|  |
| --- |
| **Изобразите схему взаимодействия файловой системы и опишите назначение уровней** Файл – поименованная целостная совокупность данных на внешнем носителе, на который наложена структура. Файловая система – часть ОС, обеспечивающая выполнение операций над файлами. **Уровень идентификации**: • по символьному имени файла определяется его уникальное имя (если в ОС один файл может иметь несколько символьных имён); • по уникальному имени определяются атрибуты файла; • сравнение полномочий пользователя или процесса с правами доступа к файлу.  **Логический уровень**: • определяются координаты логической записи в файле; (алгоритм зависит от модели логической организации).**Физический уровень**: • определяется номер физического блока, который содержит требуемую логическую запись. |
| **Постройте схему сегметно-страничной структуризации и схему механизма преобразования виртуального адреса в физический адрес**  Для получения такой **структуризации** выполняются следующие действия: **1.** исходное  пространство структурируют фиксированными страницами; **2.** сегмент  рассматривается как непрерывная последовательность номеров страниц  (размер сегмента — количество страниц); **3.** каждому сегменту  присваивается уникальный номер S; **4.** В пределах каждого сегмента  осуществляют перенумерацию страниц, по возрастанию начиная с 0; В итоге,  сегменту назначается базовый адрес As0. **В итоге** адрес указывается с  помощью четырёх координат (S, As0, R’, R)R – номер страницы в пределах сегмента, R’ –  смещение в пределах страницы. Базовый адрес страницы АБ=Аs0+R\*L  Если размер страницы кратен 2, то размер смещения к AБ  присоединяется операцией конкатенации. **Механизм**  **преобразования виртуального адреса в физический** - для  каждого сегмента создаётся своя таблица страниц. Для каждого  процесса — своя таблица сегментов. При этом адрес таблицы  сегментов процесса загружается в специальный регистр  процессора, когда процесс переходит в активное состояние. |
| **Изобразите схему системы очередей планирования и опишите**  **назначение очередей**  •При создании задачи с помощью функции f1 или f2 она  попадает в конец очереди 1 (RunList,  содержит готовые к выполнению задачи).  Планировщик выбирает первую задачу из очереди  1 и запускает её на выполнение при условии, что  очередь 4 пуста. • Если задаче  требуются  ещё итерации, то    она помещается в конец одной из  очередей в зависимости от того,  какой вызов управления был использован:  - в конец очереди 1 при вызове f3,  - в конец очереди 2 при вызове f4 или f5,  - в конец очереди 3 при вызове f6. • задачи находятся в очереди  2, после завершения попадают в очередь 1. • задачи  из очереди 3 допускаются только в том случае, если  очередь 4 пустая. Обычно в 3 очередь попадают нити, вы  полняющие фоновую работу. • Очередь 4 — самая приоритетная. Однако планировщик разрешает выполняться подряд только определённое количество задач из очереди 4, а затем запускает 1. Очередь 4 была введена для повышения производительности Novell Load Module -приложений. • Механизм организации многозадачной работы сочетается со средствами синхронизации (семафорами и сигналами). |
| **Изобразите схему кэширования диска и опишите данный процесс**  Во многих ФС при работе с внешними устройствами используется подсистема буферизации, которая работает по принципу КЭШ-памяти.  Запрос к внешнему устройству,  в котором адресация осуществляется блоками,  может быть перехвачен от систем  буферизации. Такая система представляет собой  буферный пул – совокупность однородных распределенных блоков ОП  одинаковой длины. Каждый буфер пула имеет размер, равный 1ому блоку. • Механизм работы: при поступлении запроса на чтение некоторого блока подсистемой буфер просматривает сначала свой буферный пул: - в случае обнаружения нужного блока копирует его в буфер запрашиваемого процесса (без обращения к устройству); - если блок не обнаружен, то он считывает и одновременно передает процессу и записывает его в буфер подсистемы. • Возможно, управление чтением нескольких блоков. При отсутствии свободного буфера на диск вытесняется самая редко используемая информация. Например, в Windows размер кэша может меняться в зависимости от текущей ситуации. •Достоинства: фрагментация меньше, срок службы диска увеличивается, скорость увеличивается. |
| **Приведите примеры управления КЭШ-ем** Кэш — промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Доступ в кэш осуществляется быстрее, чем в более медленную память, однако его объём существенно ограничен.  • Работа кэша: Прежде чем считать некоторый блок система сначала обращается в кеш, а если там блока данных нет, то система запрашивает его с медленной памяти, копируя в КЭШ. Самая редко используемая информация вытесняется. • Применение разных типов кеширования: - **ленивый кэш** – это самый простой вид кеширования, но его нужно использовать осторожно, так данные в нем могут устареть. Подходит для данных, которые редко меняются. Ленивый кеш имеет время устаревания, по истечению которого сбрасывается. Достаточно быстр. - **синхронизированный кэш** – самый полезный, так как отдает свежие данные и позволяет реализовать многоуровневый кеш. Например, он встроен в протокол HTTP: Сервер отдает метку изменения, а клиент кеширует у себя результат и передает эту метку. Сервер может дать ответ, что состояние не изменилось и можно использовать кэшированный на клиенте объект. Этот тип кеширования накладывает расходы на общение между системами. Поэтому часто дополняется другими типами кеширования, чтобы ускорить работу. **- кэш сквозной записи**. По методу сквозной записи обычно обновляется слово, хранящееся в основной памяти. Если в кэш-памяти существует копия этого слова, то она также обновляется. Если же в кэш-памяти отсутствует копия этого слова, то либо из основной памяти в кэш-память пересылается строка, содержащая это слово (метод WTWA — сквозная запись с распределением), либо этого не допускается (метод WTNWA — сквозная запись без распределения). |
| **Опишите способы логической организации файловой системы**  Иерархия каталогов может быть деревом или сетью. Каталоги образуют дерево, если файлу разрешено входить только в один каталог, и сеть, если файл может входить сразу в несколько каталогов. В Windows каталоги образуют древовидную структуру (ярлыки не в счет), а в UNIX - сетевую.  Как и любой другой файл, каталог имеет символьное имя и однозначно идентифицируется составным именем, содержащим цепочку символьных имён всех каталогов, через которые  Дерево Сеть проходит путь от корня до данного каталога. Пользователь работает на уровне логических записей. Логическая запись — это наименьший элемент данных, который доступен пользователю для выполнения различных операций. |
| **Приведите примеры структур файлов**  **Файлы с последовательной структурой**. Одномерный массив составных элементов, называемых записями. Длина логической единицы может быть постоянная или переменная. Доступ к элементам последовательный, т.е. после обработки i-ой записи доступна только i+1.  **Файлы с индексно-последовательной структурой.** Идентификация записи в файле по ключу (то есть добавляется учетная информация). Все записи упорядочиваются по значению ключей, выделяются группы записей, ключи которых расположены в 1 файле и могут располагаться в пределах одной дорожки на диске. Для более быстрого поиска используют специальную структуру, например индекс дорожки. При этом поиск элементов файла сначала проводится в прямом, а затем в последовательном порядке. Недостатком является увеличение обьема файла.  **Связный список блоков**. В начале каждого блока содержится указатель на следующий блок и так далее, блоки расположены в случайных участках памяти. Достоинства: Фрагментация практически отсутствует, файл может меняться во время работы, адрес файла задается номером первого блока. Недостатки: усложняется алгоритм распределения, и увеличивается время доступа.  **Перечень номеров блоков**. Номера блоков, которые занимает файл, просто перечисляются. Достоинства: снима-ются проблемы динамического расширения, почти отсутствует фрагментация. Недостатки: увеличивается время доступа, то есть усложняется алгоритм распределения.  **Связный список индексов.** В связном списке индексов с каждым блоком связывается индекс, расположенный в отдельной области диска. В DOS таблица называется FAT (File Allocation Table). Достоинства: файл — это ссылка на первый блок/индекс, фрагментация практически отсутствует, файл может расширяться, но и устраняются недостатки — для обращения к произвольному месту нужно прочитать индексы. Недостаток: больше трата памяти. |

|  |
| --- |
| **Опишите свойства распределенных файловых систем и основные решаемые вопросы.** В основе лежит модель – клиент-сервер. Клиент – обращается к файлу. Сервер – хранит файл и обеспечивает доступ к нему. **Свойства таких систем:** • сетевая прозрачность (клиенты обращаясь к удаленным файлам пользуются теми же операциями, как для локальных файлов); • прозрачность размещения (имя файла не должно определять его местоположения в сети); • независимость размещения (имя файла не зависит от физического расположения); • мобильность пользователя (доступ к файлам из любого узла сети); • устойчивость к сбоям (продолжение функционирования при неисправности отдельного компонента); • масштабируемость (для случаев увеличения нагрузки, должна существовать возможность постепенного наращивания системы); • мобильность файлов (должна быть возможность перемещения файлов из одного месторасположения в другое). **Решаемые вопросы**: • пространство имен (некоторые РФС обеспечивают однородное пространство имен, другие системы позволяют клиенту создавать свое пространство имен монтированием разделяемых поддеревьев к произвольным каталогам в иерархии файлов); • операции с сохранением и без сохранения состояний (сервер, обеспечивает хранение информации об операциях клиента между запросами, для корректного обслуживания последующих запросов); • семантика разделения (РФС должна определить семантику, применяемую когда несколько клиентов одновременно обращаются к одному файлу); • методы удаленного доступа (каждое действие инициируется клиентом, а сервер представляет собой агента, который выполняет заявки клиента). |
| **Опишите организацию работы процесса на примере современной операционной системы**  •**Процесс** — заявка на потребление системных ресурсов, как логическая единица, он - выполняет операции, и является носителем данных. В процессе две части: выполняемая программа и дескриптор процесса. Дескриптор процесса содержит информацию, которая необходима ядру пока процесс существует. Дескрипторы процессов объединены в список – таблицу процессов (Память для неё отводится динамически в области ядра). На основании информации, в таблице, происходит планирование и синхронизация процессов. Контекст процесса — это информация, о процессе, необходимая в активном состоянии, доступная только ядру через виртуальные адреса, но хранимая вместе с процессом. Контекст содержит информацию необходимую для возобновления выполнения процесса с прерванного места: содержимое регистров процессора, коды ошибок, системные вызовы, информацию об открытых файлах в том числе ввода-вывода и другие данные.  • **В UNIX** для процессов предусмотрены два режима: привилегированный, для программ ядра с возможностью управления ресурсами системы - "система" и обычный, для прикладных программ - "пользователь".  Существует два уровня приоритетов — верхний и нижний. ОС осуществляет динамическое изменение приоритетов (Например, на основе оценки соотношения t\_использованное/t\_прогнозируемое, если процесс захватил много времени, то его приоритет уменьшается, и наоборот). Перераспределение приоритетов происходит примерно раз в секунду. Этот период называют квантом мультиплексирования процессора. Для передачи данных между процессами используются события и специальные файлы, которые наследуются и дочерними процессами. Файл уничтожается, когда завершаются все процессы, имеющие его дескриптор. Количество файлов не превышает максимальное число дескрипторов. |
| **Приведите классификацию процессов для современной операционной системы, опишите их основное назначение и приведите примеры (на примере Linux) Системные процессы -** являются частью ядра и всегда находятся в оперативной памяти. Такие процессы не имеют соответствующих им программ в виде исполняемых файлов и запускаются при инициализации ядра. **Примеры**: shed (диспетчер свопинга), vhand (диспетчер страничной развертки), bdfflush (диспетчер буферного кэша), итд.  **Демоны** – не интерактивные процессы, которые запускаются путем загрузки в память соответствующих им программ. Выполняются в фоновом режиме. Обычно запускаются при инициализации системы, но после инициализации ядра. Демоны не связаны ни с одним из пользовательских сеансов работы. Большую часть времени спят, ожидая запросов от другого процесса. **Например**, демоны обеспечивают системы терминального доступа, системы печати, системы сетевого доступа.  **Прикладные процессы** – все остальные процессы. Как правило порождаются действиями пользователя. Важным пользовательским процессом является интерпретатор, который обеспечивает работу Linux. Записывается после регистрации пользователя в системе, а завершение этого процесса приводит к отключению системы. |
| **Постройте граф существования процесса и опишите условия перехода из одного состояния в другое**  • Порождение — подготавливаются все  условия для исполнения. • Готовность  — все ресурсы предоставлены (кроме  процессора). Здесь процесс ждёт, когда  диспетчер его выберет. • Активное состояние — процесс обладает  всеми ресурсами и использует процессор. • Окончание — процесс  завершается, аварийно или успешно. • Ожидание — процесс может быть  прерван: **1 род прерываний** — процесс сам является причиной прерывания:  - при попытке получить ресурс или отказаться от ресурса; - при порождении,  уничтожении или действии с другим процессом; - алгоритмическое переполнение, попытка обращения к защищённой памяти итд; **2 род прерываний** — процесс не является причиной прерывания: - процесс может быть вытеснен более приоритетным процессом или по окончании кванта времени; - из-за необходимости проведения синхронизации между другими процессами. |
| **Сформулируйте концепции проектирования операционных систем**  **Монолитные ОС**. ОС в виде набора процедур, где каждая может вызывать  любую другую (отсутствуют правила вертикального  управления). Для построения монолитной ОС необходимо  было скомпилировать все необходимые процедуры и  скомпоновать в единый объектный файл. **Примеры:** Unix,  Novel NetWare. **Структурированная монолитная ОС**. При  обращении к системным вызовам параметры помещаются  в строго  отведённом месте,  а затем выполняется  специальная  команда  прерывания - вызов ядра (супервизора). Команда переключает машину  из режима пользования в режим ядра, проверяя параметры вызова,  индексируется таблица ссылок на процедуры и вызывается  соответствующий процесс. В этой модели для каждого системного вызова имеется 1 сервисная процедура. **Многоуровневые иерархическое ОС**. Рисунок окружности, в центре ядро. Уровни: •1 распределение времени процессора, переключение процессов. •2 выполнение функции виртуальной оперативной памяти. •3 управление связью между консолью оператора и процессора. •4 управление устройствами ввода-вывода и буферизация потока. •5 пользовательские программы. •6 процесс системного оператора. Каждый уровень может  взаимодействовать только с соседним. **Микроядерные ОС**. В ядре расположены только  самые основные функции, все остальные компоненты (в тч аппаратные ресурсы) вынесены в  «серверы». Все пользовательские приложения могут получить доступ к функционалу ОС только  через микроядро, которое проверяет право на доступ. Такой подход позволяет  использовать микроядерные ОС в распределённых средах. Однако это замедляет  пересылку сообщений, ускорение её важная задача. |
| **Изобразите структуру современной операционной системы и опишите назначение основных компонентов** Windows NT состоит из двух частей:  1. **Защищённые подсистемы - серверы** - отдельные процессы, к памяти которых имеет  доступ только ядро. Предоставляют ядру пользоват.  и программный интерфейсы, обеспечивают среду  выполнения. Сервер подразумевает, что каждая  функция подсистемы обеспечивает API­интерфейс.  API реализуется на отдельном  сервере для Win32, OS/2, POSIX и др. •Подсистема  Win32 предоставляет прикладным программам API, реализует  графический интерфейс и  управляет вводом/выводом.  Остальные подсистемы  имеют свои API, но использ-  уют для взаимодействия с  пользователем подсистему  Win32. В NT (в отличие от  16­тиразрядной Windows)  для повышения производительности  были перенесены диспетчер окон и драйверы графических устройств в ядро. Это сильно ускорило работу систему. 2. **Ядро (Исполнительная система)** - является законченной ОС низкого уровня. Содержит: системные сервисы и внутренние процедуры. **Компоненты ядра** независимы друг от друга: •Диспетчер объектов - создаёт, поддерживает и уничтожает объекты. •Справочный монитор защиты – защищает ресурсы ОС и объекты и ведёт аудит. •Диспетчер процессов – создаёт, завершает и выводит информацию о процессах. •Средства локального вызова процедур (LPC) – передаёт сообщения между клиентскими и серверными процессами. •Диспетчер виртуальной памяти – выделяет и управляет виртуальной памятью, осуществляет подкачку страниц, выделяет процессам адресное пространство. •Ядро – реагирует на прерывания, направляет потоки на выполнение, осуществляет межпроцессорную синхронизацию. •Диспетчер ввода/вывода – реализует средства ввода/вывода независимо от типа устройства. |

|  |
| --- |
| **Приведите схему современной файловой системы**  **и опишите назначение основных блоков** Современная  ОС может работать сразу с несколькими ФС, имеет многоуровневую  структуру. Приложение обращается только через переключатели (в  Windows это IFS). Переключатель преобразует запросы в формат  конкретной ФС. Каждый драйвер ФС поддерживает определенную  организацию ФС, и позволяет сразу нескольким приложениям выполнять  операции над файлом. Подсистема ввода/вывода отвечает за загрузку,  инициализацию, управление драйверами нижнего уровня. Каждый  уровень драйверов устройств представляет собой определенный  тип: драйвер жестких дисков, драйверы, перехвата запроса к блочным  устройствам, драйверы портов, управляющие конкретными адаптерами. |
| **Обоснуйте необходимость синхронизации процессов на примере гонок**  Существует два типа конфликтов: при взаимодействии процессов между собой и  конфликты с общими ресурсами.  Многие процессы находятся в  зависимости от других  процессов. Например,  процесс печати файлов. Он  печатает по очереди все  файлы, имена которых другие процессы  записывают по порядку в общедоступный файл  заказов. Имеется особая переменная next, она  доступна всем процессам и содержит номер  первой свободной  позиции файла  заказов. Если два  процесса попытаются  записать что-то в общую  переменную next, то процесс  немного «опоздавший» не сможет отправить свои данные или потеряет их. То есть Next – критическая секция. На рисунке видно, что в итоге пострадает процесс S. Такую ситуацию, называют «эффект гонок»: - критическая секция – часть программы, в которой осуществляется доступ к разделенным данным; - взаимные исключения – обеспечивает, чтобы в каждый момент в критической секции находится не более одного процесса. Решить это можно использованием блокирующей переменной (семафора) для каждого ресурса. Если она ноль, то ресурс занят, единица – свободен. На время изменения блокирующей переменной все прерывания запрещаются, чтобы к ней мог иметь доступ только один процесс. |
| **Опишите способы разрешения конфликтов** Взаимоблокировка: блокировка конкурирующих процессов друг другом при конкуренции за ресурсы. все процессы не могут продолжать работу – дедлоки, клинчи и тупики. Взаимоблокировка — это тупиковая ситуация, характеризующаяся тем, что группа процессов ожидает события, которое может вызвать только другой процесс из этой же группы. •Способы восстановления: 1) через откат в ближайшую точку, без конфликта; 2) при помощи принудительной выгрузки ресурсов; 3) путем уничтожения процессов. •Проблемы синхронизации и взаимоисключения должны быть решены при помощи управления событиями и памятью. Синхронизация подразумевает сигнализацию между процессами по определенным протоколам. Такая операция не зависит от времени и её принято называть событием (объект синхронизации, используемый для информирования потоков о том, что событие произошло). •В ОС допускается возникновение нескольких событий. Чтобы их отличать, каждому событию присваивается идентификатор, который называется флагом события. |
| **Изобразите схему модели организации сетевой файловой системы и опишите принцип работы** •Клиенты сетевой ФС — это программы, которые работают на  многочисленных компьютерах, подключенных к сети, они обслуживают запросы  приложений на доступ к файлам, хранящимся на удаленном компьютере. •Клиент  сетевой ФС передает по сети запросы серверу сетевой ФС. Сервер, получив запрос,  может выполнить его либо самостоятельно, либо, передать запрос локальной  файловой системе. После сервер  передает ответ по сети клиенту, а  тот, - приложению, обратившемуся  с запросом. •Приложения  обращаются к клиенту сетевой ФС,  используя определенный  программный интерфейс, который  в данном случае является  интерфейсом сетевой файловой  системы. Этот интерфейс стараются  сделать как можно более похожим  на интерфейс локальной файловой  системы, чтобы соблюсти принцип  прозрачности. При полном  совпадении программных интерфейсов приложение может обращаться к локальным и удаленным файлам и каталогам с помощью одних и тех же системных вызовов, совершенно не принимая во внимание места хранения данных. •Клиент и сервер сетевой файловой системы взаимодействуют друг с другом по сети по определенному протоколу. В его функции входит ретрансляция серверу запросов, принятых клиентом от приложений. |