**Вопросы к экзамену**

1. **Операционные системы: определение, назначение, состав, функции.**

**Операционная система** — это комплект программ, которые служат интерфейсом между модулями вычислительных систем и прикладными программными приложениями, а также управляют компьютерным оборудованием и процессами вычислений, эффективным распределением вычислительных мощностей среди процессов вычислений.

**Операционная система** — это комплекс программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

**Операционная система** (ОС) — комплекс системных и управляющих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы (ВС) (Вычислительная система — взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации) и удобства работы с ней.

**Две основные функции (назначение)** ОС:

1) предоставлять пользователю некую расширенную виртуальную машину, с которой легче работать (легче программировать), чем непосредственно с аппаратурой реального компьютера или реальной сети;

2) управлять ресурсами вычислительной системы. Поэтому в специальной литературе ОС представляется всегда двояко: как расширенная виртуальная машина и как система управления ресурсами.

Функции ОС автономного компьютера обычно группируются в соответствии с типами локальных ресурсов, которыми управляет ОС. Такие группы называют подсистемами. Наиболее важные из них:

• подсистема управления процессами,

• подсистема управления памятью,

• подсистема управления файлами,

• подсистема управления внешними устройствами,

• подсистема пользовательского интерфейса,

• подсистема защиты данных и администрирования

**Функции** ОС:

* + прием от пользователя (оператора) заданий или команд, сформулированных на соответствующих языках, и их обработка;
  + загрузка в ОП программ и их исполнение;
  + инициация программы (передача ей управления);
  + прием и исполнение программных запросов на запуск, приостановку, остановку других программ; организация взаимодействия между задачами;
  + идентификация всех программ и данных;
  + обеспечение работы системы управления файлами и/или систем управления БД;
  + обеспечение режима мультипрограммирования (многозадачности);
  + планирование и диспетчеризация задач;
  + обеспечение функций по организации и управлению операциями ввода/вывода;
  + удовлетворение жестким ограничениям на время ответа в режиме реального времени (для соответствующих ОС);
  + управление памятью, организация виртуальной памяти;
  + организация механизмов обмена сообщениями и данными между выполняющимися программами;
  + защита одной программы от влияния другой; обеспечение сохранности данных;
  + аутентификация, авторизация и другие средства обеспечения безопасности;
  + предоставление услуг на случай частичного сбоя системы;
  + обеспечение работы систем программирования;
  + параллельное исполнение нескольких задач.

1. **Операционные системы: классификация, основные этапы развития, особенности современного этапа развития**

Существует несколько классификаций ОС.

В зависимости от **способа организации вычислений**:

* + Системы пакетной обработки – основной задачей является организация наибольшего количества вычислительных процессов за единицу времени. Определенные процессы объединяются в пакет, который затем обрабатывает ОС.
  + Системы разделения времени – создание возможности единовременного взаимодействия с устройством сразу несколькими людьми. В порядке очереди каждый пользователь получает определенный промежуток процессорного времени.
  + Системы реального времени – организация работы каждой задачи за определенный промежуток времени, присущий каждой конкретной задаче.

В зависимости от **типа ядра**:

* + OС с монолитным ядром (ядро ОС выполняет все функции, включая управление памятью, устройствами ввода-вывода и файловой системой, в одном монолитном модуле);
  + OС с микроядром (ядро ОС содержит только основные функции, такие как планирование задач, управление памятью и межпроцессное взаимодействие, а остальные функции реализуются в виде отдельных модулей, работающих в привилегированном или пользовательском режиме);
  + OС с гибридным ядром (комбинация монолитного и микроядерного подходов, где некоторые компоненты, такие как драйверы устройств, могут работать в пространстве ядра, в то время как другие функции, такие как файловая система и сетевые протоколы, могут работать в виде отдельных модулей, загружаемых по требованию).

В зависимости от **количества единовременно решаемых задач**:

* + однозадачные;
  + многозадачные;

В зависимости от **количества пользователей**:

* + однопользовательские;
  + многопользовательские.

В зависимости от **количества поддерживаемых процессоров**:

* + однопроцессорные
  + многопроцессорные

В зависимости от **возможности работы в компьютерной сети**:

* + локальные – автономные ОС, которые не позволяют работать с компьютерными сетями;
  + сетевые – ОС с поддержкой компьютерных сетей.

В зависимости от **роли в сетевом взаимодействии**:

* + серверные – ОС, открывающие доступ к ресурсам сети и осуществляющие управление сетевой инфраструктурой;
  + клиентские – ОС, которые имеют возможность получения доступа к ресурсам сети.

В зависимости от **типа лицензии**:

* + открытые – ОС с открытым исходным кодом, который можно изучать и редактировать;
  + проприетарные – ОС, связанные с определенным правообладателем и, как правило, имеющие закрытый исходный код.

В зависимости от **сферы использования**:

* + ОС мэйнфреймов – больших компьютеров;
  + ОС серверов;
  + ОС персональных компьютеров;
  + OC мобильных устройств;
  + встроенные OC;
  + OC маршрутизаторов.

**Основные этапы развития:**

## Первое поколение (1945-1955 гг.)

**Первое поколение** компьютеров строилось преимущественно на электронных лампах. Существенная часть времени уходила на подготовку запуска программы, а сами программы выполнялись строго последовательно (такой режим работы называется **последовательной обработкой** данных).

## Второе поколение (1955-1965 гг.)

**Второе поколение** компьютеров характеризуется использованием транзисторов, что повысило их надёжность и продлило время непрерывной работы.

## Третье поколение (1965-1980 гг.)

Компьютеры **третьего поколения** использовали малые интегральные схемы, что дало им преимущество в цене и качестве по сравнению с машинами второго поколения. Самым важным достижением явилась **многозадачность**. В компьютерах предыдущего поколения вычисления останавливались на время ввода-вывода. Проблема была решена разбиением памяти на несколько частей, называемых разделами, в каждом из которых выполнялось отдельное задание. При окончании выполнения каждого текущего задания операционная система могла загружать новое задание с диска в освободившийся раздел памяти и запускать это задание. Этот технический прием называется **подкачкой данных**, или **спулингом** (spooling — английское слово, которое произошло от Simultaneous Peripheral Operation On Line, то есть совместная периферийная операция в интерактивном режиме), и его также используют для выдачи полученных данных.

## Четвертое поколение (1980-2005 гг.)

Разработка БИС (большие интегральные схемы, LSI, Large Scale Integration — кремниевые микросхемы, содержащие тысячи транзисторов на одном квадратном сантиметре) привела к появлению микрокомпьютеров. Операционные системы для микрокомпьютеров принято относить к четвертому поколению.

## Пятое поколение (2005 г. - по н.в.)

Широкую популярность приобрела технология **виртуализации** — представление вычислительных ресурсов, абстрагированное от аппаратной реализации.

**Особенности современного этапа и перспективы развития ОС**

На современном этапе развития операционных систем на передний план вышли средства обеспечения безопасности. Современным операционным системам присуща многоплатформенность, т.е. способность работать на совершенно различных типах компьютеров.

1. **Компоненты архитектуры вычислительных систем, их назначение и взаимодействие.**

АЛУ – выполняет арифметические и логические операции; УУ – организует процесс выполнения программ

Процессор: Среди специальных регистров процессора стоит выделить **счетчик команд**, который содержит адрес ячейки памяти со следующей выбираемой командой, указатель стека, который ссылается на вершину текущего стека в памяти и **слово состояния программы** — PSW (Program Status Word). Каждый раз при переключении задач содержимое регистров предыдущей задачи выгружается в оперативную память и загружается содержимое регистров для следующей задачи. Этот процесс называется переключением контекста. Многие современные процессоры способны одновременно выполнять более одной команды. Подобная организация работы называется **конвейером**. Процессоры с **суперскалярной** архитектурой имеют несколько исполнительных блоков, которые могут выполнять команды не в порядке их следования, что хоть и может повысить быстродействие, но усложняет операционную систему.

Идея RISC заключается в замене сложных инструкций на комбинацию простых. Конвейер инструкций Arm-Risc:

* **(Fetch) Извлечение** инструкции из памяти и увеличение счетчика команд, чтобы извлечь следующую инструкцию в следующем такте.
* **(Decode) Декодирование** инструкции — определение, что эта инструкция делает. То есть активация необходимых для выполнения этой инструкции частей микропроцессора.
* **(Execute) Выполнение** включает использование арифметико-логического устройства (АЛУ) или совершение сдвиговых операций.
* **(Memory) Доступ к памяти**, если необходимо. Это то, что делает инструкция load.
* **(Write Back) Запись результатов** в соответствующий регистр.

Второй основной составляющей любого компьютера является память. В идеале память должна быть максимально быстрой (работать быстрее, чем производится выполнение одной инструкции, чтобы работа центрального процессора не замедлялась обращениями к памяти), довольно большой и чрезвычайно дешевой.

Магнитный жесткий диск состоит из одной или нескольких металлических пластин, вращающихся со скоростью 5400, 7200, 10 000 или 15 000 оборотов в минуту.

Еще можно найти компьютеры, использующие шину PCI (Peripheral Component Interconnect), разработанную компанией Intel. Таким образом, контроллер соединяется с памятью непосредственно, то есть передача данных между центральным процессором и памятью происходит не через шину PCI.

## Шина USB

## Видеокарта – устройство, преобразующее цифровую информацию в форму, пригодную для дальнейшего вывода на экран монитора.

**4. Задачи операционной системы по управлению и организации работы компьютера.**

Выделим три важные группы функций.

1. **Виртуальная память**. Механизм виртуальной памяти используется многими операционными системами. Она позволяет создать впечатление, будто у машины больше памяти, чем есть на самом деле.
2. **Файловый ввод-вывод**.
3. **Параллелизм** (как организовано одновременное выполнение нескольких процессов, обмен информацией и синхронизация). Под процессом можно понимать работающую программу и всю информацию об ее состоянии (памяти, регистрах, счетчике команд, вводе-выводе и т. д.).

Самыми важными задачами управления компьютерным оборудованием, осуществляемыми операционной системой, являются:

* + Параллельное функционирование модулей ввода, вывода информации и процессора.
  + Организация кэширования данных и выполнение согласования скоростей информационного обмена.
  + Разбиение модулей и информационных данных среди процессов.
  + Организация удобной работы логического интерфейса между модулями и оставшейся частью системы.
  + Организация поддержки различных устройств с обеспечением возможности просто их добавить.
  + Режим динамической загрузки и выгрузки драйверов.
  + Обеспечение поддержки набора файловых систем.

Модули, предназначенные для ввода и вывода информации, подразделяются на следующие типы:

* + Модули, ориентированные на работу с блоками.
  + Модули, ориентированные на работу с байтами.

Также следует упомянуть технологию, которая позволяет создавать несколько сред или выделенных ресурсов из единой физической аппаратной системы – называется **виртуализация**. Программное обеспечение, гипервизор, напрямую подключается к этой аппаратной системе и позволяет разбить ее на отдельные, безопасные среды – виртуальные машины. По идее, гипервизор должен распределять аппаратные ресурсы между виртуальными машинами так, чтобы процессы выполнялись быстрее. Физическая машина с гипервизором называется **хостом**, а виртуальные машины, которые используют ресурсы данного хоста – **гостями**. Для них ресурсами являются процессор, память, хранилище. Для получения доступа к этим ресурсам операторы управляют виртуальными экземплярами.

**5. Структура ядра и его функции. Объекты ядра. Основные операции над объектами ядра.**

Под архитектурой ОС обычно понимают структурную организацию ОС на основе программных модулей. Современные ОС представляют собой хорошо структурированные модульные системы.

Единой архитектуры ОС не существует, но существуют универсальные подходы к структурированию ОС:



Наиболее общим подходом к структуризации ОС является подразделение модулей на две группы:

* + модули, выполняющие основные функции ОС — **ядро ОС**;
  + модули, выполняющие вспомогательные функции ОС. Модули ядра выполняют базовые функции ОС
  + управление процессами;
  + управление памятью;
  + управление устройствами ввода-вывода.

Функции, входящие в состав ядра, можно разделить на два класса.

1. **класс**. Функции для решения внутрисистемных задач организации вычислительного процесса (переключение контекстов процессов, загрузка/выгрузка страниц, обработка прерываний). Эти функции недоступны для приложений.
2. **класс**. Функции для поддержки приложений (доступны приложениям). Эти функции создают для приложений так называемую прикладную программную среду и образуют интерфейс прикладного программирования — API. Приложения обращаются к ядру с запросами — системными вызовами. Функции API обслуживают системные вызовы — предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения.

**Объекты ядра** используются системой и приложениями для управления множеством разных ресурсов: процессами, потоками, файлами и т.д. **Объект** — это коллекция данных, являющихся частью режима ядра операционной системы, которыми управляет Диспетчер объектов Windows (Windows Object Manager).

Типичные объекты режима ядра включают следующие категории объектов:

* объекты устройств;
* объекты файлов;
* символические ссылки;
* разделы реестра;
* потоки и процессы;
* объекты диспетчера ядра, такие как объекты событий и объекты мьютексов;
* объекты обратного вызова;
* объекты section.

Например, когда мы создаем объект ядра в операционной системе Windows, функция возвращает **описатель**, идентифицирующий созданный объект (HANDLE).

Создание новых объектов, или открытие по имени уже существующих, приложение может осуществить при помощи Win32-функций, таких, как **CreateFile**, **CreateSemaphore**, **OpenSemaphore** и т.д. Полученный описатель затем нужно использовать как первый параметр в различных функциях, например, **ReadFile** и **WriteFile**, для работы с реальным файлом. Во избежание утечки памяти всегда рекомендуется закрывать объект, когда в нем отпала надобность. Впрочем, по окончании работы процесса система закрывает все его объекты. Одной из таких функций является функция **CloseHandle**. Некоторые дескрипторы требуют своей функции закрытия, например, функция **FindClose** должна применяться для закрытия дескриптора, полученного от функции **FindFirstFile**.

**6. Утилиты. Системные обрабатывающие программы. Библиотеки процедур. Программы дополнительных услуг.**

**Утилиты** — это системное программное обеспечение, которое помогает поддерживать правильное и бесперебойное функционирование компьютерной системы. Они помогают операционной системе управлять, организовывать, поддерживать и оптимизировать функционирование компьютерной системы. Утилиты предоставляют доступ к возможностям (параметрам, настройкам, установкам), недоступным без их применения, либо делают процесс изменения некоторых параметров проще (автоматизируют его).

Основные функции обрабатывающих программ:

1. **перенос информации**. Перенос может выполняться между различными устройствами или в пределах одного устройства. При этом под устройствами понимаются: ОП, устройства ВП, устройства ввода-вывода;
2. **преобразование информации**. То есть после считывания информации с устройства обрабатывающая программа преобразует эту информацию, а только затем записывает ее на это же или на другое устройство.

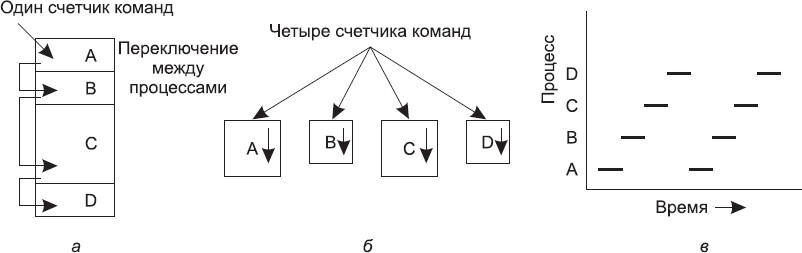
В зависимости от того, какая из этих двух функций является основной, обрабатывающие системные программы делятся на **утилиты** и **лингвистические процессоры**. Основной функцией утилиты является перенос информации, а основная функция лингвистического процессора – перевод описания алгоритма с одного языка на другой. Лингвистические процессоры делятся на **трансляторы** и **интерпретаторы**.

В результате работы **транслятора** алгоритм, записанный на языке программирования (исходная виртуальная программа), преобразуется в алгоритм, записанный на машинном языке. **Трансляторы** делятся на компиляторы и ассемблеры. **Интерпретатор** в отличие от транслятора не выдаёт машинную программу целиком. Выполнив перевод очередного оператора исходной программы в соответствующую совокупность машинных команд, интерпретатор обеспечивает их выполнение. Затем преобразуется тот исходный оператор, который должен выполняться следующим по логике алгоритма, и так далее.

Библиотеки процедур и функций различного назначения включены в категорию вспомогательных модулей операционной системы. **Библиотеки** — это набор функций, которые могут использоваться в различных программах. Библиотеки могут быть статические (библиотека привязывается к определенной программе или софт содержит данную библиотеку в своем теле.) и динамическими (библиотеки грузятся в оперативную память и используются).

**Программы доп услуг:** В эту категорию входит большое количество разнообразных программ: специальный вариант пользовательского интерфейса, калькулятор, некоторые игры (какие, например, поставляются в составе ОС).

**7. Понятие процесса. Системные и пользовательские процессы. Операции над процессами.**

**Процесс** — это просто экземпляр выполняемой программы, включая текущие значения счетчика команд, регистров и переменных. Концептуально у каждого процесса есть свой, виртуальный, центральный процессор. Постоянное переключение между процессами называется **мультипрограммированием**, или **многозадачным** режимом работы.

Компьютер: а — четыре программы, работающие в многозадачном режиме; б — концептуальная модель четырех независимых друг от друга

последовательных процессов; в — в отдельно взятый момент активна только одна программа

Процессы, которые выполняют системный код, называются **системными** и применяются к системе в целом. Они занимаются выполнением таких служебных задач, как распределение памяти, обмен страницами между внутренним и вспомогательным запоминающими устройствами, контроль устройств и т.п.

**Пользовательские** процессы выполняют собственный код и иногда обращаются к системным функциям. Выполняя собственный код, пользовательский процесс пребывает в пользовательском режиме (user mode).

**Операции над процессами.**

**Запуск процесса**. Из числа процессов, находящихся в состоянии готовность, операционная система выбирает один процесс для последующего исполнения.

**Приостановка процесса**. Работа процесса, находящегося в состоянии исполнение, приостанавливается в результате какого-либо прерывания.

**Блокирование процесса**. Процесс блокируется, когда он не может продолжать работу, не дождавшись возникновения какого-либо события в вычислительной системе.

**Разблокирование процесса**. После возникновения в системе какого-либо события операционной системе нужно точно определить, какое именно событие произошло

**Переключение контекста**

В действительности же деятельность мультипрограммной операционной системы состоит из цепочек операций, выполняемых над различными процессами, и сопровождается переключением процессора с одного процесса на другой.

**8. Организация межпроцессного взаимодействия в ОС. Сигналы. Каналы. Классические проблемы межпроцессного взаимодействия.**

При выполнении параллельных процессов может возникать проблема, когда каждый процесс, обращающийся к разделяемым данным, исключает для всех других процессов возможность одновременного с ним обращения к этим данным – это называется **взаимоисключением.** Ресурс, который допускает обслуживание только одного пользователя за один раз, называется критическим ресурсом. Для организации коммуникации между одновременно работающими процессами применяются средства межпроцессного взаимодействия (Interprocess Communication - IPC).

**Средства локального уровня** IPC привязаны к процессору и возможны только в пределах компьютера.

**Удаленные IPC** предоставляют механизмы, которые обеспечивают взаимодействие как в пределах одного процессора, так и между программами на различных процессорах, соединенных через сеть.

Под **высокоуровневыми IPC** обычно подразумеваются пакеты программного обеспечения, которые реализуют промежуточный слой между системной платформой и приложением.

**Сигнал** в операционных системах семейства Unix — асинхронное уведомление процесса о каком-либо событии, один из основных способов взаимодействия между процессами. Отдельные сигналы подразделяются на три класса:

* системные сигналы (ошибка аппаратуры, системная ошибка и т.д.);
* сигналы от устройств;
* сигналы, определенные пользователем.

**Канал** (pipe) представляет собой средство связи стандартного вывода одного процесса со стандартным вводом другого. После создания канала, процесс может при помощи обычного системного вызова **write**() выводить данные в него, а затем вводить их, вызывая соответственно функцию **read**(). При выполнении вызова **fork**() дескрипторы канала наследуются процессом-"потомком". Таким образом, оба процесса получают возможность обмениваться данными.

**Проблемы:**

**Синхронный доступ**. Если все процессы считывают данные из файла, то в большинстве случае проблем не возникает. Однако, при попытке одним из процессов изменить этот файл, могут возникнуть так называемые конкурентные ситуации (race conditions).

**Дисциплина доступа**. Если нужно, чтобы как можно большее количество пользователей могли записывать данные, организуется так называемая очередь (по правилу «один пишет, несколько читают»). Практически организуется две очереди: одна — для чтения, другая — для записи. Такую дисциплину доступа можно организовать с помощью **барьеров** (блокировок). При этом создается общий барьер для считывателей, так как несколько процессов могут одновременно считывать данные, а также отдельный барьер для процесса-писателя. Такая организация предотвращает взаимные помехи в работе.

**Голодание процессов**. Организация дисциплины доступа может привести к ситуации, когда процесс будет длительно ждать своей очереди для записи данных. Поэтому иногда нужно организовывать очереди с приоритетами.

Если нельзя точно определить, какой из процессов запрашивает или возвращает свои данные в нужный компьютер первым, используется так называемое взаимодействие по модели "клиент-сервер". При этом используются один или несколько клиентов и один сервер. Клиент посылает запрос серверу, а сервер отвечает на него. После этого клиент должен дождаться ответа сервера, чтобы продолжать дальнейшее взаимодействие. Такое поведение называется **управлением потоком**. При одновременном доступе здесь также могут использоваться очереди с приоритетами.

Классический **тупик** возникает, если процесс A получает доступ к файлу A и ждет освобождения файла B. Одновременно процесс B, получив доступ к файлу B, ждет освобождения файла A. Оба процесса теперь ждут освобождения ресурсов другого процесса и не освобождают при этом собственный файл.

**9. Концепция потока. Параллельное исполнение потоков. Главный поток процесса.**

**Поток выполнения** (англ. thread — нить) — наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы. Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, такие как память, тогда как процессы не разделяют этих ресурсов. В частности, потоки выполнения разделяют инструкции процесса (его код) и его контекст (значения переменных, которые они имеют в любой момент времени).

Термин "параллельность" рассматривается в контексте вытесняющей многозадачности операционной системы - то есть, операционная система выделяет потоку некоторый квант времени, а затем переключается на другой поток.

**Многопоточность**: процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть **без предписанного порядка во времени**.

По умолчанию процесс создается с одним потоком, называемым **главным** или основным потоком. В Linux потоки реализованы с помощью системного вызова **clone**(), который как минимум в 10 раз меньше занимает времени для создания еще одного потока, чем создание еще одного процесса при помощи **fork**(). Такая скорость достигается за счет того, что многие атрибуты процесса разделяются между потоками.

В Windows процесс может породить практически неограниченное количество потоков. Для этого используется функция **CreateThread**. Процесс будет активен, пока активен хотя бы один поток.

10. Диаграммы состояния потоков. Понятие контекста и переключения контекста.

Для каждого созданного потока в системе предусматриваются три возможных его состояния:

• состояние выполнения, когда код потока выполняется процессором; на однопроцессорных платформах в этом состоянии в каждый момент времени может находиться только один поток;

• состояние готовности к выполнению, когда поток готов продолжать свою работу и ждет освобождения ЦП;

• состояние ожидания наступления некоторого события; в этом случае поток не претендует на время ЦП, пока не наступит определенное событие (завершение операции ввода/вывода, освобождение необходимого потоку занятого ресурса, сигнала от другого потока), часто такие потоки называют блокированными. Изменение состояния потока происходит в результате соответствующих действий. Удобно для этих целей использовать следующую диаграмму состояний и переходов (рис. 2.2.5)



Переключение контекста (context switch) — в многозадачных ОС и средах — процесс прекращения выполнения процессором одной задачи (процесса, потока, нити) с сохранением всей необходимой информации и состояния, необходимых для последующего продолжения с прерванного места, и восстановления и загрузки состояния задачи, к выполнению которой переходит процессор. В процедуру переключения контекста входит так называемое планирование задачи — процесс принятия решения, какой задаче передать управление. При переключении контекста происходит сохранение и восстановление следующей информации:

• Регистровый контекст регистров общего назначения (в том числе флаговый регистр)

• Контекст состояния сопроцессора с плавающей точкой / регистров MMX (x86)

• Состояние регистров SSE, AVX (x86)

• Состояние сегментных регистров (x86)

• Состояние некоторых управляющих регистров (например, регистр CR3, отвечающий за страничное отображение памяти процесса) (x86)

11. Многозадачность в ОС. Типы многозадачности.

Многозадачность (multitasking) — свойство операционной системы или среды выполнения обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких задач. Истинная многозадачность операционной системы возможна только в распределённых вычислительных системах [1].

Существует два типа многозадачности[1]:

• Процессная многозадачность (основанная на процессах — одновременно выполняющихся программах). Здесь программа — наименьший элемент управляемого кода, которым может управлять планировщик операционной системы. Более известна большинству пользователей (работа в текстовом редакторе и прослушивание музыки).

• Поточная многозадачность (основанная на потоках). Наименьший элемент управляемого кода — поток (одна программа может выполнять 2 и более задачи одновременно)

К псевдопараллельной многозадачности можно отвести следующие типы: простое переключение, совместная (или кооперативная) многозадачность и вытесняющая (или приоритетная) многозадачность.

1. \*\*Простое переключение:\*\*

- Загружает в память два или более приложения, но предоставляет процессорное время только основному.

- Преимущества: Может использовать уже работающие программы без учета многозадачности. Недостатки: Ограниченное взаимодействие между программами.

2. \*\*Совместная многозадачность:\*\*

- Задача выполняется после объявления готовности отдать процессорное время другим задачам.

- Преимущества: Упрощает программирование, отсутствие необходимости в защите данных. Недостатки: Затруднена реализация многозадачной архитектуры ввода-вывода.

3. \*\*Вытесняющая многозадачность:\*\*

- Операционная система передает управление от одной программы к другой по различным событиям.

- Преимущества: Быстрый отклик на действия пользователя, полная реализация многозадачного ввода-вывода в ядре ОС. Недостатки: Требует дисциплины при написании кода, особые требования к реентерабельности и защите данных.

4. Реентерабельность

Реентерабельность программы означает, что её код может быть безопасно использован несколькими пользователями или процессами. Для обеспечения реентерабельности необходимо, чтобы вызываемый код не модифицировался, процедура не сохраняла информацию между вызовами, данные были уникальными для каждого пользователя, и не возвращались указатели на общие объекты. Реентерабельность важна для безопасности функций в многопоточных средах и программирования многозадачных систем, включая операционные системы.

12. Иерархия, приоритеты и планирование потоков. Динамические уровни приоритетов.

1. \*\*Иерархия и приоритеты потоков:\*\*

- \*\*Иерархия:\*\* Потоки часто группируются по уровням приоритета в зависимости от их важности и роли. Например, системные потоки могут иметь более высокий приоритет по сравнению с прикладными.

- \*\*Приоритеты:\*\* Каждому потоку присваивается приоритет, который определяет его относительную важность и влияет на решение о том, какие потоки будут выполнены в первую очередь.

2. \*\*Планирование потоков:\*\*

- \*\*Динамические уровни приоритетов:\*\* Некоторые системы поддерживают динамическое изменение приоритетов в зависимости от активности и состояния потоков. Например, при потреблении процессорного времени поток может повысить свой приоритет для более частого выполнения.

Цель планирования потоков вполне очевидна — определение порядка выполнения потоков в условиях внешней или внутренней многозадачности. Однако способы достижения этой цели существенно зависят от типа ОС. Рассмотрим сначала принципы планирования для универсальных ОС. Для таких ОС нельзя заранее предсказать, сколько и какие потоки будут запущены в каждый момент времени и в каких состояниях они будут находиться. Поэтому планирование должно выполняться динамически на основе сбора и анализа информации о текущем состоянии вычислительной системы.

\*\*Динамические уровни приоритетов в Windows:\*\*

В Windows каждый поток обладает динамическим приоритетом, который система может автоматически изменять для обеспечения отзывчивости и предотвращения голодания потоков. Динамический приоритет изначально совпадает с базовым приоритетом, и его система может повышать или понижать в зависимости от различных условий. Повышение приоритета происходит, например, при перемещении процесса на передний план, получении входных данных или выполнении условий ожидания. После повышения динамического приоритета система постепенно его уменьшает с течением времени. Этот механизм направлен на предотвращение инверсии приоритета и гарантирование эффективного использования процессорного времени.

13. Синхронизация и взаимоблокировка ресурсов. Механизмы синхронизации.

В некоторых операционных системах процессы, работающие совместно, могут сообща использовать некое общее хранилище данных. Каждый из процессов может считывать из общего хранилища данных и записывать туда информацию. Это хранилище представляет собой участок в основной памяти (возможно, в структуре данных ядра) или файл общего доступа. Местоположение совместно используемой памяти не влияет на суть взаимодействия и возникающие проблемы[1]. Ситуации, в которых два (и более) процесса считывают или записывают данные одновременно и конечный результат зависит от того, какой из них был первым, называются состояниями состязания.

Критическая область — часть программы, в которой есть обращение к совместно используемым данным. Соответственно, критический ресурс — тот ресурс, к которому осуществляется одновременный доступ

\*\*Механизмы синхронизации:\*\*

1. \*\*Критические секции:\*\*

- Области кода, где доступ к общим данным ограничивается, чтобы предотвратить конфликты.

- Предоставляют мьютекс для защиты от одновременного доступа нескольких потоков.

2. \*\*Мьютексы:\*\*

- Обеспечивают механизм блокировки, позволяющий потоку получить доступ к ресурсу или критической секции.

- Предотвращают конфликты и обеспечивают монопольный доступ к ресурсу.

3. \*\*Семафоры:\*\*

- Имеют счетчик, контролирующий количество потоков, которым разрешен доступ к ресурсу.

- Подходят для сценариев с ограниченным числом доступных ресурсов.

4. \*\*События:\*\*

- Используются для сигнализации между потоками.

- Могут быть использованы для синхронизации, где один поток ждет наступления события, а другой его сигнализирует.

5. \*\*Барьеры:\*\*

- Используются для синхронизации группы потоков, ожидающих друг друга перед выполнением следующего этапа работы.

6. \*\*Атомарные операции:\*\*

- Гарантируют неделимость выполнения операции, предотвращая переключение контекста между потоками.

- Обеспечивают безопасное выполнение операций над общими данными без необходимости блокировки.

14. Взаимоблокировка ресурсов в многозадачных системах. Решение задачи взаимоблокировки ресурсов.

Взаимоблокировка в группе процессов возникает в том случае, если каждый процесс из этой группы ожидает события, наступление которого зависит исключительно от другого процесса из этой же группы.

для возникновения ресурсных взаимоблокировок должны выполняться четыре условия:

1. Условие взаимного исключения. Каждый ресурс либо выделен в данный момент только одному процессу, либо доступен.

2. Условие удержания и ожидания. Процессы, удерживающие в данный момент ранее выделенные им ресурсы, могут запрашивать новые ресурсы.

3. Условие невыгружаемости. Ранее выделенные ресурсы не могут быть принудительно отобраны у процесса. Они должны быть явным образом высвобождены тем процессом, который их удерживает.

4. Условие циклического ожидания. Должна существовать кольцевая последовательность из двух и более процессов, каждый из которых ожидает высвобождения ресурса, удерживаемого следующим членом последовательности.

2.4.6. Решение задачи взаимоблокировки ресурсов

Чаще всего для борьбы с взаимными блокировками используются четыре стратегии: 1. Игнорирование проблемы. Может быть, если вы проигнорируете ее, она проигнорирует вас.

2. Обнаружение и восстановление. Дайте взаимоблокировкам проявить себя, обнаружьте их и выполните необходимые действия.

3. Динамическое уклонение от них за счет тщательного распределения ресурсов.

4. Предотвращение за счет структурного подавления одного из четырех условий, необходимых для их возникновения.

15. Компьютерное время. Ожидаемые таймеры.

Время Windows — это количество времени в миллисекундах, прошедшее с момента последнего запуска системы [5]. Этот формат в первую очередь предназначен для обеспечения обратной совместимости с 16-разрядной версией Windows. Чтобы обеспечить успешное выполнение приложений, предназначенных для 16-разрядной версии Windows, функция GetTickCount возвращает текущее время Windows.

Хотя система внутренне использует время в формате UTC, в приложениях обычно отображается местное время, которое является датой и временем суток для вашего часового пояса. Поэтому, чтобы обеспечить правильные результаты, необходимо знать, ожидает ли функция получать время в формате UTC или местное время, а также возвращает ли функция время в формате UTC или местное время.

Объект таймера ожидания — это объект синхронизации, состояние которого по достижении указанного срока устанавливается в значение Signaled. Существует два типа таймеров ожидания, которые можно создать: сброс вручную и синхронизация. Таймер любого типа также может быть периодическим [9]. Таймер сброса вручную Таймер, состояние которого остается сигнальным до вызова SetWaitableTimer, чтобы установить новое время выполнения. Таймер синхронизации Таймер, состояние которого остается сигнальным до тех пор, пока поток не завершит операцию ожидания в объекте таймера. Периодический таймер Таймер, который повторно активируется каждый раз, когда истечет указанный период, пока таймер не будет сброшен или отменен. Периодический таймер — это либо периодический таймер сброса вручную, либо периодический таймер синхронизации. Поток использует функцию CreateWaitableTimer или CreateWaitableTimerEx для создания объекта таймера. Поток создания указывает, является ли таймер таймером сброса вручную или таймером синхронизации. Создающий поток может указать имя объекта таймера. Потоки в других процессах могут открывать дескриптор для существующего таймера, указывая его имя в вызове функции OpenWaitableTimer. Любой поток с дескриптором объекта таймера может использовать одну из функций ожидания для ожидания, пока состояние таймера будет задано как сигнальное. Поток вызывает функцию SetWaitableTimer для активации таймера.

\*\*Компьютерное время в программировании:\*\*

1. \*\*Системное Время:\*\*

- \*\*Определение времени:\*\* Компьютеры поддерживают внутренние часы, отслеживающие текущее системное время.

- \*\*RTC (Real-Time Clock):\*\* Часы в реальном времени, хранящие время даже при выключенном компьютере.

2. \*\*Работа с Временем в Программах:\*\*

- \*\*API и Библиотеки:\*\* Языки программирования предоставляют API и библиотеки для работы с временем (например, `time` в Python, `java.time` в Java).

- \*\*Типы данных времени:\*\* Обычно используются структуры данных, представляющие момент времени, интервалы и т. д.

3. \*\*Форматы Времени:\*\*

- \*\*UNIX Timestamp:\*\* Количество секунд, прошедших с 1 января 1970 года (эпоха) в UNIX-подобных системах.

- \*\*Дата и Время в Календарной Нотации:\*\* Представление даты и времени в удобочитаемой форме (например, "2023-01-01 12:00:00").

4. \*\*Операции с Временем:\*\*

- \*\*Арифметика Времени:\*\* Добавление или вычитание интервалов времени.

- \*\*Сравнение Времени:\*\* Операции сравнения для определения порядка событий.

- \*\*Форматирование и Разбор Времени:\*\* Преобразование времени в строковый формат и обратно.

5. \*\*Часовые Пояса и Летнее/Зимнее Время:\*\*

- \*\*TimeZone:\*\* Учет различий в часовых поясах.

- \*\*DST (Daylight Saving Time):\*\* Учет переходов на летнее/зимнее время.

6. \*\*Таймеры и Задержки:\*\*

- \*\*Таймеры ОС:\*\* Использование системных таймеров для планирования событий.

- \*\*Задержки:\*\* Ожидание выполнения определенного условия или прошествия времени.

7. \*\*Системные Часы и Синхронизация:\*\*

- \*\*NTP (Network Time Protocol):\*\* Протокол синхронизации времени через сеть.

- \*\*Системные События:\*\* Обновление системного времени при включении, сетевых событиях и др.

16. Управление памятью: адресное пространство процесса, организация памяти, основные механизмы управления памятью, концепция рабочего множества.

Адресное пространство — это набор адресов, который может быть использован процессом для обращения к памяти.

\*\*Организация памяти и основные механизмы управления памятью:\*\*

1. \*\*Иерархия памяти:\*\*

- Компьютеры используют иерархию памяти с разными типами и характеристиками, включая кэш-память, оперативную память и дисковые накопители.

2. \*\*Менеджер памяти:\*\*

- Часть операционной системы, управляющая иерархией памяти, называется менеджером или диспетчером памяти.

- Задачи менеджера памяти включают выделение и освобождение памяти, отслеживание используемых областей и обеспечение эффективного использования ресурсов.

3. \*\*Физические адреса:\*\*

- Реальные адреса, используемые для выбора микросхем физической памяти.

- Организованы в виде последовательности 8-разрядных байтов.

4. \*\*Модели доступа к памяти:\*\*

- Сплошная ("плоская") модель памяти, сегментированная модель памяти, модель режима реального адреса.

- Различные модели использовались в различных поколениях компьютеров.

5. \*\*Организация физической памяти:\*\*

- Методы распределения памяти предоставляют программистам возможность эффективного использования компьютерной системы.

- Примеры включают ОЗУ (RAM), ПЗУ (ROM), и различные варианты размещения системных компонентов.

6. \*\*Замена данных (свопинг):\*\*

- Когда операционная система сохраняет содержимое памяти на диске перед загрузкой новой программы.

- Используется в простых встроенных устройствах и была распространена в ранних компьютерах.

7. \*\*Адресное пространство:\*\*

- Абстракция для памяти, позволяющая одновременное размещение в памяти нескольких приложений без взаимных помех.

- Понятие адресного пространства обеспечивает изоляцию программ.

Основная задача операционной системы в управлении памятью — создание абстракции для удобного моделирования и управления ресурсами памяти в компьютерной системе.

\*\*Концепция рабочего набора (набора) в контексте управления памятью:\*\*

- \*\*Рабочий набор процесса:\*\*

- Набор страниц в виртуальном адресном пространстве процесса, находящихся в физической памяти на текущий момент.

- Включает только страничные выделения памяти, исключая ресурсы, не поддерживающие подкачку.

- \*\*Обработка ошибок страницы:\*\*

- Если процесс обращается к странице, не входящей в рабочий набор, происходит ошибка страницы.

- Обработчик ошибки системной страницы пытается устранить ошибку, читая содержимое страницы из резервного хранилища.

- \*\*Ошибки мягкой страницы:\*\*

- Возникают, когда страница уже находится в памяти другого процесса, в состоянии перехода или при первом обращении к новой виртуальной странице.

- \*\*Удаление страниц из рабочего набора:\*\*

- Процессы могут уменьшать или очищать рабочий набор, вызывая соответствующие функции.

- Диспетчер памяти также может обрезать страницы из рабочего набора для создания дополнительной доступной памяти.

- \*\*Страницы перехода:\*\*

- Страницы, удаленные из рабочих наборов процессов, становятся страницами перехода.

- Сохраняются в оперативной памяти до повторного обращения или перепрофилирования.

- \*\*Запись грязных страниц:\*\*

- Грязные переходные страницы, измененные после последней записи на диск, могут быть записаны в их резервное хранилище.

- \*\*Размер рабочего набора процесса:\*\*

- Каждый процесс имеет минимальный и максимальный размер рабочего набора, влияющий на поведение подкачки виртуальной памяти.

- \*\*Интерфейс для получения и изменения размера рабочего набора:\*\*

- Функции GetProcessMemoryInfo, GetProcessWorkingSetSizeEx и SetProcessWorkingSetSizeEx используются для получения и управления размером рабочего набора.

Концепция рабочего набора играет ключевую роль в управлении виртуальной памятью, обеспечивая эффективное использование физической памяти и обеспечивая доступность данных для процессов.

17. Классификация запоминающих устройств. Иерархия памяти. Оперативные и постоянные запоминающие устройства.

Запоминающее устройство — носитель информации, предназначенный для записи и хранения данных. В основе работы запоминающего устройства может лежать любой физический эффект, обеспечивающий приведение системы к двум или более устойчивым состояниям.

Классификация запоминающих устройств по устойчивости записи и возможности перезаписи:

• Постоянные (ПЗУ), содержание которых не может быть изменено конечным пользователем (например, BIOS). ПЗУ в рабочем режиме допускает только считывание информации.

• Записываемые (ППЗУ), в которые конечный пользователь может записать информацию только один раз (например, CD-R).

• Многократно перезаписываемые (ПППЗУ) (например, CD-RW).

• Оперативные (ОЗУ) — обеспечивают режим записи, хранения и считывания информации в процессе её обработки. Быстрые, но дорогие ОЗУ (SRAM) строят на триггерах, более медленные, но более дешёвые разновидности ОЗУ — динамические ЗУ (DRAM) строят на элементах, состоящих из ёмкости (конденсатора) и полевого транзистора, используемого в качестве ключа разрешения записи-чтения. В обоих видах ЗУ информация исчезает после отключения от источника питания (например, тока).

По типу доступа ЗУ делятся на:

• устройства с последовательным доступом (например, магнитные ленты).

• устройства с произвольным доступом (RAM) (например, оперативная память).

• устройства с прямым доступом (например, жесткие магнитные диски).

• устройства с ассоциативным доступом (специальные устройства, для повышения производительности БД)

Классификация запоминающих устройств по геометрическому исполнению:

• дисковые (магнитные диски, оптические, магнитооптические);

• ленточные (магнитные ленты, перфоленты);

• барабанные (магнитные барабаны);

• карточные (магнитные карты, перфокарты, флэш-карты, и др.)

• печатные платы (карты DRAM).

Классификация запоминающих устройств по физическому принципу:

• перфорационные (перфокарта; перфолента);

• с магнитной записью (ферритовые сердечники, магнитные диски, магнитные ленты, магнитные карты);

• оптические (CD, DVD, HD-DVD, Blu-ray Disc);

• использующие эффекты в полупроводниках (флэш-память) и другие

\*\*Иерархия памяти в простых словах:\*\*

В компьютере есть разные виды памяти, и они разделены на уровни. В самом верху находятся очень быстрые регистры процессора. Они могут быстро предоставить данные, но их не так много.

После регистров идет кэш первого уровня — это небольшая, но очень быстрая память. Если процессор не находит нужные данные здесь, он смотрит в кэш второго уровня, который чуть больше и немного медленнее. Так продолжается до кэша третьего уровня, который уже весьма велик по объему, но не так быстр.

Если данные не найдены в кэше, процессор обращается к оперативной памяти. Она медленнее, но вмещает больше информации. Можно представить оперативную память как кэш для локального диска, который, в свою очередь, может рассматриваться как кэш для данных с удаленных серверов.

В конечном итоге, существует иерархия с разными уровнями памяти, от быстрой и маленькой до медленной и большой. Каждый уровень старается хранить копии данных, чтобы процессор мог получить к ним доступ быстро, минимизируя ожидание.

\*\*Простыми словами и вкратце Оперативные и постоянные запоминающие устройства:\*\*

\*\*ПЗУ (Постоянное Запоминающее Устройство):\*\*

ПЗУ - это место в компьютере, где хранятся важные программы, например, управление процессором, программы для запуска и выключения компьютера, а также тестовые программы для проверки работы различных блоков. Он также содержит информацию о том, где находится операционная система на диске.

\*\*ОЗУ (Оперативное Запоминающее Устройство):\*\*

ОЗУ - это место, где компьютер хранит данные и команды, которые ему нужны в данный момент для выполнения операций. ОЗУ передает данные процессору, обычно через кэш-память. Каждая ячейка в ОЗУ имеет свой уникальный адрес. Это временное хранилище, и его емкость может быть до 64 ГБ. ОЗУ устанавливается в виде модулей, таких как DIMM, и использует разные поколения DDR-памяти, которые отличаются по быстродействию и другим характеристикам.

