**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**

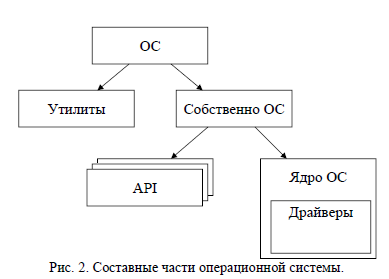
1. Увеличение пропускной способности компьютера (количество задач в единицу времени).
2. Уменьшение времени реакции системы.
3. Контроль работоспособности аппаратных и программных компонентов.
4. Обеспечение пользовательского интерфейса.
5. Управление программами и данными в процессе вычислений.
6. Адаптивность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

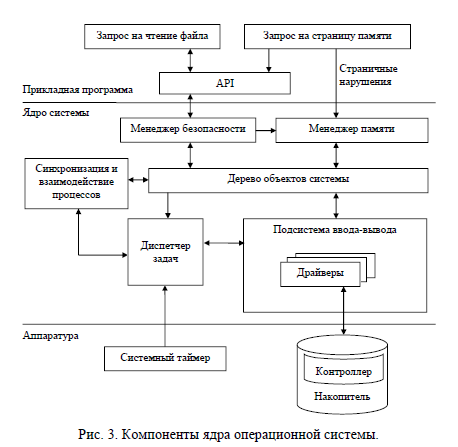
**Структура операционной системы**

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

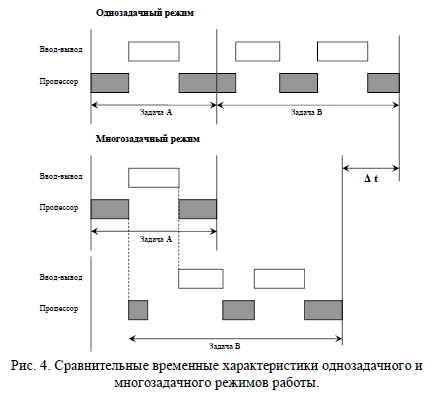
#### Задача (task)

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)

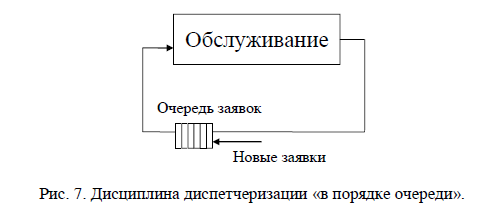
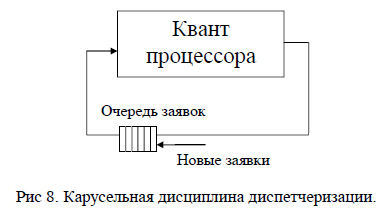
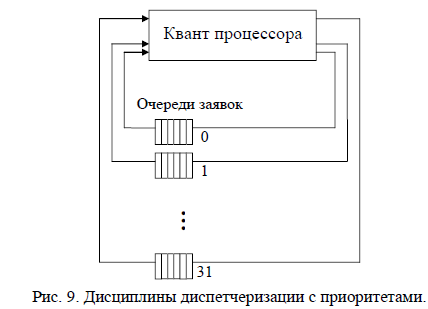
#### Диспетчер задач

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

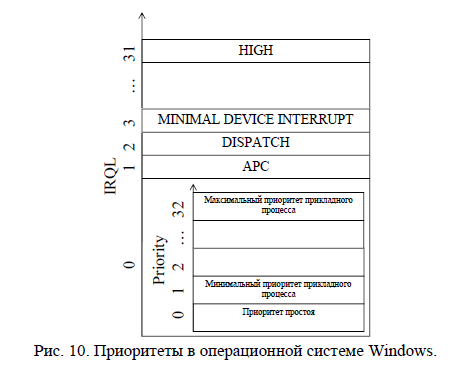
#### Программные приоритеты

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