|  |  |  |  |
| --- | --- | --- | --- |
| 1. Операционные системы: опред-е, назначение, состав, ф-ии.  **ОС** — комплекс прог, который служит связующим звеном между аппаратными ресурсами компа и прикладными прогами.  Она управляет ресами компа, обесп. их эффективное исп-ие и организует взаимод-е с юзером. ОС предоставляет прикладным программам удобный интерфейс для работы с аппаратными ресами, скрывая от них сложность управления этими ресами. Конечные юзеры взаимод-ют с ОС через юзерский интерфейс, кот. может быть командной строкой или граф. интерфейсом.  **Она выполняет две основные функции:**  • предоставляет удобный интерфейс для работы с компом;  • управляет ресами компа.  **Состав** (подсистемы): упр-я процессами, упр-я памятью, упр-я файлами, упр-я внешними устройствами, юзерского интерфейса, защиты данных и администрирования.  **Функции:**  1. Координирует работу всех аппаратных средств компа (процессор, оперативная память, диски и т.д.).  2. Контролирует выполнение прог и их взаим-е с аппаратными устройствами.  3. Контролирует доступ к данным на различных носителях, таких как жесткий диск или компакт-диск, с помощью файл. системы.  4. Ведет файловую структуру: ОС организует файлы на компьютере в структурированном виде.  5. Обесп. диалог с юзером, позволяя ему взаимод. с компом.  6. Принимает команды от пользователя и обрабатывает их.  7. Загружает программы в оперативную память для выполнения.  8. Управляет запуском, приостановкой и остановкой программ. Координирует взаимодействие между различными задачами.  9. Управление файлами и базами данных.  10. Многозадачность.  11. Опред., какие задачи д. быть выполнены и в каком порядке.  12. Упр-ет операциями ввода и вывода (чтение/запись на диск).  13. Управляет использованием оперативной и вирт. памяти.  14. Обмен сообщениями между программами.  15. Защита и обеспечение сохранности данных.  16. Аутентификация, авторизация и другие меры безопасности.  17. Обработка сбоев.  18. Поддержка программирования. | 2. Операционные системы: классификация, основные этапы развития, особенности современного этапа развития.  **ОС** — комплекс прог, который служит связующим звеном между аппаратными ресурсами компа и прикладными прогами.  **Классификации:**  1. По алг-му упр-я процессором: однозадачные, многозадачные.  2. По кол-ву пользователей: однопольз-кие, многопольз-кие.  3. По типу устройства: локальные, сетевые.  4. По кол-ву процессоров: 1-процессорные, многопроцессорные.  5. Роль в сети: серверные, клиентские.  6. Тип лицензии: открытые (с открытым исходным кодом), проприетарные (с закрытым исходным кодом).  7. Сфера исп-ия: ОС мэйнфреймов (больших компов), серверов, ПК, мобильных устройств, встроенные ОС, ОС маршрутизаторов.  **Развитие операционных систем (ОС):**  1. (1945-1955): ОС отсутствовали. Компы работали на электрон. лампах и были примитивными. Программы писались в машинных кодах и загружались вручную.  2. (1955-1965): Компы начали использовать транзисторы, что увеличило их надежность и время работы. Появились первые системные проги, такие как компилятор Fortran.  3. (1965-1980): Использование интегральных схем, что улучшило качество и снизило стоимость компов. В это время были реализованы ключевые концепции, присущие современным ОС, такие как разделение времени и многозадачность, разделение полномочий, файловые структуры и файловые системы.  4. (1980-2005 гг.): Развитие больших интегральных схем, что привело к появлению микрокомпов или ПК. ОС этого поколения (CP/M и MS-DOS) были разработаны для этих микропроцессоров. По мере увеличения сложности задач, возникла необходимость в более сложных ОС, что привело к возрождению многих функций, характерных для больших вычислительных систем. Появились ОС с графическим интерфейсом, такие как Mac OS X, Windows и UNIX-подобные системы. Появление сетевых и распределенных ОС.  5. Пятое поколение операционных систем (2005 г. - по н.в.)  • Смартфоны.  • Многоядерные процессоры.  • Конкуренция мобильных ОС.  • Виртуализация и облачные технологии.  **Современные операционные системы (ОС), направления:**  1. Безопасность.  2. Многоплатформенность.  3. Удобство использования.  4. Прозрачность сетевых ресурсов. | 3. Компоненты архитектуры вычислительных систем, их назначение и взаимодействие.  **Классическая архитектура ЭВМ:**  • арифметическо-логическое устройство (АЛУ), выполняющее арифметические и логические операции;  • управ-щее устройство (УУ), организующее процесс вып-ия прог;  • внешнее запоминающее устройство (ВЗУ), или память для хранения программ и данных;  • оперативное запоминающее устройство (ОЗУ);  • устройства ввода и вывода информации (УВВ).    **Конфигурация типичного современного компьютера:**  1. Системный блок: блок питания, корпус, мат. плата, процессор (ЦП), жесткий диск, ОЗУ, видеоадаптер, звук. адаптер, приводы дисков, шлейфы, кабели.  2. Монитор: отображает инфу и графику компа пользователю.  3. Клавиатура: ввод текста и команд в комп.  4. Мышь: перемещение указателя и взаимодействия с UI.  5. Периферийные устройства: принтеры, сканеры, веб-камеры, акустические системы и другие устройства, которые расширяют функциональность компьютера.  Любая **шина** (соединение для передачи данных между функц. блоками компьютера) состоит из трех частей:  • шина данных, по которой передается информация;  • шина адреса, определяющая, кому передаются данные;  • шина управления, регулирующая процесс обмена информацией  Для связи между компонентами комп. системы исп. общая шина, которая состоит из неск. проводников и служит для передачи данных, адресов и управляющих сигналов. Шины м. б. внутр. (соединяющими процессор и АЛУ) и внешними (соединяющими процессор с памятью и устройствами ввода-вывода).  Контроллеры (устройство или прога для передачи данных между функц. блоками компа; напр., контроллер клавы получает инфу о нажатии клавиши и передает ее в ЦП для дальнейшей обр-ки) - специализированные устройства, кот. управляют внешними устройствами и обеспечивают связь между ними и ЦП компа. Они выполняют опред. ф-ии ввода-вывода и освобождают процессор от упр-я каждым устройством непосредственно. | 4. Система прерываний.  **ОС** – ПО, кот. регулирует работу компа и обесп. взаимодействие юзеров с аппаратными и программными компонентами системы.  Основная задача ОС закл. в управлении ресурсами компа, такими как процессор, память, диск. пространство и устройства ввода-вывода. ОС осуществляет распределение ресурсов между запущенными программами и обесп. их совместную работу.  **Система прерываний** обесп. врем. прекращение выполнения последовательности команд для перехода к выполнению др. последовательности команд или для возвращения к ранее прерванной программе.  **Общая классификация прерываний**: внешние - вызываются внешними по отношению к микропроцессору событиями (это группа аппаратных прерываний); внутренние - возникают внутри микропроцессора во время вычислительного процесса (по существу это исключительные ситуации и программные прерывания). Внешние возникают по сигналу какого-нибудь внешнего устройства. Внешние прерывания подразделяются на немаскируемые и маскируемые.  (+ аппаратные, программные, исключения)  Системный вызов: Пользовательская программа делает системный вызов для получения услуг от ОС. Этот вызов перехватывается внутри ядра ОС и запускает соответствующую службу.  **Реальный и защищенный режимы** процессора отличаются по следующим основным параметрам:  1. Реальный режим: обращение к опер. памяти происходит по реальным (действительным) адресам. Набор доступных операций не ограничен, и защита памяти не используется.  2. Защищенный режим: обращение к памяти происходит по вирт. адресам с использованием механизмов защиты памяти. Набор доступных операций определяется уровнем привилегий.  **Обработка прерывания в процессорах x86** состоит из трех этапов:  Обязательными для сохранения являются регистры cs, ip, flags  1. Прекращение выполнения текущей программы и сохранение ее состояния в стеке.  2. Переход к выполнению программы-обработчика прерывания, выбранной по номеру источника прерывания из таблицы векторов прерываний.  3. Возврат управления прерванной проге после восстановления ее состояния из стека.  Все прерывания и особые ситуации имеют уникальные номера ID, наз-ые **векторами прерываний**. Они исп-ся для идентификации и обработки различных типов прерываний. (0 до 31 отведены для особых ситуаций, от 32 до 255 свободны для любого исп-ия). |
| 5. Структура ядра и его функции. Объекты ядра. Основные операции над объектами ядра.  **Наиболее общим подходом к структуризации ОС является подразделение модулей 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**:  1. Внутрисистемн. функции: обесп. работу самой системы, такие как переключение между процессами, управление памятью и обработка прерываний. Не доступны для приложений.  2. Функции поддержки приложений: доступны для приложений и обесп. им доступ к ресурсам системы. Они формируют интерфейс прикл. прогр-ия (API), кот. исп. приложениями для обращения к ядру с запросами или системными вызовами.  **Ядро** ОС — это центр. часть ОС, кот. обеспечивает приложениям координированный доступ к ресам компа (процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации). Оно также предоставляет сервисы файловой системы и сетевых протоколов.  **Объекты ядра** — коллекция данных, кот-ми упр. ОС.  Монолитное, Модульное, Микроядро, Экзоядро  Типичные объекты режима ядра: объекты устройств; объекты файлов; символические ссылки; разделы реестра; потоки и процессы; объекты диспетчера ядра, такие как объекты событий и объекты мьютексов; объекты обратн. вызова; объекты section.  **Основные операции** над объектами ядра включают:  1. Создание объекта и дескриптора объекта.  2. Получение нового дескриптора существующего объекта.  3. Закрытие дескриптора объекта. Когда приложение закрывает последний дескриптор для объекта ядра, система удаляет объект из памяти. | 6. Утилиты. Системные обраб. программы. Библиотеки процедур. Программы дополнительных услуг.  **Утилиты** – вспомогательное (или служебное) ПО.  Утилиты – специализ. проги, кот. помогают обеспечивать эффективное функционирование комп. системы. Они поддерж. работу ОС, выполняя различные задачи, включая обнаружение и удаление вирусов, резервное копирование данных, управление файлами и дисками, и многое другое.  Утилиты облегчают доступ к различным ф-ям и настройкам, кот. могут быть сложными для изм-ия без специализ. инструментов. Они могут быть частью ОС, поставляться вместе с определенным оборудованием или распространяться отдельно.  **Типы**: 1. Антивирусы. 2. Системы упр-я файлами. 3. Инструменты сжатия. 4. Инструменты управления дисками. 5. Инструменты очистки диска. 6. Дефрагментация диска. 7. Утилита резервного копирования.  **Обрабатывающие сист. программы** выполняют 2 осн. функции: перенос инфы и её преобразование. Они делятся на: 1) утилиты, осн. ф-я кот. - перенос инфы, 2) лингвистические процессоры, осн. ф-я которых - перевод алг-ма с одного языка на другой.  **Лингвистические процессоры** вкл. трансляторы (компиляторы и ассемблеры) и интерпретаторы. **Трансляторы** преобразуют алг-м, записанный на ЯП, в алгоритм, записанный на машинном языке. Они делятся на компиляторы и ассемблеры. Компиляторы исп-ся для языков высокого уровня (Паскаль и Си), а ассемблеры - для языка ассемблера. Интерпретаторы и трансляторы - обраб-щие системные программы. Интерпретаторы переводят и выполняют каждый оператор исходной проги по очереди, в то время как трансляторы преобразуют всю программу целиком.  **Библиотеки** - наборы ф-ий, которые исп-ся в разл. прогах для выполнения опред. задач. Они м. б. стат-ми (встроенными в прогу) или динам-ми (загружаются в опер. память при необх-ти).  В Windows системные библиотеки хранятся в папках System32 и SysWOW64. В Linux библиотеки обычно находятся в папках /lib и /usr/lib. Путь к библиотекам указывается в файле /etc/ld.so.conf.  ОС могут включать разл. **проги доп. услуг**, такие как встроенные браузеры, почтовые клиенты, медиаплееры, граф. редакторы, антивирусное ПО и утилиты системы. Эти проги предоставляют возм-ти для просмотра веб-страниц, отправки и получения эл. почты, воспроизведения аудио и видео, редакт-ия изображений, защиты от вредоносных прог и управления сист. ресами. | 7. Понятие процесса. Системные и пользовательские процессы. Операции над процессами.  **Процесс** — экземпляр выполняемой проги, включая тек. значения счетчика команд, регистров и пер-ных. Концептуально у каждого процесса свой вирт. центр. процессор. Постоянное переключение между процессами наз. мультипрогр-ем, или многозадачным режимом работы.    а — 4 проги, раб-щие в многозадач. режиме; б — концептуальная модель 4 независ. друг от друга послед-ных процессов; в — в отдельно взятый момент активна только одна прога.  Процессы, кот. выполняют системный код, наз. **системными** и применяются к системе в целом. Они занимаются выполнением таких служеб. задач, как распред-ие памяти, обмен страницами между внутр. и вспомог. запомин-щими устройствами, контроль устройств и т.п.  **Пользовательские** процессы выполняют собств. код и иногда обращаются к сист. ф-ям. Выполняя собств. код, юзерский процесс пребывает в пользовательском режиме.  **Операции над процессами:**  Запуск процесса. Из числа процессов, находящихся в состоянии готовности, ОС выбирает 1 процесс для послед-щего исполнения.  Приостановка процесса. Работа процесса, наход. в состоянии исполнение, приостан-ся в рез-те 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 ОС нужно точно опред., какое именно событие произошло.  Переключение контекста. В действительности же деятельность мультипрограммной ОС сост. из цепочек операций, выполняемых над различными процессами, и сопровождается переключением процессора с одного процесса на другой. | 8. Организация межпроцессного взаимодействия в ОС. Сигналы. Каналы. Классич. проблемы межпроцессного взаимодействия.  При выполнении паралл. процессов может возникать проблема, когда каждый процесс, обращающийся к разделяемым данным, исключает для всех других процессов возм-ть одновременного с ним обращения к этим данным – это наз. взаимоисключением.  Ресурс, кот. допускает обслуживание только 1 пользователя за 1 раз, называется критическим ресурсом.  Для орг-ии коммуникации между одновременно работающими процессами примен-ся ср-ва **межпроцессного взаимодействия**.  Средства локального уровня IPC привязаны к процессору и возможны только в пределах компьютера. Удаленные IPC предоставляют механизмы, кот. обесп. взаимодействие как в пределах 1 процессора, так и между программами на различных процессорах, соединенных через сеть. Под высокоуровневыми IPC обычно подразумеваются пакеты ПО, кот. реализуют промежут. слой между системной платформой и приложением.  **Сигнал** в ОС сем-ва Unix — асинхронное уведомление процесса о каком-либо событии, 1 из осн. способов взаимодействия между процессами. Отдельные сигналы подразделяются на 3 класса: сист. сигналы; сигналы от устройств; сигналы, опред-ные юзером.  **Канал** предст. собой ср-во связи стандарт. вывода 1 процесса со стандарт. вводом другого. После создания канала, процесс может при помощи обычного сист. вызова write() выводить данные в него, а затем вводить их, вызывая соответственно функцию read(). При выполнении вызова fork() дескрипторы канала наследуются процессом-"потомком". Т. о., оба процесса получают возможность обмениваться данными.  **Проблемы:** синхр. доступ, дисциплина доступа, голод-е проц-сов.  Если нельзя точно определить, какой из процессов запрашивает или возвращает свои данные в нужный комп первым, исп-ся так называемое взаимодействие по модели "клиент-сервер". При этом исп-ся 1 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 взаимод-ие. Такое поведение наз. упр-ем потоком.  Классический тупик (взаимная блокировка) 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 освоб. при этом собств. файл. |
| 9. Концепция потока. Параллельное исполнение потоков. Главный поток процесса.  **Поток выполнения** — наим. единица обработки, исполнение кот. может быть назначено ядром ОС.  Несколько потоков выполнения могут сущ. в рамках 1 и того же процесса и совместно исп-ть ресы, такие как память, тогда как процессы не разделяют этих ресурсов. В частности, потоки выполнения разд-ют инструкции процесса (код) и его контекст (значения пер-ных, кот. они имеют в любой момент времени).  Термин **"параллельность"** рассм-ся в контексте вытесняющей многозадачности ОС – т. е., ОС выделяет потоку некоторый квант времени, а затем переключается на другой поток.  Многопоточность: процесс, порождённый в ОС, 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(), кот. 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 практ-ки неогран. кол-во 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 **3 возможных его состояния**: 1) состояние выполнения, когда код потока выполняется процессором; 2) состояние готовности к выполнению, когда поток готов продолжать свою работу и ждет освобождения ЦП; 3) состояние ожидания наступления некот. события; в этом случае поток не претендует на время ЦП, пока не наступит опред. событие (завершение операции ввода/вывода, освобождение необх-ого потоку занятого ресурса, сигнала от другого потока), часто такие потоки называют блокированными. Изм-ие состояния потока происх. в рез-те соответ-щих действий.    **Переключение контекста** — в многозадачных ОС и средах — процесс прекращения выполнения процессором 1 задачи (процесса, потока, нити) с сохранением всей необх. инфы и состояния, необх. для послед-щего продолжения с прерванного места, и восстановления и загрузки состояния задачи, к выполнению кот. перех. процессор. В процедуру переключения контекста входит так наз-ое планирование задачи — процесс принятия решения, какой задаче передать управление. При переключении контекста происходит сохр-е и восстановление следующей инфы:  • 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** — свойство ОС или среды выполнения обесп. возможность парал-ной (или псевдопараллельной) обработки неск. задач. Истинная многозадачность ОС возможна только в распределённых вычислительных системах.  Существует **два типа многозадачности**: процессная и поточная.  К псевдопарал-ной многозадачности можно отнести след. типы:  1. Простое переключение: загружает в память два или более приложения, но предоставляет процессорное время только основному. Преимущества: может исп. уже работающие проги без учета многозадачности. Недостатки: огранич. Взаимод-ие между программами.  2. Совместная многозадачность: обеспечивает метод, при кот. более 2 прог могут совместно исп-ть ресурсы и время общего хост-процессора. В рамках этого метода очередь должна иметь возможность одинаково распределять ресурсы процессора внутри прог. Это означает, что задачи должны освободить ЦП, позволяя запускать след. задачу. Задача выполняется после объявления готовности отдать процессорное время другим задачам. Преимущества: упрощает прогр-ие, отсутствие необх-ти в защите данных. Нед-ки: затруднена реализация многозадачной архитектуры ввода-вывода.  3. Вытесняющая многозадачность: ОС передает упр-е от одной прогb к другой по различным событиям. Решение принимается в соответствии с приоритетами задач. В отличие от кооперативной многозадачности, управление ОС передаётся вне зависимости от состояния работающих приложений, благодаря чему, в частности, зависшие приложения, как правило, не «подвешивают» ОС. За счёт регул. переключения задач также улучшается отзывчивость системы, оперативность освобождения ресов системы, которые больше не используются задачей. Преим-ва: быстрый отклик на действия юзера, реализация многозадачного ввода-вывода в ядре ОС. Недостатки: требует дисциплины при написании кода, особые требования к реентерабельности и защите данных.  Реентерабельность программы означает, что её код может быть безопасно использован неск. юзерами или процессами. Для обеспечения реентерабельности необх-мо, чтобы вызываемый код не модифицировался, процедура не сохраняла информацию между вызовами, данные были уникальными для каждого юзера, и не возвращались ук-ли на общие объекты. Реентерабельность важна для безопасности функций в многопоточных средах и программирования многозадачных систем, включая ОС. | 12. Иерархия, приоритеты и планирование потоков. Динам-ие уровни приоритетов.  Каждому потоку присваивается **приоритет**, который определяет его относительную важность и влияет на решение о том, какие потоки будут выполнены в первую очередь.  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.  **Планирование потоков:**  Некоторые системы поддерживают динам. изм-ие 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.  В Windows каждый поток обладает **динамическим приоритетом**, который система может автоматически изменять для обеспечения отзывчивости и предотвращения голодания потоков. Динам-ий 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 динам. приоритета система постепенно его уменьшает с течением времени. Этот механизм направлен на предотвращение инверсии приоритета и гарантирование эффект. исп-ия процессорного времени. |

|  |  |  |  |
| --- | --- | --- | --- |
| 13. Синхронизация и взаимоблокировка ресурсов. Механизмы синхронизации.  В некот. ОС 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. Ситуации, в которых два (и более) процесса считывают или записывают данные одновременно и конечный результат зависит от того, какой из них был первым, наз. состояниями состязания.  Критическая область — часть проги, в которой есть обращение к совместно используемым данным. Соотв-но, крит. ресурс — тот ресурс, к которому осуществляется одновременный доступ.  **Механизмы синхронизации:**  1. Критические секции - области кода, где доступ к общим данным огран-ся, чтобы предотврат. конфликты. Предоставляют мьютекс для защиты от одновременного доступа неск. потоков.  2. Мьютексы обесп. механизм блокировки, позволяющий потоку получить доступ к ресурсу или крит. секции. Предотвращают конфликты и обеспечивают монопольный доступ к ресурсу.  3. Семафоры имеют счетчик, контролирующий кол-во потоков, которым разрешен доступ к ресурсу. Подходят для сценариев с ограниченным числом доступных ресурсов.  4. События исп-ся 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 след. этапа работы.  6. Атомарные операции гарантируют неделимость выполнения операции, предотвращая переключение контекста между потоками и обесп. безопасное выполнение операций над общими данными без необх-ти 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 м. б. принудительно отобраны у процесса. Они должны быть явным образом высвобождены тем процессом, кот. их удерживает.  4. Условие циклического ожидания. Должна существовать кольцевая последовательность из двух и более процессов, каждый из кот. ожидает высвобождения ресурса, удерживаемого следующим членом последовательности.  **Решение задачи взаимоблокировки ресурсов:**  Чаще всего для борьбы с взаимными блокировками исп. 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** — кол-во времени в мс, прошедшее с момента последнего запуска системы. Этот формат в первую очередь предназначен для обесп-я обрат. совместимости с 16-разрядной версией Windows. Чтобы обеспечить успешное выполнение приложений, предназн-ых 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.  **Объект таймера ожидания** — это объект синхронизации, состояние кот. по достижении указанного срока устанавливается в значение Signaled.  Существует 2 типа таймеров ожидания, которые можно создать: сброс вручную и синхронизация. Таймер любого типа также может быть периодическим. **Таймер сброса вручную** – таймер, состояние кот. ост. сигнальным до вызова SetWaitableTimer(), чтобы устан-ть новое время вып-ия. **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.  **Компьютерное время в программировании:**  1. Системное время: опред-е времени (компы поддерживают внутренние часы, отслеживающие текущее системное время), RTC (Real-Time Clock) - часы в реальном времени, хранящие время даже при выключенном компе.  2. Форматы Времени: UNIX Timestamp (кол-во сек, 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")).  3. Операции с Временем: арифметика времени (добавление или вычитание интервалов времени), сравнение времени (операции сравнения для опред-ия порядка событий), форматирование и разбор времени (преобр-ие времени в строк. формат и обратно).  4. Час. пояса и Летнее/Зимнее Время: TimeZone (учет различий в часовых поясах), DST (Daylight Saving Time) - учет переходов на летнее/зимнее время.  5. Таймеры и Задержки: использование системных таймеров для планирования событий, ожидание выполнения опред. условия или прошествия времени. | 16. Управление памятью: адресное пространство процесса, организация памяти, основные механизмы управления памятью, концепция рабочего множества.  Каждое приложение (процесс) в ОС имеет свое собств. адресное пространство. Это означает, что каждому процессу назначается уник. набор адресов памяти, которые он может использовать. Это помогает изолир. процессы друг от друга и предотвр-ет случайное или намеренное повреждение данных 1 процесса др. процессом.  Адреса, кот. видит приложение, явл. адресами вирт. памяти. ОС и аппаратное обесп-е компа переводят эти вирт. адреса в физ. адреса памяти.  **Адресное пространство** - набор адресов, кот. может быть использован процессом для обращения к памяти. У каждого процесса имеется собств. адр. пространство, независимое от того адр. пространства, кот. принад. Др. процессам (за искл-ем тех особых обстоятельств, при кот. процессам требуется совместное)  Также оно может быть частным, а может быть общим (если оно не явл. общим (обычно относится к ситуации, когда два или более процесса имеют доступ к одним и тем же адресам в памяти. Это означает, что эти процессы могут читать и записывать в одни и те же места в памяти. Это может быть полезно для обмена данными между процессами)).  **Организация памяти** в компе - способ, которым ОС управляет и контролирует физ. и вирт. память.  Осн. ф-ии упр-я памятью включают: отслеживание свободной и занятой памяти, выделение памяти процессам при их создании и освобождение памяти при их завершении, вытеснение процессов из опер. памяти на диск, когда размеры осн. памяти недостаточны для размещения в ней всех процессов, и возвращение их в опер. память, когда в ней освобождается место, настройка адресов программы на конкретную область физической памяти.  **Основные механизмы управления памятью:** отслеживание свободной и занятой памяти,выд-ие памяти процессам, освоб-ие памяти при завершении процессов, вытеснение процессов из опер. памяти на диск, возвращение процессов в опер. память, настройка адресов проги на конкретную область физ. памяти.  **Рабочее множество** процесса - понятие, кот. опред. кол-во памяти, требующееся процессу в заданный интервал времени. Другими словами, рабочее множество процесса - набор страниц в вирт. адр. пространстве процесса, которые в данный момент находятся в физ. Памяти. |
| 17. Классификация запоминающих устройств. Иерархия памяти. Оперативные и постоянные запоминающие устройства.  **Запоминающее устройство** — носитель инфы для записи и хранения данных. В основе работы запомин. устройства может лежать любой физ. эффект, обеспечивающий приведение системы к двум или более устойчивым состояниям.  **По устойчивости записи и возм-ти перезаписи:** постоянные (ПЗУ), записываемые, многократно перезап-ые, оперативные (ОЗУ). **По типу доступа:** устройства с последовательным доступом, с произвольным доступом (RAM), устройства с прямым доступом, устройства с ассоциативным доступом. **По геом. исполнению:** дисковые, ленточные, барабанные, карточные, печатные платы. **По физ. принципу:** перфорац-ые перфокарта, перфолента,с магнитной записью,оптические, использующие эффекты в полупроводниках.  Соврем. система памяти образует **иерархию** от быстрых типов памяти маленького размера до медленных типов памяти большого размера. Конкретный уровень иерархии кэширует или является кэшем для данных, расположенных на более низком уровне. Это значит, что он содержит копии данных с более низкого уровня. Когда процессор хочет получить какие-то данные, он их сперва ищет на самых быстрых высоких уровнях. И спускается на более низкие, если не может найти высокий уровень содержит все данные маленьких. Когда идет поиск сначала ищет на самых больших, а потом идет к маленьким.  На вершине иерархии находятся регистры процессора. Далее идёт несколько килобайт кэш-памяти первого уровня (L1), доступ к которой занимает примерно 4 такта. Потом идёт пара сотен килобайт более медленной кэш-памяти второго уровня (L2). Потом несколько мегабайт кэш-памяти третьего уровня (L3). Далее расположена относительно медленная опер. память. Опер. память можно рассм. как кэш для локального диска. Она предназначена для врем. хранения данных и команд, необх. процессору для вып-я им операций. Опер. память передаёт процессору данные непосредственно, либо через кэш-память.  ПЗУ (постоянное запом. устройство) - тип памяти в компе, исп. для хранения инфы, кот. не меняется и которая нужна компу для работы, например, ПО для загрузки ОС.  Что там хранится: прога управления работой процессора, прога запуска и останова компа, проги тестирования устройств, проги управления дисплеем, клавой, принтером, внешней памятью, инфа о том, где на диске находится ОС. | 18. Виртуальная память. Структуризация адресного пространства виртуальной памяти. Задачи управления виртуальной памятью.  **Виртуальная память** - метод управления памятью компа, кот. позволяет выполнять проги, требующие больше опер. памяти, чем имеется в компе. Это достигается за счет автоматического перемещения данных из опер. памяти в файл подкачки, расположенный на жестком диске.  Виртуальная память решает следующие задачи:  - размещает данные в запоминающих устройствах разного типа, например, часть проги в опер. памяти, а часть на диске;  - перемещает по мере необх-ти данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в опер. память;  - преобразует виртуальные адреса в физические.  Все эти действия выполняются автоматически, без участия программиста, то есть механизм виртуальной памяти является прозрачным по отношению к пользователю.  Процессоры архитектуры x86-64 поддерживают два основных режима работы: Long mode («длинный» режим) и Legacy mode («унаследованный»). Эти режимы влияют на структуризацию адр. пространства вирт. памяти. **Long mode** - «родной» режим для процессоров x86-64, кот. позволяет исп. все преимущества 64-битной архитектуры. В этом режиме можно выполнять 64-битные проги, а также поддерживается выполнение 32-битного кода для обратной совместимости. **Legacy mode** позволяет процессору работать как 32-битный процессор x86, обеспечивая полную совместимость с 32-битным кодом и ОС. В этом режиме 64-битные программы и ОС работать не будут.  **Структуризация** адр. пространства вирт. памяти включает разделение его на части и динам. трансляцию адресов (позволяет выполнять программы размер которых превышает размер оперативной памяти).  Структуризация адр. пространства вирт. памяти включает в себя разл. методы, такие как сегментация и страничная адресация.  Сегментация: вирт. адр. пространство разделено на сегменты, каждый из которых имеет свой уник. адрес. Сегмент - условно выделенная область адр. пространства опред. размера.  Страничная адресация: вирт. и физ. память разбиваются на блоки фиксированной длины, называемые страницами. Страницы вирт. и физ. памяти нумеруются. Преобразование вирт. адресов в физ. выполняется с помощью таблицы страниц. | 19. Подкачка. Алгоритмы замещения страниц. Куча (heap). Стек.  **Файл подкачки** или виртуальная память — это способ системы виртуальной памяти увеличить оперативную память, когда ее не хватает для совершения операций. Система автоматически задействует файл подкачки, когда приложениям не хватит сист. памяти ОЗУ. Хотя система сама регул. объем файла подкачки иногда может понадобиться вручную увеличить вирт. память.  Когда в ОЗУ заканчивается место, ОС перемещает некот. данные из ОЗУ в файл подкачки. Это освобождает место в ОЗУ для выполнения тек. задач.  **Алг-мы замещения страниц** - методы, кот. ОС исп. для опред-я, какую страницу памяти следует удалить, когда требуется место для новой страницы. Вот некот. из наиболее важных алгоритмов:  1. Оптимальный алгоритм замещения страниц: выбирает для удаления страницу, которая будет использоваться позже всего.  2. Алг-м исключения недавно использовавшейся страницы (LRU): выбирает для удаления страницу, кот. не исп-сь дольше других.  3. Алгоритм FIFO: выбирает для удаления страницу, которая находилась в памяти дольше других.  4. Алгоритм «второй шанс»: модификация алгоритма FIFO, кот. проверяет, была ли стр-а недавно исп-на перед ее удалением.  5. Алгоритм «часы»: модификация алг-ма FIFO, кот. использует “стрелку часов” для отслеживания страниц.  6. Алгоритм замещения наименее востребованной страницы (LFU): выбирает для удаления страницу, кот. исп-сь реже всего.  Алгоритм LRU обычно работает хорошо, но может быть дорогим в реализации. Алгоритм FIFO прост в реализации, но может привести к частым ошибкам отсутствия страниц  **Куча** - область памяти, которой можно динамически управлять.  Это как своб. пул памяти, кот. можно исп-ть, когда запускаете приложение. Размер кучи опред-ся ограничениями опер. памяти и обычно больше, чем размер стека. Мы используем кучу, когда не знаем, сколько памяти займет структура данных в нашей программе, когда нам нужно выделить больше памяти, чем доступно в стеке.  **Стек** - область памяти, где данные добавляются и удаляются по принципу LIFO. Это значит, что вы можете получить доступ только к последнему элементу, который был добавлен в стек.  Если в стек записывается слишком много данных, он может “вырасти” за свои пределы, что приводит к переполнению стека.  Чтение из стека наз. «извлечение» (pop), а запись в стек — «вталкивание» (push). | 20. Типы устройств ввода/вывода. Обраб-ка внешних прерываний. Синхронный и асинхронный ввод/вывод.  **Устройства ввода/вывода** — компонент типовой архитектуры ЭВМ, предоставляющий компу возможность взаимодействия с внешним миром и, в частности, с пользователями.  По направлению передачи информации подразделяются на:  - устройства ввода (клавиатура\мышь\тачпад),  - устройства вывода (монитор \принтер\проектор),  - устройства ввода/вывода (дисковод\модем) обычно состоят из механ. и электрон. компонента, где эл. комп-т наз. контроллером.  Раздел. условно на блочные устройства и символьные устройства.  - К блочным относятся такие устройства, кот. хранят инфу в блоках фиксированной длины, у каждого из кот. есть собств. адрес.  Важным св-вом блочного устройства явл. то, что оно способно читать или записывать каждый блок независимо от всех других блоков. Среди наиболее распр. блочных устройств жесткие диски, приводы оптических дисков и флеш-накопители USB.  - Симв. устройства выдают или воспринимают поток символов, не относящийся ни к какой блочной структуре. Не явл. адресуемыми и не имеют никакой операции позиционирования. В кач-ве симв. устройств могут рассматриваться терминалы, принтеры.  **Внешние прерывания** - сигналы, которые поступают к процессору от внешних устройств, таких как периферийные устройства. Эти сигналы могут произойти в любой произв. момент и обычно связаны с событиями, такими как сигнал от таймера, сет. карты или дискового накоп-ля, нажатие клавиш клавы, движение мыши  Обслуживание прерывания состоит из двух, а иногда и 3 этапов.  - Быстрое сохранение непостоянной информации.  - Обработка сохраненных переменных данных.  - Выполнение дополнительных работ.  **Асинхронный ввод-вывод** используется там, где можно оптимизировать производительность приложения. При асинхр. вводе-выводе приложение инициирует операцию ввода-вывода, а затем может продолжить свою работу.  При **синхронном вводе-выводе** приложение блокируется до завершения выполнения операции ввода-вывода. С точки зрения вызывающего потока асинхр. ввод-вывод более эффективен, поскольку позволяет продолжать выполнение, в то время как операция ввода-вывода ставится диспетчером ввода-вывода в очередь и впоследствии выполняется. Однако приложение, использующее асинхронный ввод-вывод, требует механизма определения завершенности этой операции. |
| 21. Файловые системы. Файлы и директории. Управление внешней памятью.  **Файловая система** — порядок, опред-щий способ организации, хранения и именования данных на носителях инфы в компах, а также в др. электрон. оборудовании. Опр-т формат содержимого и способ физ. хранения инфы, кот. принято группировать в виде файлов. Конкретная ФС опр-т размер имен файлов (и каталогов), макс. Возм. размер файла и раздела, набор атрибутов файла.  Вот некоторые из наиболее распространенных типов файловых систем: 1) NTFS поддерживает метаданные и исп. журналир-ие для повышения надежности; 2) FAT32 исп. для совместимости, особенно на съемных накопителях.  Журналируемые — сохраняет историю действий юзера, а также план проверки системы в специальном файле. Устойчивость к сбоям и сохранение целостности инфы.  Не журналируемые — не предусматривают хранение логов. Особ-ти: работают быстрее, но не гарант. сохранность данных.  **Файл** - поименованная совок-ть данных, обычно размещаемая на внешних запоминающих устройствах. **Директории** исп. для организации файлов. Могут содерж. файлы и другие директории, создавая иерархическую структуру, известную как ФС. Помогают упорядоч. файлы, делая их поиск и управление более удобными.  Можно использовать абс. пути, кот. указывают, как найти файл от корневого каталога, или отн. пути, которые указываются относительно текущего рабочего каталога: (папка относительно проги)/source/img.png.  **Управление внешней памятью** - процесс, кот. организует эффект. работу с данными, хранящимися во внешней памяти, и обесп. юзеру удобный интерфейс при работе с такими данными.  В Windows, управление внешней памятью осуществляется путем создания и управления файлом подкачки, также известного как файл подкачки или страницы файла подкачки.  В ранних ОС, управление памятью сводилось просто к загрузке проги и ее данных из некот. внешнего накопителя (перфоленты, магнитной ленты, магнитного диска) в опер. память.  В Windows 10, вы можете управлять памятью через “Параметры” -> “Система” -> "Память устройства". Также можно настроить опер. память через "Редактор реестра". | 22. Принципы организации и структура ОС Windows. Обзор версий Windows. Методы инсталляции ОС Windows.  **Архитектура ОС Windows** вкл. ядро ОС, сист. службы и прил-ия. На самом низком уровне ОС ядро состоит из самого ядра Windows и драйверов устройств низкого уровня. Ядро отвечает за прием запросов ОС от сист. служб. Затем ядро преобразует эти запросы в инструкции для аппаратного обесп-я компа, включая центр. процессор (ЦП), память и аппарат. устройства. При запуске ОС сначала инициал-ся ядро и связанные с ним низкоуровневые драйверы устройств, а затем службы ОС. Службы явл. частью ОС, а не компонентами, кот. вы устанавливаете после развертывания ОС. Кроме того, службы функционируют без каких-либо действий со стороны юзера. На верхнем уровне ОС приложения работают путем взаимодействия с юзером компа, а на нижнем уровне — путем интеграции со службами ОС.  **Реестр** — БД, в кот. Windows хранит пар-ры конфигурации юзера и компа. Всякий раз, когда вы вносите изм-ия в конфигурацию Windows, это изменение фиксируется в реестре.  **Последняя LTSC-версия Windows Server** сегодня - Windows Server 2022. Также поддерживается и WS 2019. WS 2022 обеспечивает расширенную многоуровневую безоп-ть, гибридные возм-ти с Azure и гибкую платформу прил-ий. Он имеет встроенн. возм-ти защищенного ядра, помогающие защитить оборуд-е, встроенное ПО и возможности ОС WS от расширенных угроз безопасности. 3 редакции WS 2022: Standard, Datacenter, Datacenter: Azure Edition.  **Различные выпуски Windows** удовлетв. потреб-ти потребителей, от частных лиц до крупных предприятий. Home - индивидуальное домашнее исп-е. Pro - малый и средний бизнес, продвинутые юзеры, юзеры с повыш. требованиями к производительности и хранилищу. Enterprise - крупные коммерческие организации. Enterprise LTSC - круп. корпоративн. орг-ции с ограничительными требованиями к изменениям. Pro Education - сравнимо с Pro для сотрудников школы, админов, учителей и учащихся. Education - сравнимо с Enterprise для школ. персонала, админов, учителей и учащихся. IoT Core/ Enterprise - Устройства стационарного назначения и бытовые устройства.  Существует **неск. разных вариантов развертывания** сервера Windows и настольных ОС бизнес-класса. Это могут быть ручная установка, полностью автономные (не обслуж-мые) установки, автономные готовые или дополнительно настроенные установки, а также развертывания заготовленных и, возможно, специально настроенных образов ОС. | 23. ОС Windows: организация рабочей среды юзера, работа с учетн. записями юзеров и групп, работа с профилями юзеров.  **Рабочая среда юзера** состоит из: 1) настроек раб. стола (цвета экрана, настроек мыши, размера и расположения окон); 2) настроек процесса обмена инфой по сети и с устр-вом печати; 3) пер-ных среды; 4) пар-ров реестра; 5) набора доступ-х прил-ий.  Для упр-я средой юзера предназначены след. средства Windows:  1. Сценарий входа в сеть: командные файлы (.bat) или испол-ые файлы (.exe), выполняемые при входе юзера в сеть, могут вкл. команды для создания сет. соединения или запуска прил-ий.  2. Профили юзеров: хранят настройки раб. среды юзера (среды компа, кот. определены юзером).  3. Сервер сценариев Windows (WSH): независим от языка и предн. для работы на 32-разрядных платформах Windows. Вкл. в себя ядро сценариев VBScript и JScript. Позволяет выполнение сценариев на рабочем столе Windows или в консоли команд.  **Локальные учетные записи юзеров** определяются локально на устройстве и могут назначаться только на этом устройстве.  **Учетная запись локального админа по умолчанию** — учетная запись юзера для сисадминистрирования. Им. полный контроль над файлами, каталогами, службами и др. ресурсами на локал. устройстве. Учетная запись админа может создавать других локал. юзеров, назначать права юзеров и назначать разрешения.  **Гостевая учетная запись** позв. случайным или разовым юзерам, у кот. нет учетной записи на компе, временно войти на локал. сервер или клиентский комп с ограниченными правами юзера.  Система создает **профиль юзера** при первом входе юзера на комп. При послед. входах система загружает профиль юзера, а затем другие сист. компоненты настраивают среду юзера в соответствии с инфой в профиле.  Типы профилей: лок., перемещаемые, обяз-ые, временные .  Элементы профиля:  Куст реестра: Файл NTuser.dat, загружается при входе, содержит настройки и конфигурацию реестра.  Папки профилей в файловой системе: хранят файлы профиля, такие как документы, настройки приложений.  Преимущества профилей пользователей:  Сохранение параметров между входами юзера.  Настроенный рабочий стол для каждого юзера при совместном использовании компа.  Уникальность параметров в профиле для каждого юзера.  Изменения в профиле одного юзера не влияют на др. юзеров. | 24. Системный реестр ОС Windows, его назначение и исп-ие.  **Реестр** — БД, в кот. Windows хранит пар-ры конфигурации юзера и компа. Всякий раз, когда вы вносите изменения в конфигурацию Windows, это изменение фиксируется в реестре.  **Куст** – логич. группа ключей, подразделов и значений в реестре, кот. содержит набор вспомог. файлов. Эти файлы загружаются в память при запуске ОС или при входе юзера в систему.  **Ключи реестра** - контейнеры, кот. действуют как папки со значениями или подразделами, содержащимися в них.  **Подразделы** аналогичны подпапкам в файловой системе  Значения реестра аналогичны файлам (не контейнерам) - они хранят фактические данные (настройки), кот. исп-ся ОС и устан-ми прогами.  Реестр Windows организован иерархически. На верхнем уровне имеется 5 кустов реестра, кот. предст. собой отд. набор связанных пар-ров, структур-ных как серия ключей, подразделов и значений:  HKEY\_CLASSES\_ROOT: ассоциации файлов и прил-ий для открытия файлов.  HKEY\_CURRENT\_USER: инфа о конфигурации тек. юзера.  HKEY\_LOCAL\_MACHINE: важные пар-ры конфигурации компа.  HKEY\_USERS: инфа о конфигурации всех юзеров, включая тек.  HKEY\_CURRENT\_CONFIG: инфа о тек. профиле оборудования.  Для поддержания структуры БД аналогичные настройки хранятся в папках и подпапках, известных как ключи и подразделы. Это упрощает ссылку на опред. значение реестра. Вы можете указать путь, объявив соответств-щий куст, ключ, подразделы и значение.  REG\_BINARY. Необработанные двоичные данные. Эти значения обычно отображ. в 16-ричном формате. Инфа об оборудовании часто хранится в значениях REG\_BINARY.  REG\_DWORD. 4-байтовые числа (32-битное целое число). Многие значения, связанные с драйверами устройств и службами, хранятся в значениях REG\_DWORD.  REG\_SZ. Текстовая строка фиксированной длины.  REG\_EXPAND\_SZ. Текстовая строка переменной длины.  REG\_MULTI\_SZ. Несколько строковых значений. |

|  |  |  |  |
| --- | --- | --- | --- |
| 25. ОС Windows: планирование и назначение разрешений NTFS.  **Идентификатор безоп-ти (SID)** — уник. значение пер-ной длины, используемое для идентификации доверенного лица. Каждая учетная запись имеет уник. айди безоп-ти, выданный центром сертификации, таким как контроллер домена Windows, и хранящийся в БД безоп-ти. Каждый раз, когда юзер входит в систему, система получает айди безоп-ти для этого юзера из БД и помещает его в маркер доступа для этого юзера. Система исп. SID в маркере доступа для идентификации юзера во всех послед. взаимод-ях с безоп-тью Windows. Если SID исп-ся в кач-ве уник. айди юзера или группы, его нельзя исп. повторно для ид-ции другого юзера или группы.  Безоп-ть Windows исп. айди безоп-ти в след. эл-тах безоп-ти:  • в дескрипторах безоп-ти для идент-ции владельца объекта и основной группы;  • в записях контроля доступа для определения доверенного лица, для которого разрешен, запрещен или проверен доступ;  • в маркерах доступа для идентификации юзера и групп, к кот. принадлежит юзер.  **Маркер доступа** — объект, описывающий контекст безоп-ти процесса или потока. При каждом входе юзера система создает маркер доступа для этого юзера. Маркер доступа содержит айди безоп-ти юзера, его права и SID для всех групп, к кот. принадлежит юзер. Этот маркер предост. контекст безоп-ти для любых действий, выполняемых юзером на этом компе.  **Список упр-я доступом (ACL)** — список записей упр-я доступом (ACE). Каждая запись ACE в ACL идентифицирует доверенное лицо и указывает разрешенные, запрещенные или проверяемые права доступа для этого лица. Дескриптор безоп-ти защищ-го объекта может содержать 2 типа списков ACL: DACL и SACL.  **Список упр-я дискреционным доступом (DACL)** опред. довер. лиц, которым разрешен или запрещен доступ к защищаемому объекту. Когда процесс пытается получить доступ к защищ. объекту, система проверяет записи ACE в списке DACL объекта, чтобы опред., предоставлять ли к нему доступ.  **Список упр-я доступом к системе (SACL)** позволяет админам регистрировать попытки доступа к защищ. объекту. Каждый ACE определяет типы попыток доступа со стороны указанного доверенного лица, кот. заставляют систему создавать запись в журнале событий безопасности. | 26. Средства автоматической настройки в ОС Windows.  В Windows есть две оболочки ком. строки: cmd и PowerShell.  Каждая оболочка — прога, кот. обесп. прямую связь между оператором и ОС или приложением, предоставляя среду для автоматизации ИТ-операций.  **cmd** была 1-ой оболочкой, встр. в Windows, для автоматизации повседн. задач, с пакетными (.bat или .cmd) файлами.  **PowerShell** был разраб. для расширения возм-тей ком. оболочки для выполнения команд PowerShell, называемых командлетами. Командлеты похожи на команды Windows, но предоставляют более расширяемый язык сценариев.  **BAT-файл** – текстовый файл, содержащий команды системы, кот. могут выполняться последовательно. Комменты начинаются с REM (remark). Строки, начинающиеся с @(лигатура), не выводятся на экран. Допускаются программные конструкции. Для вызова другого скрипта используется команда call. Для создания пер-ной используется команда set.  **Инструментарий управления Windows (WMI)** — инфраструктура для упр-я данными и операциями в Windows. Он представляет собой реализацию стандарта WBEM, ориент-ого на создание универс. интерфейса мониторинга и упр-я разл. системами.  **Event Tracing for Windows (ETW)** — служба, кот. позволяет получать события от 1 или неск. поставщиков событий в режиме реального времени или из файла \*.etl за некот. врем. период.  **Групповая политика** — важный элемент любой среды Microsoft Active Directory (AD). Её осн. цель — дать ИТ-админам возм-ть централизованно управлять юзерами и компами в домене. Сост. из набора политик, называемых объектами групп. политики (GPO). При создании домена AD автом-ки создаются два объекта групп. политики: п-ка домена и п-ка контроллеров домена.  **Административные шаблоны** (файлы) в ф-те XML, написанные на разл. языках, кот. задают пар-ры групповых политик, основанные на значениях реестра. Позв. админам устанавливать и изменять настройки безопасности и поведения на компе или сети через групп. политики. Два вида: 1) ADMX — указывает кол-во и тип пар-ров политики, распол-ие по категориям согласно отобр-ю файла в редакторе лок. групповых политик; 2) ADML — предост. связанные с языком сведения для ADMX-файла. Позв. параметру политики отображаться в редакторе лок. групповых политик на нужном языке. | 27. Основные понятия системы UNIX. Пользователи системы, атрибуты пользователя. Файловая структура ОС.  **Unix** - семейство ОС, обладающих сходной архитектурой и интерфейсом с юзером. Современные варианты: Linux, BSD, AIX, HPUX, Solaris, SCO.  Важнейшие стандарты, обесп-щие целостность семейства UNIX:  POSIX (Portable Operating System Interface) и ANSI С (c89 и c99)  Классическая архитектура UNIX двухуровневая  **Ядро** — управляет ресами компа и предлагает прогам базовый набор услуг (системные вызовы). Системные проги (управление сетью, терминалами, печатью), прикладные проги (редакторы, утилиты, компиляторы и т.д.). ОС UNIX обладает классич. монолитным ядром, в кот. можно выделить след. осн. части: файловая подсистема, упр-е процессами и драйверы устройств.  Функции ядра: 1) инициализация системы; 2) упр-е процессами и потоками; 3) упр-е памятью; 4) упр-е файлами; 5) обмен данными между процессами; 6) программный интерфейс (API).  Ядро изолирует программы пользователя от аппаратуры.  **Оболочка** - прога, кот. позволяет связываться с ОС. Она считывает команды, кот. вы вводите, и интерпретирует их как запросы на выполнение других программ, на доступ к файлу или обеспечение вывода. Оболочка также поддерживает мощный ЯП, кот. обесп. условное выполнение и упр-е потоками данных.  Прежде чем клиент сможет начать работу с ОС UNIX, он должен стать пользователем системы, т.е. получить имя, пароль и ряд других атрибутов. **Пользователь** - объект, кот. обладает опред. правами и может запускать на выполнение проги и владеть файлами. В системе сущ. 1 юзер, облад-щий неогран. правами - суперюзер или админ системы (root)  **UID** – уник. айди юзера. **Группа** — список юзеров, имеющих сходные задачи. **GID** – каждая группа имеет уник. имя, а система разл. группы по групповому айди.  **Файл** в Linux — послед-ть байтов произв. длины (от 0 до некот. максимума), содержащая произв. инфу. Не делается различия между текст. файлами, двоич. файлами и любыми др. типами файлов. Значение битов в файле целиком опред-ся владельцем файла. Имена файлов ограничены 255 символами. В именах файлов разрешается исп. все ASCII-символы, кроме NULL.  Для удобства файлы могут группироваться в каталоги. Каталоги хранятся на диске в виде файлов. Корневой каталог наз. /. | 28. ОС UNIX: особ-ти процессов, сигналы, обработка сигналов.  Осн. активными сущностями в Linux явл. **процессы**. Каждый процесс выполняет 1 прогу и изначально получает 1 поток упр-я. Иначе говоря, у процесса есть 1 счетчик команд, кот. отслеживает след. исполняемую команду. Linux позволяет процессу создавать доп. потоки (после того, как он начинает выполнение).  FORK создает точную копию исходного процесса, называемого род, процессом. Новый процесс наз. дочерним процессом. У род. и у доч. процессов есть собств. образы памяти. Если род. процесс впоследствии изменяет какие-либо свои пер-ные, то эти изм-я остаются невидимыми для доч. процесса и наоборот.  fork возвращает доч. процессу число 0, а род. — отличный от нуля PID дочернего процесса.  **Типы процессов:**  Сист. процессы - часть ядра, всегда в опер. памяти. Не имеют соответствв. исполняемых файлов, запускаются особым образом при инициализации ядра. Примеры: shed, vhand, bdfflush и init.  Демоны – фон. процессы, работающие в фон. режиме и обслуж-щие разл. сист. задачи или услуги. Они часто запускаются при загрузке системы, работают незав-мо от активности конкр. юзера.  Прикладные процессы - остальные, выполняющиеся в системе, часто порождаемые юзерским сеансом. Время их выполнения ограничено сеансом работы.  **Сигналы** — способ информирования процесса со стороны ядра о происшествии некот. события. 1 процесс может посылать сигнал другому через системный вызов kill.  **Обработка сигналов** в Linux осущ. через установку обработчиков сигналов. Когда процесс получает сигнал, ядро отправляет управление к обработчику, кот. был предварительно установлен для этого сигнала. Процесс может выбрать разл. действия при получении сигнала: 1) игнорирование (процесс может проигнор. сигнал, если установлен обработчик SIG\_IGN); 2) по умолчанию (если не установлен юзерский обработчик, применяется действие по умолчанию); 3) юзерский обработчик (функция): процесс может установить свою собств. функцию-обработчик. Когда сигнал поступает, управление передается этой функции.  Сигналы явл. прогр-ыми прерываниями. 1 процесс может послать другому так называемый сигнал. Процессы могут сообщить системе, какие действия следует предпринимать, когда придет входящий сигнал. |
| 29. Основные принципы функционирования Linux. Основные компоненты Linux. Дистрибутивы Linux. ФС Linux.  Каждый юзер работает с системой через вирт. терминалы, кот. может быть до 12-ти. Существует возможность изменять существующие интерфейсы и создавать свои собств.  В Linux все файлы организованы в непрерывный поток байтов. Данные, вводимые с клавы, представляют собой входной поток данных, а отображаемые данные – выходной поток. Они могут свободно взаимод-ть с файлами монитор и клава рассм-ся системой как файлы.  **Linux обладает разл. комп-тами и возм-ями:** 1) Планировщик; 2) Файл подкачки; 3) Модули; 4) ФС; 5) Механизмы защиты; 6) Инструменты администрирования; 7) Серверные возм-ти; 8) Система упр-я пакетами.  Мы можем распределить **дистрибутивы Linux** на три группы:  1) Enterprise Grade Linux. Предоставляют более старые, но стабильные версии программного обеспечения.  2) Consumer Grade Linux. Более быстр. внедрение нов. техн-ий.  3) Experimental and Hacker Linux. Предоставляют возможность для тестирования будущих функций.  Дистрибутивы состоят из отдельных пакетов, каждый из кот. содержит какое-то прил-е, утилиту или сервис. Отдельный пакет может содержать, например, веб-браузер, библиотеку для работы с граф. файлами в формате PNG, набор шрифтов и так далее. ПО, содержащееся в пакете, поставляется в одном из двух основных видов: 1) в виде бин. файлов; 2) в виде исходных текстов.  **ФС в Linux** как и в большинстве др. систем имеет иерархическую (древовидную) структуру. Все объекты являются файлами, в том числе и директории для организации доступа к файлам.  Существуют след. типы файлов: 1) обычные файлы; 2) каталоги; 3) символьные ссылки; 4) блочные устройства; 5) символьные устройства; 6) сокеты, 7) каналы.  FHS (Filesystem Hierarchy Standard) – стандарт ФС. В FHS все файлы и каталоги находятся внутри корневого каталога, даже если они расположены на различных физических носителях. | 30. ОС Linux: управление процессами, выполнение задач в фон. режиме, изменение приоритетов выполняющихся программ.  **Каталог /proc** – вирт. файловая система, его содержимое не записывается на диск, а находится в памяти. Динамически заполняется каждый раз при загрузке компа и постоянно отражает тек. состояние системы.  Каталог /proc содержит инфу о: 1) выполняющихся процессах; 2) конфигурации ядра; 3) аппаратном обеспечении.  Файл /proc/cpuinfo «хранит» инфу о процессоре системы.  Файл /proc/cmdline хран. строки, переданные ядру при загрузке.  Файл /proc/modules показ. список модулей, загруженных в ядро.  Каталог /proc/sys содержит параметры конфигурации ядра.  Файлы распределены по катег-ям, для каждой кат-ии создан свой подкаталог. Процессы существуют в иерархии: после загрузки ядра в память запускается первый процесс (init или systemd), который, в свою очередь, запускает другие процессы, которые, опять же, могут запускать другие процессы. Процесс имеет уник. идентификатор (PID) и идентификатор род. процесса (PPID).  Команда top динамически отображает все запущ. процессы. Команда ps выводит статическую инфу о процессах. Без опций выводит только процессы, относящиеся к текущей оболочке. Завершить процесс можно с помощью команды kill.  Для **вып-ия команды в фоновом режиме** достаточно добавить в конце символ амперсанда &.  Узнать состояние всех остановленных и выполняемых в фоновом режиме задач в рамках текущей сессии терминала можно при помощи утилиты jobs c использованием опции -l. В любое время можно вернуть процесс из фонового режима на передний план. Для этого служит команда fg. Если в фон. режиме выполняется неск. прог, следует также указывать номер.  **Изменение приоритетов выполняющихся программ**  Приоритет nice — число, указывающее планировщику процессов ядра ОС приоритет, который пользователь хотел бы назначить процессу. Утилита nice, запущенная без аргументов, выводит приоритет nice, унаслед-ый от род. процесса. nice принимает аргумент «смещение» в диапазоне от -20 (наивысший приоритет) до +19 (низший приоритет). Посмотреть приоритет процессов можно, например с помощью утилиты top. Чтобы установить значение nice ниже 0, требуются права суперюзера. | 31. Понятие безопасности ОС. Основные угрозы безопасности ОС. Методы и защитные механизмы ОС.  **Безопасность ОС** вкл. в себя меры и механизмы, направленные на защиту данных, ресурсов и юзеров от несанкц. доступа, атак и повреждений. Осн. цели безоп-ти ОС: конфиденциальность, целостность, доступность.  **Классификация угроз:**  1. По источнику: внешние угрозы, внутренние угрозы.  2. По типу атаки: на конфиденциальность, на целостность, на доступность (DDoS-атаки, блокировка доступа к ресурсам).  3. По способу реализации: физ. угрозы, логические угрозы.  **Zero Trust** — концепция безоп-ти, согласно кот. ни 1 устройство или юзер не считаются надежными по умолчанию, независимо от того, наход. ли они внутри или вне сети. Основные принципы: 1) проверка всех юзеров и устройств; 2) минимизация прав доступа; 3) мониторинг и анализ.  **Аутентификация** - процесс проверки личности юзера. Методы аут-ии могут вкл.: пароли, биометр. данные, 1-разовые коды.  **Авторизация** - процесс определения, какие действия и ресурсы разрешены аутентифицированному юзеру. Это может включать: ролевую модель доступа (RBAC), модель управления доступом на основе атрибутов (ABAC).  **Криптография** исп. для защиты данных от несанкц. доступа и обеспечения их целостности. Основные элементы:  Шифрование: преобразование данных в неразборчивый формат для защиты конфиденциальности. Может быть симметричным (один ключ) или асимметричным (пара ключей).  Цифровые подписи: исп. для подтверждения подлинности и целостности данных.  Хэширование: преобр-ие данных в фиксированный размер для обеспечения целостности (например, SHA-256).  **Права доступа** опред., какие действия могут выполнять юзеры и группы с файлами и ресурсами системы. Основные аспекты:  Типы прав: чтение, запись, выполнение.  Управление правами: использование команд (например, chmod, chown) для задания и изменения прав доступа.  Модели управления доступом:  DAC (Discretionary Access Control): права опр-ся владельцем реса.  MAC (Mandatory Access Control): права контролируются политиками безоп-ти и не могут быть изменены владельцем. | 32. Механизмы безопасности в ОС семейства Windows.  **Идент-р безоп-ти (SID)** — уник. значение пер-ной длины, исп-мое для идентификации доверенного лица. Каждая учет. запись имеет уникальный идентификатор безопасности, выданный центром сертификации. Каждый раз, когда юзер входит в систему, система получает идентификатор безопасности для этого юзера из БД и помещает его в маркер доступа для этого юзера.  **Маркер доступа** - объект, сод-ий инфу о контексте безоп-ти процесса или потока. Он вкл. удостоверение и привилегии учет. записи юзера, связанной с процессом или потоком. После успешной аутентификации юзера при входе в систему, система создает маркер доступа. Каждый процесс, выполняемый от имени данного юзера, обладает копией этого маркера доступа.  **Списки управления доступом (ACL)** сост. из записей управления доступом (ACE). Каждый ACE в ACL идентифицирует доверенного лица и указывает права доступа, разрешенные, запрещенные или регистрируемые для этого доверенного лица.  Дескриптор безопасности для защищаемого объекта может содержать два типа списков управления доступом: DACL и SACL  Список управления доступом (DACL) опред. доверенные лица, которым разрешен или запрещен доступ к защищ-му объекту.  Системный список управления доступом (SACL) позволяет регистрировать попытки доступа к объекту, создавая записи аудита в журнале событий безопасности.  **Active Directory (AD)** доменная служба явл. службой каталогов, предоставляющей иерархическую структуру для хранения и предоставления сведений об объектах в сети. В контексте AD DS (Active Directory Domain Services), эта служба хранит информацию о пользователях, группах, серверах и других ресурсах в сети. Администраторы и авторизованные пользователи могут получать доступ к этим данным для управления и использования в сети  Active Directory также включает следующие компоненты: схема, глоб. каталог, механизм запроса и индекса, служба репликации.  Active Directory имеет два типа групп: безопасности и рассылки.  Каждая группа имеет область, опред-щий степень применения группы в дереве домена или лесу. Область группы определяет, где можно предоставить разрешения сети для группы.  Active Directory определяет следующие три области для групп: универсальная, глобальная и локальная в домене. |
| 33. Механизмы безопасности в ОС семейства Linux.  **Механизмы безопасности в Linux:** 1. Упр-е юзерами и группами;2. Права доступа к файлам;3. Механизм контроля доступа DAC и MAC;4. Аудит и журналир-е;5. Шифрование;6. Защита памяти.  **Типы файлов в ОС Linux:**  В ОС GNU/Linux() существуют следующие **типы файлов**:  • обычные файлы (-) все файлы с данными  • каталог (d) тип файла, данными которого является список имен других файлов и каталогов, вложенных в данный каталог  • символьные ссылки (l) файл, в данных кот. содержится адрес другого файла по его имени (а не индексному дескриптору).  • блочные устройства (b)  • символьные устройства ® файлы устройств предназначены для обращения к аппаратному обеспечению компа  • сокеты (s)  • каналы (p)  **Владельцы файлов:**  Владелец (User): обычно это юзер, создавший файл, кот. имеет полный контроль над ним.  Группа (Group): Группа пользователей, которой тоже могут быть предоставлены определенные права доступа.  Другие (Others): Все остальные пользователи в системе.  **Управление правами доступа в файловой системе:**  Права доступа делятся на три типа:  Чтение ®: позволяет просматривать содержимое файла.  Запись (w): позволяет изменять содержимое файла.  Исполнение (x): позволяет выполнять файл как программу.  Права можно устанавливать и изменять с помощью команд:  chmod: Изменение прав доступа.  Chown: Изменение владельца файла.  Chgrp: Изменение группы файла.  **Атрибуты файлов:**  Стандартные атрибуты: содержат информацию о владельце, группе, времени создания и последнего изменения.  Доп. атрибуты: вкл. такие пар-ры, как immutable (недоступный для изменения) или append-only (только добавление данных).  **Управление свойствами файлов:**  Команда lsattr: отображает атрибуты файлов.  Команда chattr: изменяет атрибуты файлов, что может быть полезно для обеспечения безопасности и защиты данных. |  |  |  |