимущества NTFS

1. **Журналирование**:
   * NTFS поддерживает транзакционную модель обработки данных, что обеспечивает восстановление после сбоев.
   * Запись изменений в журнал позволяет сохранить целостность данных даже в случае внезапного отключения питания.
2. **Безопасность**:
   * NTFS хранит атрибуты безопасности для каждого файла и каталога.
   * Операционная система проверяет права доступа, сравнивая их с маркером пользователя.
3. **Поддержка Unicode**:
   * Имена файлов могут содержать символы национальных алфавитов (в формате UTF-16), максимальная длина имени — 255 символов.

#### Дополнительные возможности

* Потоки (streams): файлы разделены на потоки, включая данные и атрибуты. Потоки могут быть скрытыми для стандартных приложений.
* Масштабируемость: поддержка больших объёмов данных и файлов, размеры которых могут достигать терабайтов.



### Ответ на вопрос 31: Службы в ОС Windows. Программа управления службой (страницы 50-55)

#### Общие сведения о службах

Службы — это специальные программы операционной системы Windows, которые обеспечивают выполнение системных функций, таких как:

* Удалённый вызов процедур.
* Сетевая печать.
* Терминальный доступ и другие задачи.

Службы работают как фоновые процессы, не взаимодействующие с пользователем напрямую. Управление ими осуществляется через диспетчер управления службами (SCM — Service Control Manager), приложение-службу и программу управления службой【6, стр. 50-51】.

#### Диспетчер управления службами (SCM)

* Представляет собой системный компонент (файл Services.exe), автоматически запускаемый при старте системы.
* Обеспечивает интерфейс для управления службами, включая запуск, остановку, приостановку и возобновление работы служб【6, стр. 51】.

#### Программа управления службой

Программа управления службой — это приложение, которое:

1. Позволяет устанавливать, удалять и настраивать службы.
2. Работает с диспетчером управления службами.
3. Может запускать службы автоматически, вручную или запрещать их запуск.
4. Реализует обработку ошибок, например, в случае сбоя службы можно:
   * Перезагрузить систему.
   * Отправить сообщение в системный журнал.
   * Повторить запуск службы через заданный интервал времени【6, стр. 51-52】.

#### Основные функции программного интерфейса

* OpenSCManager — для открытия диспетчера управления службами.
* CreateService — для создания новой службы.
* StartService — для запуска службы.
* ControlService — для управления состоянием службы (остановка, приостановка и т. д.).
* DeleteService — для удаления службы.
* QueryServiceStatus — для проверки текущего состояния службы【6, стр. 54-55】.

Если требуется продолжение темы, напишите!