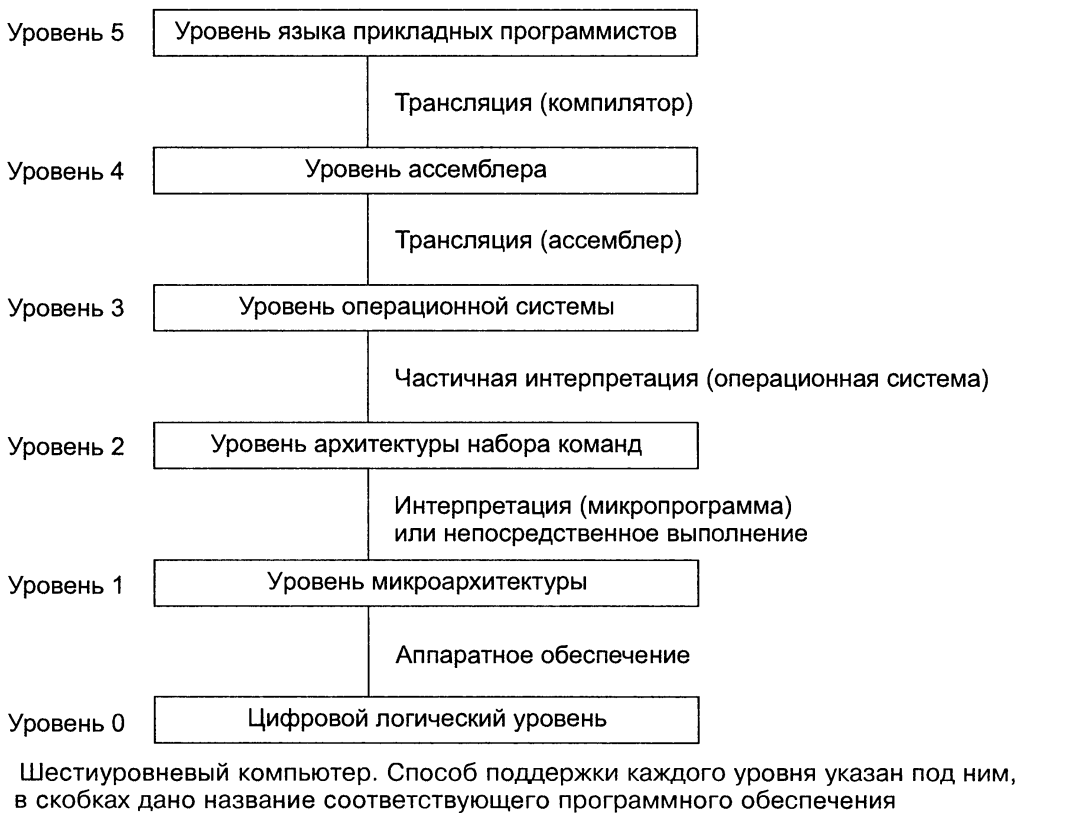
1. **Большинство современных компьютеров состоит из двух и более уровней;**



Под ос понимается то программное обеспечение, которое запускается в *режиме ядра* (*супервизора)*.

**Трансляция**. Замена каждой команды языка Я1 на эквивалентный набор команд в языке Я0. В этом случае компьютер выполняет новую программу, написанную на языке Я0, вместо программы, написанной на языке Я1.

И**нтерпретация -** написание программы на языке Я0, которая берет программы, написанные на языке Я1, в качестве входных данных, рассматривает каждую команду по очереди и сразу выполняет эквивалентный набор команд языка Я0. Различие лишь в том, что при трансляции вся программа **Я1** переделывается в программу **Я0**, программа **Я1** отбрасывается, а новая программа на **Я0** загружается в память компьютера и затем выполняется.

Большинство современных компьютеров **состоит из двух и более уровней**.

1. **Цифровой логический уровень или аппаратное обеспечение машины**. Электронные схемы выполняют программы. Объекты— вентили. Состоят из аналоговых компонентов, таких как транзисторы. Вентиль вычисляет простые функции такие как И или ИЛИ. Каждый вентиль формируется из нескольких транзисторов. Несколько вентилей формируют 1 бит памяти, который может содержать 0 или 1. Биты памяти, объединенные в группы, например, по 16, 32 или 64, формируют **регистры**. Каждый регистр может содержать одно двоичное число до определенного предела.
2. ***Микроархитектурный* уровень**. На этом уровне можно видеть совокупности 8 или 32 регистров, которые формируют **локальную память и АЛУ (арифметико-логическое устройство).** Регистры вместе с АЛУ формируют тракт данных, по которому поступают данные. Основная операция: выбирается один или два регистра, АЛУ производит над ними какую-то операцию, а результат помещается в один из этих регистров. Работа тракта контролируется микропрограммой или аппаратным обеспечением.
3. **Уровень *архитектуры команд***. Команды используют регистры и другие возможности аппаратуры. Команды формируют уровень ISA (Instruction Set Architecture), называемый машинным языком. Команды служат преимущественно для перемещения данных по компьютеру, выполнения арифметических операций и сравнения величин.
4. **Гибридный**. Набор новых команд, другая организация памяти, способность выполнять две и более программы одновременно и некоторые другие. Новые средства выполняются интерпретатором, который работает на втором уровне. Этот интерпретатор был когда-то назван *операционной системой. О*дна часть команд интерпретируется операционной системой, а другая – микропрограммой.
5. **Ассемблер**. Эти программы сначала транслируются на язык уровня 1, 2 или 3, а затем интерпретируются машиной.
6. **Уровни прикладных программистов** (C, C++, C#, VBA и др.).
7. **Принципиально важными универсальными подходами к разработке архитектуры ОС являются;**
8. модульная организация.
9. функциональная избыточность; выполнение звеном системы не свойственной ему функции при отказе другого звена
10. функциональная избирательность (модули, составляющие ядро.);
11. параметрическая универсальность;
12. концепция многоуровневой иерархической системы (многослойной структурой);
13. разделение модулей по функциям: ядро – модули, выполняющие **основные** функции ОС, и модули, выполняющие **вспомогательные** функции ОС;
14. разделение модулей по размещению в памяти: **резидентные** (постоянно в ОП), и **транзитные** (загружаемые на время);
15. реализация двух режимов работы: привилегированного режима (режима ядра) и пользовательского режима;
16. ограничение функций ядра до минимального количества.
17. **К ОС предъявляется ряд важных эксплуатационных требований;**
18. *Эффективность*.
19. ***Надежность и отказоустойчивость***. Количество исправленных сбоев к общему количеству сбоев
20. ***Безопасность*****авторизации** *(защищенность)*. **Аутентификации** – определения легальности пользователей,– предоставления легальным пользователям установленных им прав доступа к ресурсам.
21. ***Предсказуемость***. Пользователь должен иметь представление, когда ему ожидать выдачи результатов.
22. **Расширяемость**. Когда дополнения и изменения могут вноситься без нарушения целостности системы.
23. ***Переносимость***. Перенос с процессора одного типа на процессор другого типа.
24. ***Совместимость***. возможность выполнить свои приложения в новой операционной системе.
25. ***Удобство***. Наличие у ОС графического пользовательского интерфейса и всевозможных программ,
26. ***Масштабируемость***. Различное числом процессоров, возрастание производительности при увеличении числа процессоров

**13. Все многообразие существующих ОС можно классифицировать по множеству различных признаков;**

1. По назначению ОС делятся на **универсальные** и **специализированные**. Специализированные ОС работают с фиксированным набором программ (функциональных задач). Универсальные ОС рассчитаны на решение любых задач пользователей
2. По способу загрузки: ***загружаемые*** *ОС* (большинство) и *системы,* ***постоянно*** *находящиеся в памяти* вычислительной системы.
3. По особенностям *алгоритмов управления ресурсами*
   1. Поддержка **многозадачности** (однопрограммные, многопрограммные). Однопрограммные ОС предоставляют пользователю виртуальную машину, Многозадачные ОС, кроме того, управляют разделением совместно используемых ресурсов (процессор, память, файлы и т.д)
   2. Поддержка **многопользовательского** режима. По числу одновременно работающих пользователей ОС делятся
   3. Виды **многопрограммной** работы. Способ распределения времени между процессами (или потоками). Процесс либо сам отдает управление системе, либо система решает.
   4. Многопроцессорная обработка.
4. *По* **области использования** *и* **форме эксплуатации**.
   1. системы пакетной обработки (решения задач в основном вычислительного характера, не требующих быстрого получения результатов)
   2. системы разделения времени (имеет терминал и может вести диалог со своей программой.)
   3. системы реального времени (для управления техническими объектами (станок, спутник)
5. *По* **аппаратной платформе** (типу вычислительной техники).
   1. Операционные системы для смарт-карт.
   2. Встроенные операционные системы. Управляют карманными компьютерами
   3. Операционные системы для персональных компьютеров, например.
   4. Операционные системы мини-ЭВМ
   5. Операционные системы мэйнфреймов (больших машин)
   6. Серверные операционные системы
   7. Кластерные операционные системы. Совокупность нескольких вычислительных систем

**11. Назначение операционных систем;**

1. **Интерфейс** между приложениями и пользователями и аппаратурой компьютера. Список основных сервисов (*Разработка программ, Исполнение программ, Доступ к устройствам ввода-вывода, Контролируемый доступ к файлам, Системный доступ, Обнаружение ошибок и их обработка, Учет использования ресурсов)*

*2.* **Организация эффективного использования** ресурсов компьютера. ОС является диспетчером ресурсов компьютера (***планирование ресурса*** – определение, какому процессу, когда и в каком качестве следует выделить данный ресурс; ***удовлетворение запросов на ресурсы*** – выделение ресурса процессам; ***отслеживание состояния и учет использования ресурса*** – поддержание оперативной информации о занятости ресурса и распределенной его доли; ***разрешение конфликтов между процессами***, претендующими на один и тот же ресурс)

3. **Облегчение процессов** эксплуатации аппаратных и программных средств вычислительной системы (наборы служебных программ, обеспечивающие резервное копирование, архивацию данных, проверку, очистку и дефрагментацию дисковых устройств и др.)

4. **Возможность** развития.

1. **В современных операционных системах различают следующие виды ядер**

Ядро операционной системы (Kernel) - часть операционной системы: постоянно находящаяся в оперативной памяти, управляющая всей операционной системой, содержащая: драйверы устройств, подпрограммы управления памятью, планировщик заданий, реализующая системные вызовы и т.п. Одной из самых важных функций, реализованных в ядре, является обработка прерываний. **Ядро — командный интерпретатор с программного языка на язык машинных кодов.**

**Наноядро** (НЯ). Сигнал от ПО или аппаратного обеспечения, сообщающий процессору что необходимо прервать текущий код, и выполнить более приоритетный. Выполняет обработку аппаратных прерываний.

**Микроядро** (МЯ). Небольшой набор низкоуровневых примитивов\сервисов\модулей\механизмов для работы со страницами памяти, драйверами, процессами и потоками по выделению процессорного времени и т.д.. В микроядерной ОС можно, не прерывая ее работы, загружать и выгружать новые драйверы, файловые системы и т. д.

**Экзоядро** (ЭЯ) предоставляет набор сервисов для взаимодействия между приложениями, а также необходимый минимум функций, связанных с защитой: выделение и высвобождение ресурсов, контроль прав доступа и т. д. Абстракции для физических ресурсов выносятся в LIBOS – библиотеку пользовательского уровня.

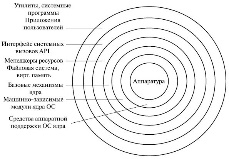
**Монолитное ядро** (МнЯ) предоставляет широкий набор абстракций (один из способов сокрытия деталей реализации определенного набора функций) оборудования. Все части ядра работают в одном адресном пространстве. МнЯ требуют перекомпиляции при изменении состава оборудования. Компоненты операционной системы являются составными частями одной программы. МнЯ работает как один большой процесс.

**Модульное** **ядро**. Модульные ядра не требуют полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера. Вместо этого они предоставляют тот или иной механизм подгрузки модулей, поддерживающих то или иное аппаратное обеспечение (например, драйверов). Подгрузка модулей может быть, как динамической, так и статической (при перезагрузке ОС после переконфигурирования системы). Они предоставляют программный интерфейс (API) для связывания модулей с ядром, для обеспечения динамической подгрузки и выгрузки модулей.

**Гибридное ядро** (ГЯ) – модифицированные микроядра, позволяющие для ускорения работы запускать "несущественные" части в пространстве ядра. Микроядро обеспечивает управление виртуальной памятью и работу низкоуровневых драйверов. Все остальные функции, в том числе взаимодействие с прикладными программами, осуществляются монолитным ядром

**12. Многослойная структура ядра ОС;**

Каждый слой обслуживает вышележащий слой, выполняя для него некоторый набор функций, которые образуют межслойный *интерфейс*. Модули каждого слоя можно изменять без необходимости изменений в других слоях (но не меняя межслойных интерфейсов!).



1. **Средства аппаратной поддержки** *ОС*. Значительная часть функций ОС может выполняться аппаратными средствами
2. **Машинно-зависимые модули** *ОС*. Модули, в которых отражается специфика аппаратной платформы компьютера. Назначение этого слоя – "экранирование" вышележащих слоев ОС от особенностей аппаратуры (слой *HAL* (Hardware *Abstraction Layer*), уровень аппаратных абстракций).
3. **Базовые механизмы ядра**. Программное ***переключение*** ***контекстов процессов***, диспетчерскую прерываний, перемещение страниц между основной памятью и диском и т.п. Модули обрабатывают решения, принятые модулями вышележащих уровней.
4. **Менеджеры ресурсов**. Задачи по управлению ресурсами.
5. **Интерфейс системных вызовов**. Это верхний слой ядра ОС, взаимодействующий с приложениями и *системными утилитами*, он образует **прикладной программный интерфейс ОС**. Функции API, обслуживающие системные вызовы, предоставляют **доступ к ресурсам системы в удобной компактной форме,** без указания деталей их физического расположения.

**Менеджеры** *ресурсов*, вынесенные в пользовательский режим, называются **серверами** ОС, т.е. модулями, основным назначением которых является обслуживание запросов локальных приложений и других модулей ОС.

1. **Концепция процессов и потоков. Задание, процессы, потоки (нити), волокна. Методы взаимоисключений;**

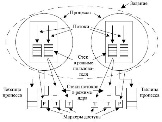
*Процесс* – абстрактное понятие, описывающее работу программы.

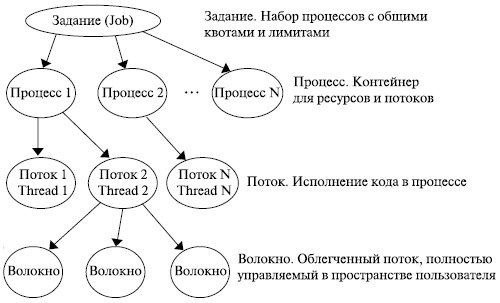
Создаются и поддерживаются таблицы, содержащие информацию по состоянию каждого процесса и потока.

**Задание** представляет собой набор из одного или нескольких процессов, управляемых как единое целое.

**Поток** – набор команд.

**Процессы** – несколько потоков. Рассматриваются ОС как заявки для ресурсов (кроме процессорного времени. Это *ресурс* распределяется ОС между потоками.) Каждый процесс начинается с одного потока, но новые потоки могут создаваться процессом.





Когда поток обращается к системному вызову, он переключается в режим ядра. После завершения системного вызова *поток* продолжает выполняться в режиме пользователя. У каждого потока есть два стека, один используется в режиме ядра, другой – в режиме пользователя. Помимо состояния идентификатора и двух стеков, у каждого потока есть ***контекст*** (в котором сохраняются его регистры, когда он не работает), приватная **область локальных переменных**, а также может быть собственный **маркер доступа** (*информация* о защите). Процесс завершается, когда прекратит существование последний *активный* *поток*.

***Волокна*** (*Fiber*), подобные потокам, но планируемые в пространстве пользователя создавшей их программой. У каждого потока может быть несколько волокон, с той разницей, что, когда волокно логически блокируется, оно помещается в *очередь* блокированных волокон, после чего для работы выбирается другое волокно. При этом ОС "не знает" о смене волокон, так как все тот же *поток* продолжает работу.

**Методы взаимоисключения**: для того чтобы процессы могли избежать проблемы взаимоисключений, необходимо и достаточно чтобы выполнялись 5 условий:

1) Задача должна быть решена программным способом

2) Не должно существовать никаких предположений об относительных скоростях выполняющихся процессов

3)Если процесс находится в критической секции (участке), то не существует других процессов, которые исполняют код в своих соответствующих критических участках

4)Процессы которые находятся вне своих критических участков не должны препятствовать входу других процессов в свой критический участок

5)Не должно быть бесконечного ожидания для входа в критические участки

Критический участок - участок кода, который захватывает неразделяемый ресурс и выполняет с ним какие-то действия

**16.Создание процессов и потоков. Модели процессов и потоков. Системные вызовы**

**Создать процесс** – создать описатель процесса: сведения о процессе, необходимые операционной системе для управления им. Включает загрузку кодов и данных программы с диска в оперативную память.

Образ процесса – множество, в которое входят программа, данные, стеки и атрибуты процесса.

Три основных события, приводящие к созданию процессов:

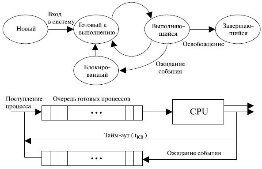
Загрузка системы

Работающий процесс подает системный вызов на создание процесса

Запрос пользователя на создание процесса

Местонахождение образа процесса зависит от используемой схемы управления памятью.

Процессу присваивается числовой *идентификатор*, который может быть просто индексом в первичной таблице процессов, чтобы найти *по* идентификатору процесса соответствующие ему таблицы.



Модель потоков.

При создании потоков ОС генерирует *описатель потока* (*идентификатор* потока, данные о правах доступа и приоритете, о состоянии потока и другую информацию). Описатель потока можно разделить на две части: *атрибуты* блока управления и *контекст* потока.

*Ядро* о потоках ничего не знает и управляет обычными однопоточными процессами.

Если управление потоками происходит в пространстве пользователя, каждому процессу необходима собственная *таблица* потоков. Она отслеживает такие характеристики потоков, как *счетчик* команд, *указатель* вершины стека, регистры состояния и т. п. Когда *поток* переходит в состояние готовности или блокировки, вся *информация*, необходимая для повторного запуска, хранится в таблице потоков.

*По* умолчанию *приложение* в начале своей работы состоит из одного потока и его выполнение начинается как выполнение этого потока. Такое *приложение* вместе с составляющим его потоком размещается в одном процессе, который управляется ядром. Выполняющийся *поток* может породить новый *поток*, который будет выполняться в пределах того же процесса. Новый *поток* создается с помощью вызова специальной подпрограммы из библиотеки, предназначенной для работы с потоками. Передача управления этой программе происходит в результате вызова соответствующей процедуры.

*Системный вызов* позволяет приложению обратиться к операционной системе с просьбой выполнить то или иное действие, оформленное как процедура кодового сегмента ОС.

1. Управление процессами и потоками. Взаимоблокировки;

При управлении процессами ОС использует **два** основных типа информационных структур: **блок управления процессом** (дескрипторы, объединяются в таблицу (в таблице находятся три категории: информация по идентификации, по состоянию, используемая при управлении) и **контекст процесса** (содержит информацию, позволяющую системе останавливать и возобновлять выполнение процесса с прерванного места).

***Информация*** ***по* состоянию и управлению** процессом включает следующие основные данные:

* состояние процесса, определяющее готовность процесса к выполнению (выполняющийся, готовый к выполнению, ожидающий какого-либо события, приостановленный);
* данные о приоритете;
* информация о событиях – идентификация события, наступление которого позволит продолжить выполнение процесса;
* указатели, позволяющие определить расположение образа процесса в ОП и на диске;
* указатели на другие процессы (в частности, находящиеся в очереди на выполнение);
* флаги, сигналы и сообщения, имеющие отношение к обмену информацией между двумя независимыми процессами;
* Права доступа;
* указатели на ресурсы, которыми управляет процесс (например, перечень открытых файлов);
* Журнал истории использования;
* Планирование (время ожидания, кол-во выполненных операций ввода-вывода).

Основные функции этой подсистемы

1. создание процессов и потоков. Все процессы имеют родителя. Процесс необходимо зарегистрировать в таблице процессов.
2. обеспечение процессов и потоков необходимыми ресурсами
3. *изоляция процессов* (для того чтобы процессы не могли вмешаться в *распределение ресурсов*, а также не могли повредить коды и данные друг друга)
4. планирование выполнения процессов и потоков (пользователи посылают задание, а ОС создает новый процесс и запускает следующее задание из очереди, когда освобождаются необходимые ресурсы.);
5. диспетчеризация потоков (прием заявок на выделение ресурсов и последующее выполнение.)
6. организация межпроцессного взаимодействия (Один процесс не должен обращаться в область другого процесса)
7. синхронизация процессов и потоков (степень осведомленности)
8. завершение и уничтожение процессов и потоков (завершение задания, выгрузка процесса)

Переход от выполнения одного потока к другому осуществляется в результате *планирования* и *диспетчеризации*.

**Планирование** - определение момента, когда необходимо прервать выполнение текущего потока и начать выполнение другого.

**Диспетчеризация** – переключение процессора с одного потока на другой в результате планирования. (сохранение контекста текущего потока; загрузка контекста потока; запуск нового потока на выполнение.)

**Взаимоблокировка** - не отдают ресурсы, в ожидании другого ресурса

**Голодание** - невозможность использовать нужный процессу ресурсу

**Пакетное задание** - поддержка ОС, дефрагментация оперативной памяти, обработка графики

**Служба** - приложения, автоматически (если настроено) запускаемые системой при запуске Windows и выполняющиеся вне зависимости от статуса пользователя.

Для того чтобы произошла *взаимоблокировка*, должны выполняться все эти четыре условия. Если хотя бы одно отсутствует, *тупиковая ситуация* невозможна.

1. Условие взаимного исключения. Каждый ресурс в данный момент или отдан ровно одному процессу, или доступен.
2. Условие удерживания и ожидания. Процессы, в данный момент удерживающие полученные ранее ресурсы, могут запрашивать новые ресурсы.
3. Условие отсутствия принудительной выгрузки ресурсов. У процесса нельзя забрать принудительно ранее полученные ресурсы. Процесс, владеющий ими, должен сам освободить ресурсы.
4. Условие циклического ожидания. Должна существовать круговая последовательность из двух и более процессов, каждый из которых ждет доступа к ресурсу, удерживаемому следующим членом последовательности.

При столкновении с взаимоблокировками используются четыре стратегии (1. Пренебрежение проблемой в целом. 2. Обнаружение и восстановление. Позволить взаимоблокировке произойти, обнаружить ее и предпринять какие-либо действия. 3. Избегать тупиковых ситуаций с помощью аккуратного распределения ресурсов. 4. Предотвращение с помощью структурного опровержения одного из четырех условий)

**17.Взаимодействие и синхронизация процессов и потоков;**

Способы взаимодействия процессов (потоков) можно классифицировать по степени осведомленности одного процесса о существовании другого.

1. Процессы *не осведомлены* о наличии друг друга (например, процессы разных заданий). ОС должна решать вопросы конкурентного использования ресурсов.
2. Процессы *косвенно осведомлены* о наличии друг друга (например, процессы одного задания). Эти процессы не обязательно должны быть осведомлены о наличии друг друга с точностью до идентификатора процесса, однако они разделяют доступ к некоторому объекту, например, буферу ввода-вывода, файлу или БД. Такие процессы демонстрируют сотрудничество при разделении общего объекта.
3. Процессы *непосредственно осведомлены* о наличии друг друга (например, процессы, работающие последовательно или поочередно в рамках одного задания). Такие процессы способны общаться один с другим с использованием идентификаторов процессов и изначально созданы для совместной работы. Эти процессы также демонстрируют сотрудничество при работе.

Каждый из процессов не подозревает о наличии остальных и не повергается никакому воздействию с их стороны. Каждый процесс не должен изменять состояние любого ресурса, с которым он работает.

Между конкурирующими процессами не происходит никакого обмена информацией.

В случае конкурирующих процессов (потоков) возможно возникновение трех проблем. Первая из них – необходимость взаимных исключений (mutual exclusion).

1. **Планирование заданий, процессов и потоков;**

Основная **цель** планирования - распределении времени процессора.

Все виды планирования делятся на **долгосрочное** (Решение о добавлении задания (процесса) в пул выполняемых в системе), **среднесрочное** (Решение о добавлении процесса к числу процессов, полностью или частично размещенных в основной памяти), **краткосрочное**/ *dispatcher* (Решение о том, какой из доступных процессов (потоков) будет выполняться процессором) и **планирование ввода-вывода**(Решение о том, какой из запросов процессов (потоков) на операцию ввода-вывода будет выполняться свободным устройством ввода-вывод).

В мультипрограммной системе *поток* (или процесс) может находиться в одном из ***трех основных состояний***:

* выполнение – активное состояние потока. поток обладает всеми необходимыми ресурсами и выполняется процессором;
* ожидание – пассивное состояние потока. поток заблокирован по своим внутренним причинам (ждет осуществления некоторого события, например, освобождения необходимого ресурса);
* готовность – также пассивное состояние потока, но в этом случае поток заблокирован в связи с внешним по отношению к нему обстоятельством (имеет все требуемые ресурсы, готов выполняться, но процессор занят выполнением другого потока).

**Невытесняющие алгоритмы** - потоку позволяется выполняться пока он *по* своей инициативе не отдает управление ОС.

Вытесняющие алгоритмы – решение о переключении процессора с выполнения одного потока на выполнение другого потока принимается операционной системой, а не активной задачей.

Смена активного потока происходит, если: 1) поток завершается и покинул систему; 2) произошла ошибка; 3) поток перешел в состояние ожидания; 4) исчерпан квант времени, отведенный данному потоку.

*Поток*, который исчерпал свой квант, переводится в состояние готовности и ожидает, когда ему будет предоставлен новый *квант процессорного времени*.

***Диспетчер*** реализует найденное решение, т.е. переключает *процессор* с одного процесса (потока) на другой. Он вызывается при наступлении события, которое может приостановить текущий *процессор* или предоставить возможность прекратить выполнение данного процесса (потока) в пользу другого. Примерами этих событий могут быть:

* прерывание таймера;
* прерывание ввода-вывода;
* вызовы операционной системы;
* сигналы.

Среднесрочное планирование является частью системы свопинга. Обычно решение о загрузке процесса в *память* принимается в зависимости от степени многозадачности.

1. **Организация памяти современного компьютера;**

1) Чем меньше время доступа, тем дороже бит

2) Чем выше емкость, тем ниже стоимость бита

3) Чем выше емкость, тем больше время доступа

Закон Паркинсона: Любая программа стремится заполнить всю возможную память

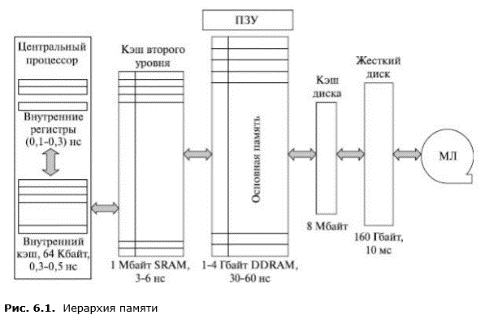
**Виртуализацией оперативной памяти** – временная выгрузка неактивные процессы из ОП на жесткий диск. *О*существляется совокупностью аппаратных и программных средств (схемами процессора и операционной системой).

Эта технология возможна с помощью двух подходов:

1) Свопинг - подкачка данных. Весь образ процесса выгружается из оперативной памяти на ЖД.

2) Виртуальная память. На ЖД выгружаются только определенные объекты (страницы, сегменты и блоки). Если у нас 20 ГБ оперативной памяти, то сколько будет занимать каждый адрес виртуальной памяти? - вопрос на контрольной (для 16-бит 2^32)

Основная и внешняя память организована как линейное адресное пространство, состоящее из последовательности байтов.



снижается стоимость бита;

возрастает емкость;

возрастает время доступа;

снижается частота обращений процессора к памяти

Иерархия запоминающих устройств: ЦП (внутренние регистры, внутренний кэш(64КБ)-Кэш 2 уровня (1 МБ) – ПЗУ (1-4 Гб DDR) – Кэш диска (8 МБ) – Жесткий диск(160ГБ)

снижается стоимость бита - возрастает емкость - возрастает время доступа - снижается частота обращений процессора к памяти.

18.Функции ОС по управлению ОП;

Перечислим функции операционной системы *по* управлению памятью в этом случае.

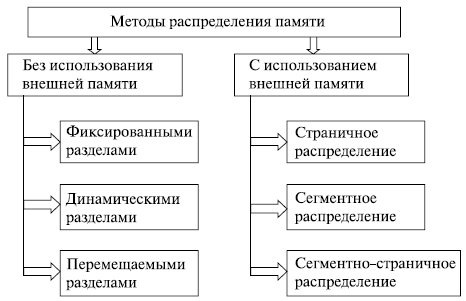
1. Перемещение всех занятых участков в сторону старших или младших адресов при каждом завершении процесса в случае отсутствия раздела достаточного размера.
2. Коррекция таблиц свободных и занятых областей.
3. Изменение адресов команд и данных, к которым обращаются процессы при их перемещении в памяти, за счет использования *относительной адресации*.
4. Аппаратная поддержка процесса динамического преобразования относительных адресов в абсолютные адреса основной памяти.
5. Защита памяти от взаимного влияния других процессов.
6. Распределение памяти

*Память* разделяется на две части - для операционной системы (*ядро*), для выполняющейся программы «пользовательская».

Для идентификации переменных и команд используются **символьные имена (**присваивает *пользователь*,) **виртуальные** (вырабатывает *транслятор*, переводящий программу на *машинный язык*) и **физические** адреса (соответствуют номерам ячеек оперативной памяти)

Два подхода к преобразованию виртуальных адресов в физические.

* 1. Преобразование выполняется один раз во время начальной загрузки программы в *память* с помощью перемещающего *загрузчика*. Содержит данные о начальном адресе физической памяти и информацию об адресно-зависимых элементах.
  2. Динамическое преобразование адресов. Базовый регистр и граничный *регистр*. Относительный адрес + базовый регистр = абсолютный адреса. Абсолютный *адрес* сравнивается с граничным регистром. Если полученный абсолютный *адрес* принадлежит данному процессу, *команда* выполняется.



1. Фиксированное распределение: Одинаковый или разный размер разделов (в первом случае процесс загружается в любой доступный раздел, - во втором в наименьшем разделе)

2. Динамическое распределение - процессы загружаются в *память*, начиная с адреса, где заканчивается ОС, и используют столько памяти, сколько требуется.

**Внешняя фрагментация** - наличие *множества* мелких свободных участков памяти, в которых нет возможности разместить какой-либо новый процесс.

**Уплотнение** - перемещением всех занятых участков так, чтобы вся свободная *память* образовала единую свободную область. Необходима корректировка таблиц свободных и занятых областей.

**19.Страничная организация виртуальной памяти;**

Разделение памяти на страницы, небольшие разделы фиксированного размера, которые перемещаются между оперативной памятью и жестким диском.

Чаще всего страницы весят 4 КБ. 4096 Байт

Как только программе не хватает памяти, тогда она запрашивает еще страницу

При старте процесса выделяется страница на **кодовый сегмент** и **сегмент данных**.

**Сегмент данных** - место хранения данных на жестком диске. Содержит начальный и конечный адрес.

Файл страниц содержит следующую информацию:

1) номер физической страницы 2) Признак присутствия 3) Признак модификации 4) Признак обращения 5) Другие управляющие виды

Признак присутствия, модификации, обращения - это 3 признака, которые могут принять значение 0 или 1

Признак модификации и обращения определяется в 0, если значение за время использования не менялось. Каждые 30 секунд признаки сбрасываются до 0.

Если ОС доходит до 1 1 1, то она будет удалять процесс по порядку, начиная с 1

В настоящее время существуют 6 стратегий для управления страничной виртуальной памятью:

1) Стратегия выборки (отвечает на вопрос, когда)

2) Стратегия размещения (отвечает на вопрос где)

3) Стратегия замещения (отвечает на вопрос какие) FIFO

4) стратегия управления резидентным множеством (отвечает на вопрос сколько)

5) стратегия очистки (отвечает на вопрос, когда)

6) стратегия управления загрузкой (отвечает на вопрос сколько)

10. **Оптимизация функционирования страничной виртуальной памяти**;

Методы повышения эффективности:

1. более сложная структура виртуального адресного пространства, например, двухуровневая (типичная для 32-битовой адресации);

2. использование специального высокоскоростного кэша для хранения части записей таблицы страниц, который обычно называют буфером быстрого преобразования адреса, или буфером поиска трансляции (translation lookaside buffer – TLB);

3. выбор оптимального размера страницы виртуальной памяти;

4. эффективное управление страничным обменом.

Какая стратегия LIFO реализуется в ОС? - НИКАКАЯ.

**20. Сегментная организация виртуальной памяти;**

Сегмент - нефиксированные разделы памяти

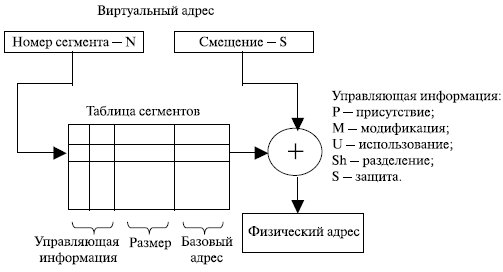
* 1. Сегмент содержит линейную последовательность адресов от 0 до некоторого максимума.
  2. Различные сегменты могут быть различной длины.
  3. Длины могут изменяться.
  4. Каждый сегмент составляет отдельное адресное пространство, разные сегменты могут расти и сокращаться независимо друг от друга.

Максимальный размер сегмента определяется разрядностью виртуального адреса, например, при 32-разрядном микропроцессоре он равен 2^32 = 4 Гбайт.

При загрузке процесса в ОП помещается только часть его сегментов, полная копия виртуального адресного пространства находится в дисковой памяти.

Таблица сегментов процесса:

* базовый физический адрес начала сегмента в оперативной памяти;
* размер сегмента;
* правила доступа к сегменту;
* признаки модификации, присутствия и обращения к данному сегменту, а также некоторая другая информация.



**14. Эффекты виртуализации;**

Виртуализация позволяет запускать изолированные и безопасные виртуальные машины на одном физическом узле, но не позволяет запускать операционные системы с ядрами, отличными от типа ядра базовой операционной системы. Хостовая операционная система отвечает за разделение аппаратных ресурсов между несколькими виртуальными машинами и поддержку их независимости друг от друга.

Виртуализация обеспечивает высокий уровень работоспособности и безопасности благодаря нескольким ключевым возможностям:

1.Локализация неисправностей. Программный сбой в одном разделе никак не влияет на работу приложения в другом разделе.

2.Гибкая обработка отказов. Виртуальные разделы можно настроить так, чтобы обеспечить автоматическую обработку отказов для одного или нескольких приложений. Если требуется еще более высокий уровень работоспособности, аварийный раздел можно разместить на отдельной платформе.

**15.Мультипрограммирование. Формы многопрограммной работы. Семафоры и мониторы;**

**Семафор** - переменная типа boolean, которая указывает на ресурсе, что он занят.

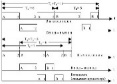
**Монитор** - имеет тип struct. В контекст процесса добавляется структура со всеми возможными ресурсами, и ОС по этой структуре мониторит занятые ресурсы этим процессом

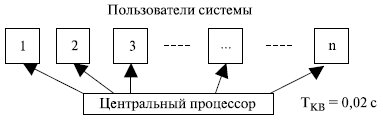
Мультипрограммирование призвано повысить эффективность использования вычислительной системы. Показателями эффективности:

* 1) пропускная способность – количество задач, выполняемых системой в единицу времени;
* 2) удобство работы пользователей, работа в интерактивном режиме с несколькими приложениями на одной машине;
* 3) реактивность системы – способность выдерживать заранее заданные (возможно, очень короткие) интервалы времени между запуском программы и получением конечного результата.

**ОС делятся** на: **системы пакетной обработки**, **системы разделения времени** и **системы реального времени**.

Системы пакетной обработки: пакет заданий формируется так, чтобы получающаяся мультипрограммная смесь сбалансированно загружала все устройства машины. Общее время выполнения смеси задач меньше, чем суммарное время их последовательного выполнения. При этом времени выполнения отдельной задачи может быть затрачено больше, чем при монопольном ее выполнении.





В системах разделения времени пользователям предоставляется возможность интерактивной работы сразу с несколькими приложениями. Для этого каждое приложение должно регулярно получать возможность "общения" с пользователем. ОС принудительно приостанавливает приложения, не дожидаясь, когда они "добровольно" освободят процессор. Всем приложениям попеременно выделяются кванты времени процессора. Если время кванта выбрано достаточно небольшим, то у всех пользователей складывается впечатление единоличной работы на машине.

**Семафор** - представляет собой целую переменную, принимающую неотрицательные значения, доступ любого процесса к которой, за исключением момента ее инициализации, может осуществляться только через две атомарные операции: P (проверять) и V (увеличивать).

P(S): пока S == 0 процесс блокируется; S = S - 1; V(S): S = S + 1; при выполнении операции P над семафором S сначала проверяется его значение. Если оно больше 0, то из S вычитается 1. Если оно меньше или равно 0, то процесс блокируется до тех пор, пока S не станет больше 0, после чего из S вычитается 1. При выполнении операции V над семафором S к его значению просто прибавляется 1.

Одной из типовых задач, требующих организации взаимодействия процессов, является задача производитель-потребитель. Пусть два процесса обмениваются информацией через буфер ограниченного размера. Производитель закладывает информацию в буфер, а потребитель извлекает ее оттуда. Если буфер забит, то производитель должен ждать, пока в нем появится место, чтобы положить туда новую порцию информации. Если буфер пуст, то потребитель должен дожидаться нового сообщения.

Попутно заметим, что семафоры использовались здесь для достижения двух целей: организации взаимоисключения на критическом участке и синхронизации скорости работы процессов.

Монитор обладает своими собственными переменными, определяющими его состояние. Значения этих переменных извне монитора могут быть изменены только с помощью вызова функций методов, принадлежащих монитору. Блок инициализации внутренних переменных содержит операции, которые выполняются только один раз: при создании монитора или при самом первом вызове какой-либо функции-метода до ее исполнения.

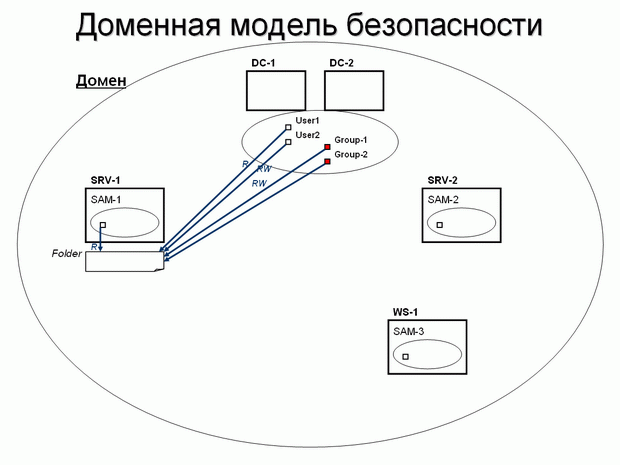
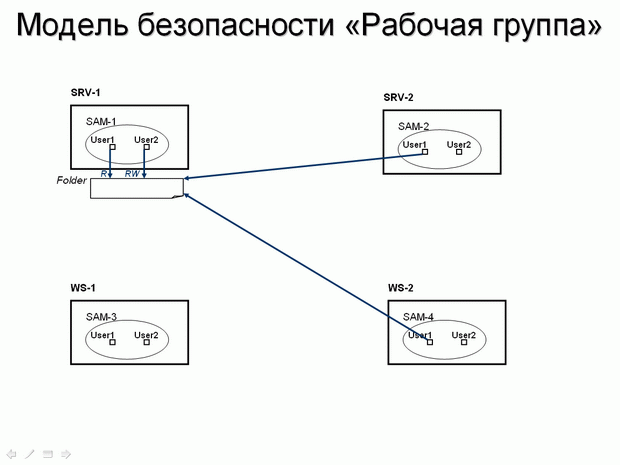
В любой момент времени только один процесс может быть активен, т. е. находиться в состоянии "готовность" или "исполнение", внутри данного монитора. В мониторах было введено понятие **условных переменных**, над которыми можно совершать две операции – wait и signal.

Если функция монитора не может выполняться дальше, пока не наступит некоторое событие, она выполняет операцию wait над какой-либо условной переменной. При этом процесс, выполнивший операцию wait, блокируется, становится неактивным, и другой процесс получает возможность войти в монитор.

Когда ожидаемое событие происходит, другой процесс внутри функции-метода совершает операцию signal над той же самой условной переменной. Это приводит к пробуждению ранее заблокированного процесса, и он становится активным. Если несколько процессов дожидались операции signal для этой переменной, то активным становится только один из них

**22. Модель "Рабочая группа", Доменная модель;**

Компьютер в сети имеет свою собственную локальную БД(SAM) учетных записей и с помощью этой *локальной БД* осуществляется управление доступом к ресурсам данного компьютера. *Локальная БД* хранится в реестре операционной системы.



В данном примере изображены два сервера (SRV-1 и SRV-2) и две рабочие станции (WS-1 и WS-2). Их базы данных SAM обозначены соответственно. В каждой БД есть учетные записи пользователей User1 и User2. Если пользователь с именем User1 локально зарегистрируется в системе на компьютере, например, WS-2, то при попытке получить доступ с этого компьютера по сети к папке Folder на сервере SRV-1 сервер запросит пользователя ввести имя и пароль (исключение составляет тот случай, если у пользователей с одинаковыми именами одинаковые пароли).

**Доменная модель**

Существует единая база данных служб каталогов, доступная всем компьютерам сети. Для этого в сети устанавливаются специализированные серверы, называемые *контроллерами домена*. Серверы DC-1 и DC-2 — контроллеры домена, они хранят доменную базу данных учетных записей (каждый контроллер хранит у себя свою собственную копию БД, но все изменения, производимые в БД на одном из серверов, реплицируются на остальные контроллеры).

В такой модели, если, например, на сервере SRV-1, являющемся членом домена, предоставлен общий доступ к папке Folder, то права доступа к данному ресурсу можно назначать не только для учетных записей локальной базы SAM данного сервера, но, самое главное, учетным записям, хранящимся в доменной БД. На рисунке для доступа к папке Folder даны права доступа одной локальной учетной записи компьютера SRV-1 и нескольким учетным записям домена (пользователя и группам пользователей). В доменной модели управления безопасностью пользователь регистрируется на компьютере ("входит в систему") со своей *доменной учетной записью* и, независимо от компьютера, на котором была выполнена регистрация, получает доступ к необходимым сетевым ресурсам. И нет необходимости на каждом компьютере создавать большое количество локальных учетных записей, все записи созданы *однократно в доменной БД*. И с помощью доменной базы данных осуществляется *централизованное управление доступом* к сетевым ресурсам *независимо от количества компьютеров в сети*.

23. Серверы Глобального каталога и Хозяева операций;

Такие контроллеры домена называются *Хозяевами операций – специально выделенные контроллеры домена*

Всего имеется пять специализированных ролей:

1. **Хозяин схемы**: контролирует возникающие изменения Схемы базы данных AD (добавление и удаление классов объектов, модификация набора атрибутов). *Один* контроллер домена *в масштабе всего леса* выполняет эту роль (по умолчанию — самый первый контроллер в лесу).
2. **Хозяин именования доменов**: контролирует процесс добавления или удаления доменов в лесу.
3. **Эмулятор PDC**: управляет изменениями паролей; является предпочтительным для редактирования групповых политик; является сервером времени для остальных контроллеров.
4. **Хозяин RID, распределитель идентификаторов учетных записей**: выделяет контроллерам домена пулы относительных идентификаторов (RID, которые являются уникальной частью идентификаторов безопасности SID).
5. **Хозяин инфраструктуры**: отвечает за обновление связей "пользователи — группы" между доменами. Эта роль не должна храниться на контроллере домена, который также является *сервером Глобального Каталога* – *хозяин инфраструктуры* не будет работать в данном сценарии (за исключением случая, когда в домене всего один контроллер).

**Глобальный каталог** — это перечень *всех объектов* леса Active Directory. Содержится информация *о каждом объекте* (хотя и не обо всех атрибутах этих объектов), находящемся в данном лесу.

*Сервер глобального каталога* выполняет две очень важные **функции**:

* поиск объектов по всему лесу
* аутентификация пользователей

24. Управление пользователями и группами. Типы пользовательских учетных записей. Профиль. Область действия групп

Учетные записи пользователей домена хранятся на контроллерах домена, хранящих копию базы данных Active Directory.

* **Локальные учетные записи пользователей**. Эти учетные записи существуют в локальной базе данных *SAM* (*Security Accounts Manager*).
* **Учетные записи пользователей домена**. Эти учетные записи хранятся в Active Directory и могут использоваться для входа в систему и доступа к ресурсам по всему лесу AD.
* **Встроенные учетные записи**. Эти учетные записи создаются самой системой и не могут быть удалены. По умолчанию любая система создает две учетные записи – *Administrator* (*Администратор*) и *Guest* (*Гость*).

Два основных вида имен входа — это с использованием *основного имени пользователя* и имени входа пользователя в системах пред-Windows 2000.

Основное имя пользователя (*UPN*, *User Principle Name*) имеет включает в себя имя входа пользователя, затем значок " @ " и имя домена.

Также можно создавать дополнительные доменные суффиксы (та часть имени, которая стоит после знака @).

Все UPN в лесу должны быть уникальными.

Доменные учетные записи пользователей (а также компьютеров и групп) хранятся в специальных контейнерах AD. Это могут быть либо стандартные контейнеры *Users* для пользователей и *Computers* для компьютеров, либо созданное администратором Организационное подразделение (ОП). Исключение составляют учетные записи контроллеров домена, они всегда хранятся в ОП с названием *Domain Controllers*.

*Профиль* — это настройки рабочей среды пользователя. Профиль содержит: настройки рабочего стола, настройки просмотра папок компьютера, настройки обозревателя Интернета и других.

Различают следующие виды профилей:

* *локальные* — хранятся в папке "*Documents and Settings* " на том разделе диска, где установлена операционная система;
* *перемещаемые* (сетевые, или *roaming* ) — хранятся на сервере в папке общего доступа, загружаются в сеанс пользователя на любом компьютере, с которого пользователь вошел (зарегистрировался) в домен, давая возможность пользователю иметь одинаковую рабочую среду на любом компьютере (путь к папке с профилем указывается на данной закладке в виде адреса \\server\share\%username%,);
* *обязательные* (*mandatory*) — настройки данного типа профиля пользователь может изменить только в текущем сеансе работы в Windows, при выходе из системы изменения не сохраняются.

Типы групп

* Группы безопасности — каждая группа данного типа имеет *идентификатор безопасности* ( *Security Identifier*, или *SID* ), поэтому группы безопасности используются для назначения разрешений при определении прав доступа к различным сетевым ресурсам.
* Группы распространения — главное назначение — организация списков рассылки для почтовых программ (например, для Microsoft Exchange Server).

Область действия групп

* **Локальные в домене** могут содержать — глобальные группы из любого домена, универсальные группы, глобальные учетные записи пользователей из любого домена леса, используются — при назначении прав доступа только к ресурсам "своего" домена;
* **Глобальные** могут содержать — только глобальные учетные записи пользователей "своего" домена, используются — при назначении прав доступа к ресурсам любого домена в лесу;
* **Универсальные** могут содержать — другие универсальные группы всего леса, глобальные группы всего леса, глобальные учетные записи пользователей из любого домена леса, используются — при назначении прав доступа к ресурсам любого домена в лесу.

**21. Встроенные локальные группы (на рабочей станции или простом сервере);**

|  |  |
| --- | --- |
|  |  |
| *Администраторы* | все административные задачи. \**Администраторы домена*. |
| *Операторы резервного копирования* | Резервное копирование и восстановление данных |
| *Администраторы DHCP* | администрировать службу DHCP Server. |
| *Операторы сетевой конфигурации* | Изменять настройки TCP/IP, а также обновлять и освобождать IP-адреса |
| *Пользователи монитора производительности* | Следить за счетчиками производительности на сервере. |
| *Пользователи журнала производительности* | администрировать журналы производительности, счетчики и оповещения |
| *Опытные пользователи* | создавать и модифицировать учетные записи пользователей, а также устанавливать программы на локальном компьютере, но не могут просматривать файлы других пользователей. создавать и удалять локальные группы, а также добавлять и удалять пользователей в группах, которые они создали. добавлять и удалять пользователей в группах *Опытные пользователи*, *Пользователи* и *Гости*. |
| *Операторы печати* | управлять принтерами и очередями печати на конкретном сервере. |
| *Пользователи удаленного рабочего стола* | Членам группы разрешается выполнять подключение к удаленному рабочему столу компьютера. |
| *Пользователи* | Не могут устанавливать программы или вносить изменения в систему. |

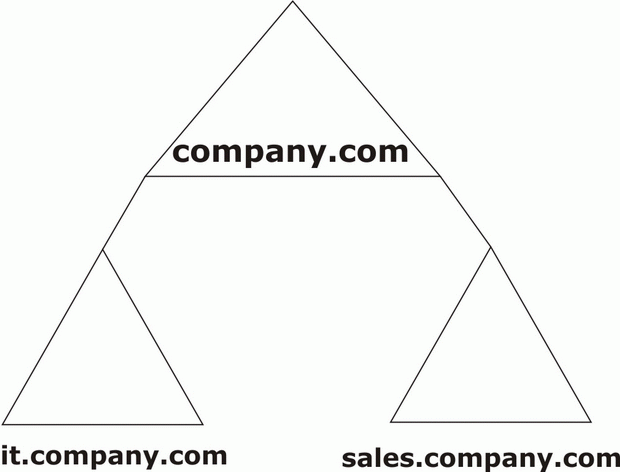
31. Встроенные доменные локальные группы;

|  |  |
| --- | --- |
| **Название группы** | **Описание** |
| *Администраторы* | права администратора на всех контроллерах домена и в самом домене. |
| *Операторы учетных записей* | создавать, удалять и управлять пользователями и группами. Они не могут модифицировать группу *Администраторы*, *Администраторы домена*, *Контроллеры домена* или любую из групп *Операторы*. |
| *Операторы резервного копирования* | выполнять вход, резервное копирование и восстановление данных на всех контроллерах домена, и завершать их работу. |
| *Администраторы DNS* | Члены группы имеют административный доступ к серверам DNS. |
| *Операторы сетевой конфиг.* | изменять настройки TCP/IP на контроллерах доменов. |
| *Пользователи монитора производительности* | следить за счетчиками производительности на контроллерах домена. |
| *Пользователи журнала производительности* | управлять журналами производительности, счетчиками и оповещениями на контроллерах домена. |
| *Операторы печати* | управлять работой принтеров домена |
| *Операторы сервера* | выполнять большинство административных задач на контроллерах домена, за исключением изменения параметров безопасности. |
| *Пользователи* | входить в систему, работать с программами, сохранять документы и завершать работу пк. Не могут устанавливать программы или вносить изменения в систему. |

33. Логическая и физическая структуры, управление репликацией AD;

**Домен** — логическая группа пользователей и компьютеров, которая поддерживает централизованное администрирование и управление безопасностью. Домен является единицей безопасности – это означает, что администратор для одного домена, по умолчанию, не может управлять другим доменом.

**Дерево** является набором доменов, которые связаны отношениями "дочерний"/"родительский", а также используют связанные (смежные, или прилегающие) пространства имен. При этом дочерний домен получает имя от родительского. Между доменами автоматически устанавливаются двухсторонние транзитивные доверительные. Это означает, что доверительные отношения могут быть использованы всеми другими доменами данного леса для доступа к ресурсам данного домена.



**Лес** — это одно или несколько деревьев, которые разделяют общую *схему*, серверы *Глобального каталога* и *конфигурационную информацию*. В лесу все домены объединены транзитивными двухсторонними доверительными отношениями.

**Организационное подразделение (ОП)** - контейнер, который помогает группировать объекты для целей администрирования или применения групповых политик. ОП могут быть созданы для организации объектов в соответствии с их функциями, местоположением, ресурсами и так далее.

*Логическая структура* Active Directory позволяет организовать ресурсы корпоративной сети таким образом, чтобы они отражали структуру самой компании.

Физическая структура Active Directory служит для связи между логической структурой AD и *топологией* корпоративной сети.

Основные элементы физической структуры Active Directory — *контроллеры домена* и *сайты*.

**Сайт** — логическое объединение контроллеров домена и клиентов для управления служебным трафиком службы каталогов. Назначение сайтов — управление процессом репликации между контроллерами доменов и процессом аутентификации пользователей (использование глобального каталога для поиска объектов). В самом начале создания леса автоматически создается сайт по умолчанию с именем *Default-First-site-Name*. В дальнейшем сетевой администратор должен сам планировать и создавать новые сайты и определять входящие в них подсети, а также перемещать в сайты соответствующие контроллеры доменов.

Внутри сайта контроллеры домена соединены линиями с высокой пропускной способностью. Поэтому репликация между контроллерами производится каждые 5 минут, данные при передаче не сжимаются, для взаимодействия между серверами используется **технология вызова удаленных процедур (RPC).** Для репликации между сайтами может использоваться также протокол **SMTP**, данные при передаче сжимаются, передача изменений происходит по определенному расписанию. Если имеется несколько маршрутов передачи данных, то система выбирает маршрут с наименьшей стоимостью.

**Топологию репликации**, т.е. порядок, в котором серверы опрашивают друг друга для получения изменений в базе данных, серверы строят автоматически (эту задачу выполняет компонента служб каталогов, называемая *Knowledge Consistency Checker*, или *KCC*). При достаточно большом количестве контроллеров KCC строит кольцевую топологию репликации, причем для надежности образует несколько колец, по которым контроллеры передают данные репликации.

Для каждого вида межсайтового транспорта определяется " *соединение сайтов* " ( *site link* ), с помощью которого строится управление репликацией между двумя и более сайтами. Именно для соединения ("линка") задаются такие параметры как " *Расписание репликации* " и " *Стоимость*

34. Управление Организационными подразделениями, делегирование. Групповые политики. Управление приложениями;

Назначение *Организационных подразделений* — организация иерархической структуры объектов AD внутри домена.

**Делегирование полномочий** - распределение нагрузки по администрированию на уровне всего домена.

**Механизм *Групповых политик*** *-* позволяет автоматизировать процесс управления рабочими станциями, серверами, пользователями. С помощью групповых политик (*ГП*) можно настраивать различные параметры компьютеров и пользовательской рабочей среды сразу в масштабах сайта AD, домена, организационного подразделения.

***Объект групповых политик*** состоит из 2 частей:

* 1. ***контейнера групповых политик,*** хранящегося в БД Active Directory
  2. ***шаблона групповых политик***, хранящегося в файловой системе контроллера домена

Каждый объект политик содержит два раздела: **конфигурация компьютера** и **конфигурация** **пользователя**.

Задание параметров групповых политик производится **Редактором групповых политик**

Имеются следующие методы управления применением групповых политик

* блокировка наследования политик на каком-либо уровне иерархии AD;
* запрет блокировки конкретного объекта групповых политик;
* управление приоритетом применения политик на конкретном уровне AD
* разрешение на применение политик

Кроме применения политик в момент загрузки компьютера или входа пользователя в систему, каждый компьютер постоянно запрашивает обновленные политики на контроллерах домена, загружает их и применяет обновленные параметры (и к пользователю, и к компьютеру). Рабочие станции домена и простые серверы запрашивают обновления каждые 90 ± 30 минут, контроллеры домена обновляют свои политики каждые 5 минут.

Управление приложениями

Групповые политики могут использоваться для установки прикладных программ в масштабах всего домена или отдельного организационного подразделения.

Используются следующие способы управления установкой приложений:

* **назначение приложений компьютерам** (при данном способе приложение, назначенное компьютеру, автоматически устанавливается при загрузке компьютера);
* **назначение приложений пользователям** (приложение устанавливается при первом вызове данного приложения);
* **публикация приложений пользователям** (название приложения добавляется к списку доступных для установки программ в окне " *Установка и удаление программ* " в *Панели управления* ).

**32. Назначение службы каталогов Active Directory;**

*Active* *Directory* отвечает за создание и организацию доменов, OU (организационные *подразделения*) и сайты.

*Служба каталогов* *Active* *Directory* (сокращенно — *AD*) обеспечивает эффективную работу сложной корпоративной среды, предоставляя следующие возможности:

* ***Единая регистрация в сети***; Пользователи могут регистрироваться в сети с одним именем и паролем и получать при этом доступ ко всем сетевым ресурсам и службам (службы сетевой инфраструктуры, службы файлов и печати, серверы приложений и баз данных и т. д.);
* ***Безопасность информации***. Средства аутентификации и управления доступом к ресурсам, встроенные в службу Active Directory, обеспечивают централизованную защиту сети;
* ***Централизованное управление***. Администраторы могут централизованно управлять всеми корпоративными ресурсами;
* ***Администрирование с использованием групповых политик***. При загрузке компьютера или регистрации пользователя в системе выполняются требования групповых политик; их настройки хранятся в *объектах групповых политик* (*GPO*) и применяются ко всем учетным записям пользователей и компьютеров, расположенных в сайтах, доменах или организационных подразделениях;
* ***Интеграция с DNS***. Функционирование служб каталогов полностью зависит от работы службы DNS. В свою очередь серверы DNS могут хранить информацию о зонах в базе данных Active Directory;
* ***Расширяемость каталога***. Администраторы могут добавлять в схему каталога новые классы объектов или добавлять новые атрибуты к существующим классам;
* ***Масштабируемость***. Охват доменов, деревьев и леса;
* ***Репликация информации***. В службе Active Directory используется репликация служебной информации, что позволяет модифицировать БД Active Directory на любом контроллере домена.
* ***Гибкость запросов к каталогу***. БД Active Directory может использоваться для быстрого поиска любого объекта AD, используя его свойства (например, имя пользователя или адрес его электронной почты, тип принтера или его местоположение и т. п.);
* ***Стандартные интерфейсы программирования***. Для разработчиков программного обеспечения служба каталогов предоставляет доступ ко всем возможностям (средствам) каталога и поддерживает принятые стандарты и интерфейсы программирования (API).

25. Служба DHCP. Планирование серверов DHCP;

Служба *DHCP* ( *Dynamic Host Configuration Protocol* ) — это одна из служб поддержки протокола TCP/IP, разработанная для упрощения администрирования IP-сети за счет использования специально настроенного сервера.

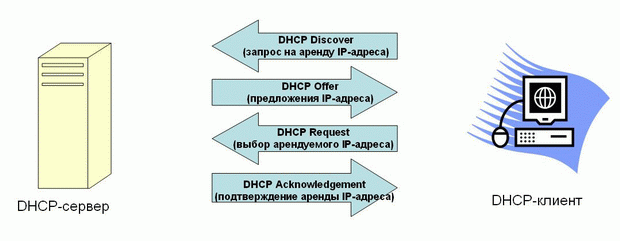
Определим основные термины, относящиеся к службе DHCP

* **Клиент DHCP** — сетевой узел с динамическим IP-адресом, полученным от сервера DHCP;
* **Период аренды** — срок, на который клиенту предоставляется IP-адрес;
* **Область** — это полный последовательный диапазон допустимых IP-адресов в сети
* **Исключаемый диапазон** — последовательность IP-адресов, которая исключается из числа адресов, предлагаемых службой DHCP;
* **Доступный пул** — адреса, оставшиеся после определения области DHCP и исключаемых диапазонов
* **Резервирование** — назначение DHCP постоянного IP-адреса.

Сервер DHCP избавляет сетевого администратора от необходимости ручного выполнения таких операций, как:

* назначение сетевым узлам IP-адресов и прочих параметров протокола TCP/IP;
* недопущение дублирования IP-адресов;
* освобождение удаленных IP-адресов;
* ведение централизованной БД выданных IP-адресов.

Рассмотрим технологию предоставления IP-адресов DHCP-сервером DHCP-клиентам



Запрос – предложения – выбор – подтверждение.

Основные рекомендации при развертывании службы DHCP в корпоративной сети будут следующими:

* отдельный DHCP-сервер в каждый IP Сети.
* *агент ретрансляции DHCP-запросов*. Чтобы он пересылал широковещательные запросы DHCP из подсети, в которой нет DHCP-сервера, на соответствующий DHCP-сервер, а на самом DHCP-сервере создать области для всех обслуживаемых IP-сетей;
* для повышения отказоустойчивости следует установить несколько серверов DHCP, при этом на каждом DHCP-сервере, кроме областей для "своих" IP-сетей, необходимо создать области для других подсетей (при этом диапазоны IP-адресов в таких резервных областях не должны пересекаться с основными областями, созданными на серверах DHCP в "своих" подсетях);
* в больших IP-сетях DHCP-серверы должны иметь мощные процессоры, достаточно большие объемы оперативной памяти и быстродействующие дисковые подсистемы, т.к. обслуживание большого количества клиентов требует интенсивной работы с базой данных DHCP-сервера.

36. Настройка DHCP-клиентов. Агент ретрансляции DHCP-запросов;

Клиентом DHCP может быть практически любое сетевое устройство, настроенное на автоматическое получение IP-адреса. Из операционных систем корпорации Microsoft клиентом DHCP могут быть все системы, имеющие стек TCP/IP (вплоть до системы MS-DOS).

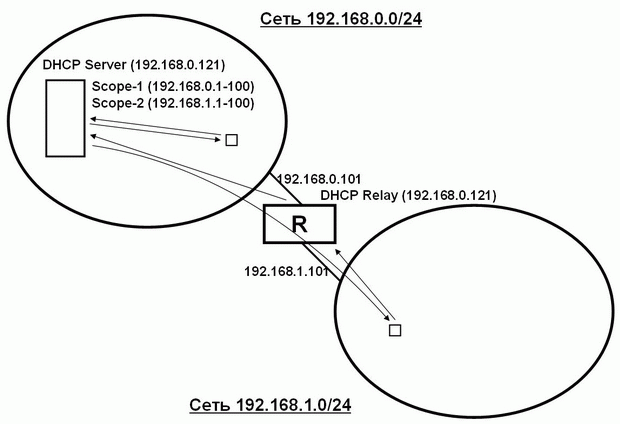
Клиенты посылают запрос на аренду IP-адреса при своей первой загрузке, при смене настройки получения IP-адреса со статической на динамическую, а также с помощью команд ipconfig /release (освобождение арендованного IP-адреса) и ipconfig /*renew* (запрос на новую аренду).

**Внимание**! Сервер DHCP обязательно должен иметь **статические IP-адреса** на всех установленных в нем сетевых адаптерах.

**Замечание**. При отсутствии в сети DHCP-сервера клиенты, настроенные на автоматическую настройку IP-адреса будут самостоятельно назначать себе IP-адреса из подсети класса B — 169.254.0.0/16. Данная технология называется *автоматической IP-адресацией* ( *APIPA, Automatic Private IP Addressing*)

**Агент ретрансляции DHCP-запросов**

Чтобы широковещательные запросы на аренду IP-адреса достигали DHCP-сервер из любой подсети, необходимо на маршрутизаторах, объединяющих разные IP-сети в единую сеть, установить и настроить *агент ретрансляции DHCP* ( *DHCP Relay Agent* ).



Две IP-сети класса C — 192.168.0.0/24 и 192.168.1.0/24. DHCP установлен в первой подсети и содержит 2 области — *Scope-1* и *Scope-2*. Между двумя подсетями установлен маршрутизатор *R*, имеющий в первой подсети сетевой интерфейс с IP-адресом 192.168.0.101, а во второй подсети сетевой интерфейс с IP-адресом 192.168.1.101. На маршрутизаторе запущен агент ретрансляции DHCP-запросов, настроенный на перенаправление широковещательных DHCP-запросов на сервер с IP-адресом 192.168.0.121.

Клиенты DHCP, находящиеся в первой подсети, посылают широковещательные запросы на аренду IP-адреса, которые напрямую попадают на DHCP-сервер.

Клиенты DHCP, находящиеся во второй подсети, также посылают широковещательные запросы на аренду IP-адреса, которые не могут напрямую попасть на DHCP-сервер, т.к. маршрутизаторы не пропускают широковещательные пакеты из одной подсети в другую. Но если широковещательный пакет является запросом на аренду IP-адреса, то агент ретрансляции перехватывает данный пакет и пересылает его напрямую на DHCP-сервер. DHCP-сервер, видя от агента ретрансляции, что запрос пришел из второй подсети, выдает клиенту IP-адрес из пула адресов, заданных во второй области.

26. Пространство имен NetBIOS. Процесс разрешения имен в пространстве NetBIOS;

**NetBIOS** - программный интерфейс для приложений клиент – сервер, линейный список имен компьютеров, точнее работающих на компьютере служб. Имена компьютеров состоят из 15 видимых символов плюс 16-й служебный символ. Если видимых символов меньше 15, то оставшиеся символы заполняются двоичными нулями. 16-й символ соответствует службе, работающей на компьютере с данным именем.

Просмотреть список имен пространства NetBIOS, которые имеются на данном компьютере, можно с помощью команды " nbtstat –n ". В угловых скобках указан **шестнадцатеричный код** 16-го служебного символа какого-либо имени. Например, " 00 " соответствуют службе " *Рабочая станция* ", которая выполняет роль *клиента* при подключении к ресурсам файлов и печати. Имя WORLD соответствует либо Net-BIOS-имени домена world.ru, либо имени т.н. сетевой *рабочей группы*, отображаемой в *Сетевом окружении* любого Windows-компьютера.

**Процесс разрешения имен в пространстве NetBIOS**

Клиент должен определить IP-адрес сервера, по имени этого сервера. Это может быть сделано одним из трех способов:

* широковещательный запрос;
* обращение к локальной базе данных NetBIOS-имен,
* обращение к централизованной БД имен NetBIOS.

Разрешение имен осуществляется с помощью различных комбинаций перечисленных способов:

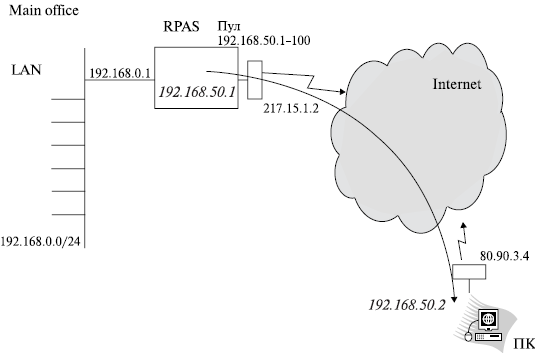
* **b-узел** (*широковещательный узел*) — компьютер, которому нужно разрешить имя, рассылает по локальной сети широковещательное сообщение с запросом IP-адреса по имени компьютера;
* **p-узел** (*peer node*, *узел "точка — точка"*) - когда клиенту нужно разрешить имя компьютера в IP-адрес, клиент отправляет Wins`у имя, а тот в ответ посылает адрес;
* **m-узел** (*смешанный узел*) — комбинирует запросы b- и р-узла (WINS-клиент смешанного типа сначала пытается применить широковещательный запрос, а в случае неудачи обращается к WlNS-серверу;
* **h-узел** (*гибридный узел*) — также комбинирует запросы b-узла и р-узла, но при этом сначала используется запрос к WINS-серверу и лишь в случае неудачи начинается рассылка широковещательного сообщения.

27. Виртуальные частные сети. Технологии виртуальных частных сетей;

*Виртуальные частные сети* — технология создания защищенных подключений между компьютерами, подключенными к публичным сетям (например, к сети Интернет).

* как получить доступ в корпоративную сеть;
* как защитить данные, передаваемые через Интернет

Обе эти задачи решаются созданием VPN-подключений между удаленным пользователем и сервером удаленного доступа. Необходимо создать еще одно подключение, но не через модем, а через " *Подключение к виртуальной частной сети* ". При этом в качестве "телефонного номера" выступит IP-адрес внешнего интерфейса сервера удаленного доступа.



Между ПК мобильного пользователя и сервером удаленного доступа будет установлен защищенный "виртуальный" канал, клиентский ПК получит IP-адрес из пула адресов сервера RRAS (таким образом будет решена задача маршрутизации IP-пакетов между клиентом и корпоративной сетью), все пакеты, передаваемые между клиентом корпоративной сетью, будут шифроваться.

Для создания виртуальных частных сетей в системах семейства Windows используются два различных протокола — *PPTP* и *L2TP*. Основной принцип работы обоих протоколов заключается в том, что они создают защищенный " *туннель* " между пользователем и корпоративной сетью или между двумя подсетями. Туннелирование состоит в том, что пакеты, передаваемые в защищенной сети, снабжаются специальными заголовками (у обоих протоколов свои заголовки), содержимое данных в этих пакетах шифруется, а затем пакет, предназначенный для защищенной корпоративной сети и имеющий заголовок с IP-адресами внутренней корпоративной сети, инкапсулируется в пакет, передаваемый по сети Интернет и имеющий соответствующий заголовок и IP- адреса отправителя и получателя.

**Политики удаленного доступа**

Разрешениями на подключения к службе удаленного доступа можно управлять с помощью *политик удаленного доступа*.

Каждая политика состоит из трех компонент:

* ***Условия*** — определяются условия подключения пользователя (в сетях на базе MS Windows Server наиболее интересные условия — *день недели и время*, а также *членство в определенной группе*);
* ***Профиль*** — определяются некие параметры подключения (например, тип аутентификации или вид коммуникаций);
* ***Разрешения*** — разрешить или запретить подключение.

В начале проверки политики всегда проверяются условия — если ни одно из условий не совпадает с параметрами учетной записи пользователя, то происходит переход к следующей политике. Если условия совпали, то проверяются параметры профиля подключения, если параметры политики и пользователя не совпадают, то также происходит переход к следующей политике. Если же параметры профиля совпали, и данная политика разрешает подключение, то пользователю выдается разрешение на подключение к серверу удаленного доступа. Если же политика запрещает подключение, то пользователю выдается отказ на подключение к серверу.

28. Разбиение сетей на подсети с помощью маски подсети;

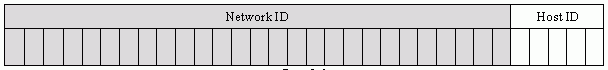
Для более эффективного использования пространства адресов IP-сети с помощью маски подсети могут быть разбиты на более мелкие подсети или объединены в более крупные сети.

Рассмотрим на примере разбиение сети 192.168.1.0/24 (сеть класса C) на более мелкие подсети. В исходной сети в IP-адресе 24 бита относятся к идентификатору сети и 8 бит — к идентификатору узла. Используем маску подсети из 27 бит, или, в десятичном обозначении, — 255.255.255.224, в двоичном обозначении — 11111111 11111111 11111111 11100000. Получим следующее разбиение на подсети:

|  |  |  |
| --- | --- | --- |
|  | | |
| **Подсеть** | **Диапазон IP-адресов** | **Широковещательный адрес в подсети** |
| 192.168.1.0/27 | 192.168.1.1–192.168.1.30 | 192.168.1.31 |
| 192.168.1.32/27 | 192.168.1.33–192.168.1.62 | 192.168.1.63 |
| 192.168.1.64/27 | 192.168.1.65–192.168.1.94 | 192.168.1.95 |
| 192.168.1.96/27 | 192.168.1.97–192.168.1.126 | 192.168.1.127 |
| 192.168.1.128/27 | 192.168.1.129–192.168.1.158 | 192.168.1.159 |
| 192.168.1.160/27 | 192.168.1.161–192.168.1.190 | 192.168.1.191 |
| 192.168.1.192/27 | 192.168.1.193–192.168.1.222 | 192.168.1.223 |
| 192.168.1.224/27 | 192.168.1.225–192.168.1.254 | 192.168.1.255 |

Таким образом, мы получили 8 подсетей, в каждой из которых может быть до 30 узлов. Напомним, что идентификатор узла, состоящий из нулей, обозначает всю подсеть, а идентификатор узла, состоящий из одних единиц, означает широковещательный адрес (пакет, отправленный на такой адрес, будет доставлен всем узлам подсети).

IP-адреса в данных подсетях будут иметь структуру:

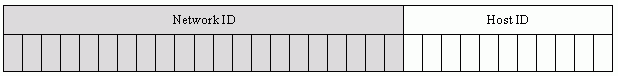


Отметим очень важный момент. С использованием такой маски узлы с такими, например, IP-адресами, как 192.168.1.48 и 192.168.1.72, находятся в различных подсетях, и для взаимодействия данных узлов необходимы маршрутизаторы, пересылающие пакеты между подсетями 192.168.1.32/27 и 192.168.1.64/27.

Примечание. Согласно стандартам протокола, TCP/IP для данного примера не должно существовать подсетей 192.168.1.0/27 и 192.168.1.224/27 (т.е. первая и последняя подсети). На практике большинство операционных систем (в т.ч. системы семейства Microsoft Windows) и маршрутизаторов поддерживают работу с такими сетями.

Аналогично, можно с помощью маски подсети объединить мелкие сети в более крупные.

Например, IP-адреса сети 192.168.0.0/21 будут иметь следующую структуру:



Диапазон IP-адресов данной сети: 192.168.0.1–192.168.7.254 (всего — 2046 узлов), широковещательный адрес подсети — 192.168.7.255.

Преимущества подсетей внутри частной сети:

* разбиение больших IP-сетей на подсети (subnetting) позволяет снизить объем широковещательного трафика (маршрутизаторы не пропускают широковещательные пакеты);
* объединение небольших сетей в более крупные сети (supernetting) позволяет увеличить адресное пространство с помощью сетей более низкого класса;
* изменение топологии частной сети не влияет на таблицы маршрутизации в сети Интернет (хранят только маршрут с общим номером сети);
* размер глобальных таблиц маршрутизации в сети Интернет не растет;
* администратор может создавать новые подсети без необходимости получения новых номеров сетей.

Старшие биты IP-адреса используются рабочими станциями и маршрутизаторами для определения класса адреса. После того как класс определен, устройство может однозначно вычислить границу между битами, использующимися для идентификации номера сети, и битами номера устройства в этой сети. Однако при разбиении сетей на подсети или при объединении сетей для определения границ битов, идентифицирующих номер подсети, такая схема не подходит. Для этого как раз и используется 32-битная маска подсети, которая помогает однозначно определить требуемую границу. Напомним, что для стандартных *классов сетей* маски имеют следующие значения:

* 255.0.0.0 – маска для сети класса А;
* 255.255.0.0 - маска для сети класса В;
* 255.255.255.0 - маска для сети класса С.

Отказ от использования только стандартных классов IP-сетей (A, B, и C) называется бесклассовой междоменной маршрутизацией (Classless Inter-Domain Routing, *CIDR*).

39. Служба DNS: пространство имен, домены. Алгоритмы работы итеративных и рекурсивных запросов DNS;

**DNS** - **это иерархическая база**данных, сопоставляющая имена сетевых узлов и их сетевых служб IP-адресам узлов. Содержимое этой базы, с одной стороны, распределено по большому количеству серверов службы DNS, а с другой стороны, является централизованно управляемым. В основе *иерархической структуры базы данных* DNS лежит **доменное пространство имен** (domain namespace), основной структурной единицей которого является **домен**, объединяющий сетевые узлы (хосты), а также поддомены.

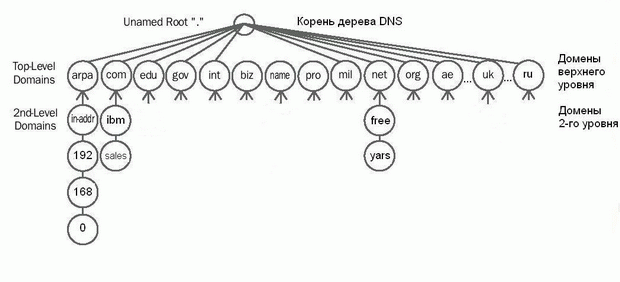
**Разрешение** имени узла в пространстве имен DNS - Процесс поиска в БД службы DNS имени некоего сетевого узла и сопоставления этому имени IP-адреса

Служба DNS состоит из трех основных компонент:

* **Пространство имен DNS и соответствующие ресурсные записи (RR, resource record)** - это сама распределенная база данных DNS;
* **Серверы имен DNS** - компьютеры, хранящие базу данных DNS и отвечающие на запросы DNS-клиентов;
* **DNS-клиенты (DNS-clients, DNS-resolvers)** -компьютеры, посылающие запросы серверам DNS для получения ресурсных записей.

Пространство имен.

**Пространство имен DNS** - иерархическая древовидная структура, начинающаяся с корня, не имеющего имени и обозначаемого точкой ".".



Для доменов 1-го уровня различают 3 категории имен:

* **ARPA** - специальное имя, используемое для обратного разрешения DNS (из IP-адреса в полное имя узла);
* **Общие имена 1-го уровня** - 16 имен, назначение которых приведено;
* **Двухбуквенные имена для стран** - имена для доменов, зарегистрированных в соответствующих странах (например, ru - для России).

**Ресурсные записи (RR)** - Для непосредственного отображения пространства имен в пространство IP-адресов. Каждый сервер DNS содержит ресурсные записи для той части пространства имен, за которую он несет ответственность.

**Серверы имен DNS** - это компьютеры, на которых хранятся те части БД пространства имен DNS, за которые данные серверы отвечают, и функционирует программное обеспечение, которое обрабатывает запросы DNS-клиентов на разрешение имен и выдает ответы на полученные запросы.

DNS-клиенты.

**DNS-клиент** - это любой сетевой узел, который обратился к DNS-серверу для разрешения имени узла в IP-адрес или, обратно, IP-адреса в имя узла.

**Алгоритмы работы итеративных и рекурсивных запросов DNS**

Все запросы, отправляемые DNS-клиентом DNS-серверу для разрешения имен, делятся на два типа:

* итеративные запросы (клиент посылает серверу DNS запрос, в котором требует дать наилучший ответ без обращений к другим DNS-серверам);
* рекурсивные запросы (клиент посылает серверу DNS запрос, в котором требует дать окончательный ответ даже если DNS-серверу придется отправить запросы другим DNS-серверам; посылаемые в этом случае другим DNS-серверам запросы будут итеративными).

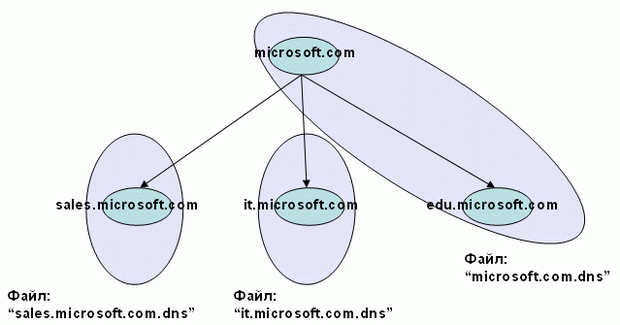
Клиентская часть протокола TCP/IP рабочей станции пользователя сначала просматривает свой локальный кэш разрешенных ранее имен. Если имя не найдено, то клиент посылает запрос DNS-серверу, (назовем данный DNS-сервер *"локальным DNS-сервером"* ), на разрешение имени в IP-адрес данного узла. Далее DNS-сервер обрабатывает запрос.

29. Служба DNS: домены и зоны. Зоны прямого и обратного просмотра;

DNS - это *иерархическая база данных*, сопоставляющая имена сетевых узлов и их сетевых служб IP-адресам узлов. Информация о доменах, хранящаяся в БД сервера DNS, организуется в особые единицы, называемые зонами. *Зона* - это способ представления информации о домене и его поддоменах в хранилище тех серверов DNS, которые отвечают за данный домен и поддомены

Типы зон:

* **Стандартная основная** - главная копия стандартной зоны; только в данном экземпляре зоны допускается производить какие-либо изменения, которые затем реплицируются на серверы, хранящие дополнительные зоны;
* **Стандартная дополнительная** - копия основной зоны, доступная в режиме "только-чтение", предназначена для повышения отказоустойчивости и распределения нагрузки между серверами, отвечающими за определенную зону; процесс репликации изменений в записях зон называется "**передачей зоны**" (информация в стандартных зонах хранится в текстовых файлах, файлы создаются в папке "%system root%\system32\dns");
* **Интегрированная в Active Directory** - вся информация о зоне хранится в виде одной записи в БД Active Directory; в интегрированных зонах можно более жестко управлять правами доступа к записям зоны; изменения в записях зоны между разными экземплярами интегрированной зоны производятся не по технологии передачи зоны службой DNS, а механизмами репликации службы Active Directory;
* **Зона-заглушка** - особый тип зоны, которая для данной части пространства имен DNS содержит самый минимальный набор ресурсных записей (начальная запись зоны, список серверов имен, отвечающих за данную зону, и несколько записей типа A для ссылок на серверы имен для данной зоны).



Пространство имен DNS начинается с домена microsoft.com, который содержит 3 поддомена: sales.microsoft.com, it.microsoft.com и edu.microsoft.com (домены на рисунке обозначены маленькими горизонтальными овалами).

Делегирование управления - передача ответственности за часть пространства имен другим серверам DNS.

**Зоны прямого и обратного просмотра**

Зоны, рассмотренные в предыдущем примере, являются ***зонами прямого***. Данные зоны служат для разрешения имен узлов в IP-адреса.

Для определения имени узла по его IP-адресу служат зоны **обратного просмотра**, основной тип записи в "обратных" зонах - PTR. Для решения данной задачи создан специальный домен с именем in-addr.arpa. Для каждой IP-сети в таком домене создаются соответствующие поддомены, образованные из идентификатора сети, записанного в обратном порядке. Записи в такой зоне будут сопоставлять идентификатору узла полное FQDN-имя данного узла. Например, для IP-сети 192.168.0.0/24 необходимо создать зону с именем "0.168.192.in-addr.arpa". Для узла с IP-адресом 192.168.0.10 и именем host.company.ru в данной зоне должна быть создана запись "10 PTR host.company.ru".

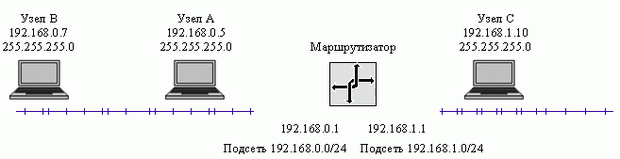
30. IP-маршрутизация.

* **сетевой узел** (node) — любое сетевое устройство с протоколом TCP/IP;
* **хост** (host) — сетевой узел, не обладающий возможностями маршрутизации пакетов;
* **маршрутизатор** (router) — сетевой узел, обладающий возможностями маршрутизации пакетов

IP-маршрутизация — это процесс пересылки трафика от узла-отправителя к узлу –получателю в IP-сети с произвольной топологией.

Заголовок содержит IP-адрес отправителя и IP-адрес получателя.

1. Определение положения (локальная сеть или удаленная). Отправитель логически умножает свое IP на маску подсети и ip получателя на маску. Если результаты совпадают, значит, оба узла находятся в одной подсети.
2. Узел-отправитель проверяет ARP-кэш на наличие в ARP-таблице MAC-адреса узла-получателя. Узел-отправитель посылает ARP-запрос для IP-адреса узла-получателя, ответ помещает в ARP-таблицу и после этого передача пакета производится между сетевыми адаптерами компьютеров.
3. Если узел-отправитель и узел-получатель расположены в разных IP-сетях, то узел-отправитель посылает данный пакет "Основному шлюзу". **Основной шлюз** — это маршрутизатор, который отвечает за отправку пакетов в другие подсети.



Каждый сетевой узел принимает решение о маршрутизации пакета на основе таблицы маршрутизации, которая хранится в оперативной памяти данного узла. Таблицы маршрутизации существуют не только у маршрутизаторов с несколькими интерфейсами, но и у рабочих станций, подключаемых к сети через сетевой адаптер. Каждая таблица маршрутизации содержит набор записей.

***Список интерфейсов*** — список сетевых адаптеров, установленных в компьютере.

***Сетевой адрес*** — диапазон IP-адресов, которые достижимы с помощью данного маршрута.

***Маска сети*** — маска подсети, в которую отправляется пакет с помощью данного маршрута.

***Адрес шлюза*** — IP-адрес узла, на который пересылаются пакеты, соответствующие данному маршруту.

***Интерфейс*** — обозначение сетевого интерфейса данного компьютера, на который пересылаются пакеты, соответствующие маршруту.

***Метрика*** — условная стоимость маршрута, это количество маршрутизаторов, которые должен пройти пакет, чтобы попасть в нужную сеть.

38. Адресация узлов в IP-сетях. Отображение IP-адресов на физические адреса;

В сетях TCP/IP принято различать **адреса сетевых узлов трех уровней**

* физический (или локальный) адрес узла (МАС-адрес сетевого адаптера или порта маршрутизатора); эти адреса назначаются производителями сетевого оборудования;
* IP-адрес узла (например, 192.168.0.1), данные адреса назначаются сетевыми администраторами или Интернет-провайдерами;
* символьное имя (например, www.microsoft.com); эти имена также назначаются сетевыми администраторами компаний или Интернет-провайдерами.

выделяют 3 типа IP-адресов:

1. *Unicast*-адрес (единичная адресация конкретному узлу) — используется в коммуникациях "один-к-одному".
2. Broadcast-адрес (широковещательный адрес, относящийся ко всем адресам подсети). В этих адресах поле *идентификатора устройства* заполнено единицами.
3. Multicast-адрес (групповой адрес для многоадресной отправки пакетов) — используется в коммуникациях "один-ко-многим". Для групповой передачи рабочие станции и маршрутизаторы используют протокол IGMP, который предоставляет информацию о принадлежности устройств определенным группам.

128.10.2.30 - традиционная десятичная форма представления адреса,

10000000 00001010 00000010 00011110 - двоичная форма представления этого же адреса.

На рисунке 3.1 показана структура IP-адреса.

Класс А **0 N Сети N узла**

Класс В **1 0 N сети N узла**

Класс С **110 N сети N узла**

Класс D **1110 адрес группы multicast**

Класс Е **11110 зарезервирован**

Адрес состоит из двух логических частей - **номера сети** и номера **узла в сети**. Какая часть адреса относится к номеру сети, а какая к номеру узла, определяется значениями первых битов адреса:

* Если с 0, то сеть относят к классу А, и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса А имеют номера в диапазоне от 1 до 126
* Если первые два равны 10, то сеть относится к классу В и является сетью средних размеров с числом узлов 28 - 216. В сетях класса В под адрес сети и под адрес узла отводится по 16 битов, то есть по 2 байта.
* Если 110, то это сеть класса С с числом узлов не больше 28. Под адрес сети отводится 24 бита, а под адрес узла - 8 битов.
* Если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес - multicast. Если в пакете в качестве адреса назначения указан адрес класса D, то такой пакет должны получить все узлы, которым присвоен данный адрес.
* Если адрес начинается с последовательности 11110, то это адрес класса Е, он зарезервирован для будущих применений.

За отображение IP-адресов адаптеров на их физические адреса отвечает протокол ARP (Address Resolution Protocol. Р**азрешением адресов -** соответствие между физическим адресом устройства и его IP-адресом. Протокол ARP предполагает, что каждое устройство знает как свой IP-адрес, так и свой физический адрес. ARP динамически связывает их и заносит в специальную таблицу, где хранятся пары "IP-адрес — физический адрес" (обычно каждая запись в ARP-таблице имеет время жизни 10 мин.). Эта таблица хранится в памяти компьютера и называется кэш протокола ARP (ARP-cache).

37. Служба RRAS;

Служба *RRAS* (*Routing and Remote Access Service*, *Служба Маршрутизации и Удаленного Доступа* ) — служба системы Windows Server, позволяющая решать следующие задачи:

* подключение мобильных (или домашних) пользователей к корпоративной сети через коммутируемые телефонные линии и другие средства коммуникаций (например, сети Frame Relay, X.25);
* подключение к сети главного офиса компании удаленных офисов (через телефонные линии и сети Frame Relay, X.25);
* организация защищенных соединений (виртуальные частные сети) между мобильными пользователями, подключенными к сетям общего пользования (например, Интернет), и корпоративной сетью;
* организация защищенных соединений между офисами компании, подключенными к сетям общего пользования;
* маршрутизация сетевого трафика между различными подсетями корпоративной сети, соединенными как с помощью технологий локальных сетей, так и с помощью различных средств удаленных коммуникаций (например, по коммутируемым телефонным линиям).

Службы удаленного доступа, реализованные различными производителями, используют два основных коммуникационных протокола, которые образуют уровень, промежуточный между средствами коммуникаций (коммутируемые телефонные линии, Frame Relay, X.25) и сетевыми протоколами (TCP/IP, IPX/SPX):

* протокол *SLIP* ( *Serial Line Interface Protocol* ) — достаточно старый протокол, реализованный преимущественно на серверах удаленного доступа, функционирующих в системах семейства UNIX (разработан специально для подключения пользователей к сети Интернет); системы семейства Windows поддерживают данный протокол только на клиентской части (SLIP позволяет работать только с сетевым стеком TCP/IP, требует написания специальных сценариев для подключения клиента к серверу, не позволяет создавать виртуальные частные сети);
* протокол *PPP* ( *Point-to-Point Protocol* ) — используемый повсеместно коммуникационный протокол (точнее, семейство протоколов), позволяющий пользователям прозрачно подключаться к серверу удаленного доступа, использовать различные сетевые протоколы (TCP/IP, IPX/SPX, *NetBEUI*, AplleTalk), создавать виртуальные частные сети (служба удаленного доступа серверов Windows использует именно этот коммуникационный протокол).

Начнем с примера, показывающего процесс установки начальной настройки службы, и обсудим терминологию, технологии, а также все необходимые нам параметры, функции и возможности данной службы.

40.Диагностические утилиты TCP/IP и DNS.

|  |  |
| --- | --- |
| **Название утилиты** | **Комментарии** |
| ipconfig | Служит для отображения всех текущих параметров сети TCP/IP и обновления параметров DHCP и DNS. |
| arp | Отображение и изменение ARP-таблиц. |
| ping | Команда ping позволяет проверить:   * работоспособность IP-соединения; * правильность настройки протокола TCP/IP на узле; * работоспособность маршрутизаторов; * работоспособность системы разрешения имен FQDN или NetBIOS; * доступность и работоспособность какого-либо сетевого ресурса. |
| tracert | Служебная программа, используемая для определения пути, по которому IP-дейтаграмма доставляется по месту назначения. |
| pathping | Средство трассировки маршрута, сочетающее функции программ ping иt racert и обладающее дополнительными возможностями.  Эта команда показывает степень потери пакетов на любом маршрутизаторе или канале, с ее помощью легко определить, какие маршрутизаторы или каналы вызывают неполадки в работе сети. |
| netstat | Используется для отображения статистики протокола и текущих TCP/IP-соединений. |
| nbtstat | Средство диагностики разрешения имен NetBIOS |

35. Протокол NetBEUI. Стек протоколов IPX/SPX;

Протокол *NetBEUI* - протокол для использования в сетях, где не более 200 рабочих станций. Этот протокол содержит много полезных сетевых функций, которые можно отнести к сетевому, транспортному и сеансовому уровням модели OSI, однако с его помощью невозможна маршрутизация пакетов.

**Стек протоколов IPX/SPX**

Протоколы сетевого и сеансового уровня *Internetwork* Packet Exchange (IPX) и Sequenced Packet Exchange (SPX).

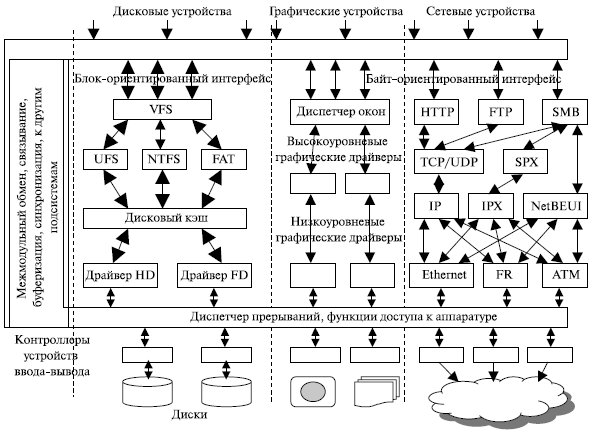
Многие особенности стека IPX/SPX обусловлены ориентацией ранних версий ОС NetWare (до версии 4.0) на работу в локальных сетях небольших размеров, состоящих из персональных компьютеров со скромными ресурсами. Понятно, что для таких компьютеров Novell нужны были протоколы, на реализацию которых требовалось бы минимальное количество оперативной памяти и которые бы быстро работали на процессорах небольшой вычислительной мощности. В результате протоколы стека IPX/SPX до недавнего времени хорошо работали в локальных сетях и не очень — в больших корпоративных сетях, так как они слишком перегружали медленные *глобальные связи* широковещательными пакетами, которые интенсивно используются несколькими протоколами этого стека (например, для установления связи между клиентами и серверами). Это обстоятельство, а также тот факт, что стек IPX/SPX является собственностью фирмы Novell, и на его реализацию нужно получать у нее лицензию, долгое время ограничивали распространенность его только сетями NetWare. С момента выпуска версии NetWare 4.0 Novell внесла и продолжает вносить в свои протоколы серьезные изменения, направленные на приспособление их для работы в корпоративных сетях. Сейчас стек IPX/SPX реализован не только в NetWare, но и в нескольких других популярных сетевых ОС, например, Microsoft Windows NT. Начиная с версии 5.0 фирма Novell в качестве основного протокола своей серверной операционной системы стала использовать протокол TCP/IP, и с тех пор практическое применение IPX/SPX стало неуклонно снижаться.

Как уже говорилось выше, стек протоколов IPX/SPX является фирменным запатентованным стеком компании Novell. Реализация данного протокола в операционных системах Microsoft называется NWLink (или IPX/SPX-совместимый протокол).

В серверных системах Windows (до Windows 2000 включительно) имелась также служба под названием " *Шлюз для сетей NetWare* ", позволявшая клиентам сетей Microsoft без установки клиента сетей NetWare получать доступ к ресурсам серверов под управлением Novell NetWare (через шлюз, установленный на сервере Windows NT/2000). В системе Windows 2003 служба шлюза отсутствует.

47.Многослойная (иерархическая) модель подсистемы ввода-вывода;

В каждой вертикальной подсистеме – несколько слоев модулей. Нижний слой образует **аппаратные драйверы**, управляющие аппаратурой внешних устройств, осуществляя обмен байтами и блоками байтов.



Наряду с модулями, отражающими специфику внешних устройств, в подсистеме ввода-вывода имеются **модули универсального назначения**. Эти модули организуют согласованную работу всех остальных компонентов подсистемы ввода-вывода, взаимодействие с пользовательскими процессами и другими подсистемами ОС. Эти организующие функции распределяются по всем уровням, образуя оболочку, называемую **менеджером ввода-вывода**.

Верхний слой менеджера составляют **системные вызовы ввода-вывода**, которые принимают от пользовательских **процессов** **запросы** на ввод-вывод и **переадресуют** их отвечающим за определенный класс устройств **модулям** и **драйверам**, а также возвращают процессам результаты операций ввода-вывода. Таким образом, этот слой поддерживает пользовательский интерфейс ввода-вывода, создавая для прикладных программистов максимум удобств по манипулированию внешними устройствами и расположенными на них данными.

Нижний слой менеджера реализует непосредственное взаимодействие с контроллерами внешних устройств, экранируя драйверы от особенностей аппаратной платформы компьютера – шин ввода-вывода, системы прерываний и т.п. Этот слой принимает от драйверов запросы на обмен данными с регистрами контроллеров в некоторой обобщенной форме с использованием независимых от шины ввода-вывода адресации и формата, а затем преобразует эти запросы в зависящий от аппаратной платформы формат.

Диспетчер прерываний может входить в состав менеджера ввода-вывода или представлять отдельный модуль ядра. В последнем случае менеджер ввода-вывода выполняет для диспетчера прерываний первичную обработку запросов прерываний, передавая диспетчеру обобщения сведения об источнике запроса.

**Функцией**: создание стандартного внутреннего интерфейса взаимодействия модулей ввода-вывода между собой. Это облегчает включение новых драйверов и файловых систем в состав ОС.

**Функцией**: организация взаимодействия модулей ввода-вывода с модулями других подсистем ОС, таких как подсистема управления процессами, виртуальной памятью и другими.

41.Назначение, задачи и технологии подсистемы ввода-вывода;

**Подсистема ввода-вывода** – система которая выполняет обмен данными между пользователями, приложениями и периферийными устройствами компьютера.

**Основными компонентами** подсистемы ввода-вывода являются **драйверы**, управляющие внешними устройствами, и **файловая система**. В работе подсистемы ввода-вывода активно участвует **диспетчер прерываний**.

**Файловая система** – это основное хранилище информации. Она активно использует остальные части подсистемы ввода-вывода.

На подсистему ввода-вывода возлагаются следующие функции:

* **организация** параллельной работы устройств ввода-вывода и процессора;
* **согласование** скоростей обмена и кэширование данных;
* **разделение** устройств и данных между процессами (выполняющимися программами);
* **обеспечение** удобного логического интерфейса между устройствами и остальной частью системы;
* **поддержка** широкого спектра драйверов с возможностью простого включения в систему нового драйвера;
* **динамическая** загрузка и выгрузка драйверов без дополнительных действий с операционной системой;
* **поддержка** нескольких различных файловых систем;
* **поддержка** синхронных и асинхронных операций ввода-вывода.

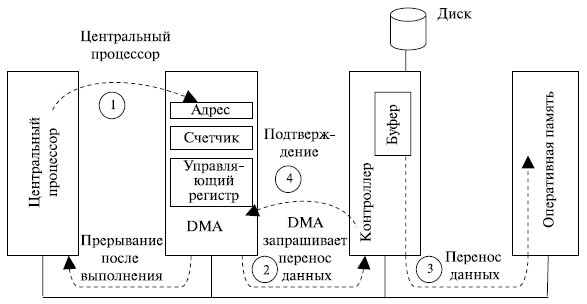
Для персональных компьютеров операции ввода-вывода могут выполняться **3 способами**.

1. **С помощью программируемого ввода-вывода**. Процессору выполняет команду ввода-вывода, посылая соответствующие команды контроллеру ввода-вывода. Это устройство выполняет требуемое действие, а затем устанавливает соответствующие биты в регистрах состояния ввода-вывода и не посылает никаких сигналов, в том числе сигналов прерываний. Процессор периодически проверяет состояние модуля ввода-вывода с целью проверки завершения операции ввода-вывода.
2. **Ввод-вывод, управляемый прерываниями**. Процессор посылает необходимые команды контроллеру ввода-вывода и продолжает выполнять текущий процесс. Наличие прерываний процессор проверяет в конце каждого цикла выполняемых команд.
3. **Прямой доступ к памяти** (DMA). Управляет обменом данных между основной памятью и контроллером ввода-вывода. Процессор посылает запрос на передачу блока данных модулю прямого доступа к памяти, а прерывание происходит только после передачи всего блока данных.

Контроллер содержит несколько регистров, доступных центральному процессу для чтения и записи (**регистр адреса памяти, счетчик байтов, управляющие регистры**).

Перед выполнением операции обмена ЦП программирует DMA-контроллер, устанавливая его регистры (шаг 1 на рис. 7.1). Затем ЦП дает команду дисковому контролеру прочитать внести данные во внутренний буфер и проверить контрольную сумму. После этого процессор продолжает свою работу. Когда данные получены и проверены контроллером диска, DMA может начинать работу.

DMA-контроллер начинает перенос данных, посылая дисковому контроллеру по шине запрос чтения (шаг 2). Адрес памяти уже находится на адресной шине, так что контроллер знает, куда пересылать следующее слово из своего буфера. Запись в память является еще одним стандартным циклом шины (шаг 3). Когда запись закончена, контроллер диска посылает сигнал подтверждения контролеру DMA (шаг 4). Затем контроллер DMA увеличивает используемый адрес памяти и уменьшает значение счетчика байтов. После этого шаги 2, 3 и 4 повторяются, пока значение счетчика не станет равным нулю. По завершению цикла копирования контроллер DMA инициирует прерывание процессора, сообщая ему о завершении операции ввода-вывода.



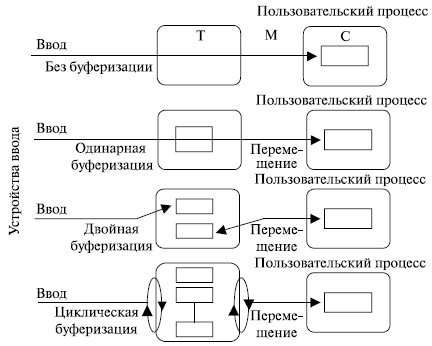
42.Согласование скоростей обмена и кэширования данных;

Согласование достигается **буферизацией данных**. В подсистеме ввода-вывода часто используется буферизация в оперативной памяти. В этих случаях часто используют в качестве буфера дисковый файл, называемый **спул-файлом**. Типичный пример применения спулинга – вывод данных на принтер.

Другим решением проблемы является использование большой буферной памяти в контроллерах внешних устройств.

Существует два типа устройств – **блочные** и **символьные**. Первые сохраняют информацию блоками фиксированного размера и передают поблочно (диски, ленты). Вторые выполняют передачу в виде неструктурированных потоков байтов (терминалы, принтеры, манипулятор мыши, сканеры и др.).

Возможные схемы буферизации ввода-вывода приведены на рис. 7.3.



**Одинарный буфер**. В тот момент, когда пользовательский процесс выполняет запрос ввода-вывода, операционная система назначает ему буфер в системной части оперативной памяти. Осуществляется передача входных данных в системный буфер, процесс перемещает блок в пользовательское пространство и немедленно производит запрос следующего блока. Такая процедура называется **опережающим считыванием** вводом.

**Использования двух буферов**. Теперь процесс выполняет передачу данных в один буфер (или считывает из него), в то время как ОС освобождает (или заполняет) другой. Эта технология известна как двойная буферизация.

Буферизация данных позволяет не только согласовать скорости работы процессора и внешних устройств, но и сократить количество реальных операций ввода-вывода за счет кэширования данных.

43.Разделение устройств и данных между процессами;

ОС должна обеспечивать контроль доступа путем проверки прав пользователя или группы пользователей.

Устройства ввода-вывода могут предоставляться процессам:

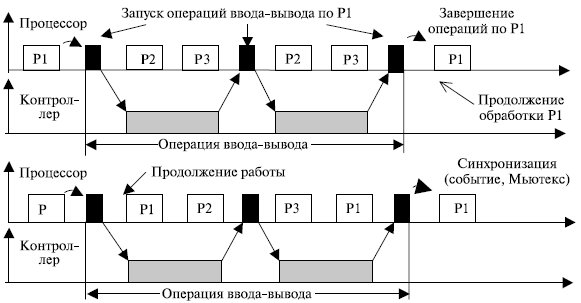
* 1. в монопольном (последовательные порты и алфавитно-цифровые терминалы). ОС должна оптимизировать последовательность операций ввода-вывода для различных процессов
  2. разделенном режиме (диск). возникнуть необходимость в разграничении данных процессов друг от друга. Пример – принтер. организуется очередь заданий на вывод, при этом каждое задание представляет собой порцию данных, которую нельзя разрывать

Для хранения очереди заданий используется **спул-файл**, который согласует скорость работы принтера и оперативной памяти и позволяет организовать разбиение данных на логические порции. Процессы могут одновременно выполнять вывод на принтер, помещая данные в свой раздел спул-файла.

46.Поддержка синхронных и асинхронных операций ввода-вывода;

**Синхронный режим**: программный модуль приостанавливает свою работу до тех пор, пока операция ввода-вывода не будет завершена.

**Асинхронный режим:** программный модуль продолжает выполняться в мультипрограммном режиме одновременно с операцией ввода-вывода.



Отличие заключается в том, что операция ввода-вывода может быть инициирована не только пользовательским процессом – в этом случае операция выполняется в рамках системного вызова, – но и кодом ядра, например, кодом подсистемы виртуальной памяти для считывания отсутствующей страницы.

Системы вызовы ввода-вывода чаще оформляются как синхронные процедуры в связи с тем, что такие операции длятся долго и пользовательскому процессу или потоку все равно придется ждать получения результатов потоков операции, для того чтобы продолжить свою работу.

Внутренние вызовы операций ввода-вывода из модулей ядра обычно выполняются в виде асинхронных процедур, так как кодам ядра нужна свобода в выборе дальнейшего поведения после запроса ввода-вывода.

48.Драйверы;

**Драйвер** - программный модуль, который:

* входит в состав ядра ОС, работая в привилегированном режиме;
* управляет внешним устройством, взаимодействуя с его контроллером с помощью команд ввода-вывода компьютера;
* обрабатывает прерывания от контроллера *устройства*;
* предоставляет программисту логический интерфейс работы с устройством, экранируя от него низкоуровневые детали управления устройством и организации его данных;
* взаимодействует с другими модулями ядра ОС с помощью строго оговоренного интерфейса, описывающего формат передаваемых данных, структуру буферов, способы включения драйвера в состав ОС, способы вызова драйвера, набор общих процедур подсистемы ввода-вывода, которыми драйвер может пользоваться и т.п.

Драйвер преобразует данные в линейную последовательность блоков. В результате прикладная программа работает с файлами, таблицами баз данных, текстовыми окнами на мониторе и т.п.

**Высокоуровневые** драйверы, которые располагаются в общей модели подсистемы ввода-вывода над традиционными драйверами. Появление таких драйверов можно считать развитием идеи многоуровневой организации подсистемы ввода-вывода, когда ее функции декомпозируются между несколькими модулями в соседних слоях иерархии (таких примеров много, например, семиуровневая модель сетевых протоколов).

**Традиционные (**низкоуровневыми) драйверы освобождаются от высокоуровневых функций и занимаются только низкоуровневыми операциями. Эти низкоуровневые операции составляют фундамент, на котором можно построить тот или иной набор операций в драйверах более высоких уровней.

Несколько драйверов, управляющих одним устройством, но на разных уровнях, можно рассматривать как один многоуровневый драйвер.

На практике используют **от двух до пяти** уровней драйверов, поскольку с увеличением числа уровней снижается скорость выполнения операций ввода-вывода.

Драйвер обычно имеет определенную структуру, в которой выделяется **секция обработки прерываний** (*Interrupt Service* Routine, *ISR*), которая и вызывается от соответствующего устройства диспетчером прерываний.

В унификацию драйверов большой вклад внесла ОС UNIX, в которой все драйверы были разделены на два класса: **блок-ориентированные** и **байт-ориентированные** драйверы. Например, драйверы графических устройств и сетевых устройств относятся к классу байт-ориентированных.

**Блок-ориентированные драйверы** управляют устройствами прямого доступа, которые хранят информацию в блоках фиксированного размера, каждый из которых имеет свой адрес. Адресуемость блоков приводит к тому, что для дисков, являющихся устройствами прямого доступа, появляется возможность кэширования данных в оперативной памяти.

Устройства, с которыми работают **байт-ориентированные драйверы**, не адресуют данные и не позволяют производить операции поиска данных, они генерируют или потребляют последовательность байта (терминалы, принтеры, сетевые адаптеры и т.п.).

45.Поддержка широкого спектра драйверов;

Драйвер взаимодействует, с одной стороны, с модулями ядра ОС (модулями подсистемы ввода-вывода, модулями системных вызовов, модулями подсистем управления процессами и памятью), а с другой стороны – с контроллерами внешних устройств. Поэтому существует **два вида интерфейсов**: интерфейс "драйвер-ядро" (DKI) и интерфейс "драйвер-устройство" (Driver Device Interface).

Интерфейс "драйвер-ядро" должен быть стандартизован в любом случае. Подсистема ввода-вывода может поддерживать несколько различных интерфейсов DKI/DDI, предоставляя специфический интерфейс для устройств определенного класса. К наиболее общим классам относятся *блочные устройства*, например, диски, и *символьные устройства*, такие как клавиатура и принтеры. Может существовать класс сетевых адаптеров и др. В большинстве современных ОС определен стандартный интерфейс, который должен поддерживать все блочные драйверы, и второй стандартный интерфейс, поддерживаемый всеми символьными адаптерами. Эти интерфейсы включают наборы процедур, которые могут вызываться остальной операционной системой для обращения к драйверу. К этим процедурам относятся, например, процедуры чтения блока или записи символьной строки.

Кроме того, подсистема ввода-вывода поддерживает большое количество системных функций, которые драйвер может вызывать для выполнения некоторых типовых действий. Например, это операции обмена с регистрами контроллера, ведения буферов промежуточного хранения данных ввода-вывода, взаимодействия с DMA-контроллером и контроллером прерываний и др.

У драйверов устройств есть множество функций, перечисленных ниже [17].

1. Обработка запросов записи-чтения от программного обеспечения управления устройствами. *Постановка запросов в очередь*.
2. Проверка входных параметров запросов и обработка ошибок.
3. Инициализация устройства и проверка статуса устройства.
4. Управление энергопотреблением устройства.
5. Регистрация событий в устройстве.
6. Выдача команд устройству и ожидание их выполнения, возможно, в блокированном состоянии, до поступления прерывания от устройства.
7. Проверка правильности завершения операции.
8. Передача запрошенных данных и статуса завершенной операции.
9. Обработка нового запроса при незавершенном предыдущем запросе (для реентерабельных драйверов).

Наиболее очевидная функция состоит в обработке абстрактных запросов чтения и записи независимого от устройств программного обеспечения, расположенного над ними. Но, кроме этого, они должны выполнять еще несколько функций. Например, драйвер должен при необходимости инициализировать устройство. Ему может понадобиться управлять энергопотреблением устройства и регистрацией событий.

Многие драйверы обладают сходной общей структурой. Типичный драйвер начинает работу с проверки входных параметров. Если они не удовлетворяют определенным критериям, драйвер возвращает ошибку. В противном случае драйвер преобразует абстрактные термины в конкретные. Например, дисковый драйвер преобразует линейный номер кластера в номер головки, дорожки и сектора.

Затем драйвер может проверить, не используется ли это устройство в данный момент. Если устройство занято, запрос может быть поставлен в очередь. Если устройство свободно, проверяется статус устройства, чтобы понять, может ли запрос быть обслужен прямо сейчас. Может оказаться необходимым включить устройство или запустить двигатель, прежде чем начнется перенос данных. Как только устройство включено и готово, начинается собственно управление устройством.

Управление устройством подразумевает выдачу ему серии команд. Именно в драйвере определяется последовательность команд в зависимости от того, что должно быть сделано. Определившись с командой, драйвер начинает записывать их в регистры *контроллера устройства*. После записи каждой команды в контроллер, возможно, будет нужно проверить, принял ли контроллер команду и готов ли принять следующую. Такая последовательность действий продолжается до тех пор, пока контроллеру не будут переданы все команды. Некоторые контроллеры способны принимать связные списки команд, находящихся в памяти. Они сами считывают и выполняют их без дальнейшей помощи ОС.

После того как драйвер передал все команды контроллеру, ситуация может развиваться по двум сценариям. Во многих случаях драйвер устройства должен ждать, пока контроллер не выполнит для него определенную работу, поэтому он блокируется до тех пор, пока прерывание от устройства не разблокирует его. В других случаях операция завершается без задержек и драйверу не нужно блокироваться. В любом случае по завершении выполнения операции драйвер должен проверить, завершилась ли операция без ошибок. Если все в порядке, драйверу, возможно, придется передать данные (например, только что прочитанный блок) независимому от устройств программному обеспечению. Наконец, драйвер возвращает информацию о состоянии для информирования вызывающей программы о *статусе завершения* операции. Если в очереди находились другие запросы, один из них теперь может быть выбран и запущен. В противном случае драйвер блокируется в ожидании следующего запроса.

Для поддержки процесса разработки драйверов операционной системы выпускается так называемый пакет *DDK* (Driver Development Kit), представляющий собой набор инструментальных средств-библиотек, компиляторов и отладчиков.

44.Обеспечение логического интерфейса между устройствами и системой. Динамическая загрузка и выгрузка драйверов;

Любое устройство выглядит для прикладного программиста последовательным набором байт, с которым можно работать с помощью унифицированных системных вызовов (например, read, write), задавая имя файла-устройства и смещение от начала последовательности байт.

Ценным свойством ОС является возможность динамически загружать в оперативную *память* требуемый *драйвер* (без остановки ОС) и выгружать его. Такое свойство ОС может существенно сэкономить *системную область* памяти.

Альтернативой динамической загрузке драйверов при изменении текущей конфигурации внешних устройств компьютера является повторная **компиляция** кода ядра с требуемым набором драйверов, что создает между всеми компонентами ядра статические связи вместо динамических. Например, таким образом решалась данная проблема в ранних версиях ОС *UNIX*.

49.Файловые системы. Основные понятия;

Цели и задачи файловой системы

Необходимо хранить данные с соблюдением следующих требований.

1. Устройства должны позволять хранить очень большие объемы данных.
2. Информация должна длительно и надежно сохраняться после прекращения работы процесса, =
3. Должно быть обеспечено совместное использование данных.

**Файл** – это именованная совокупность данных, хранящаяся на каком-либо носителе информации.

При рассмотрении отдельных файлов и их совокупностей используются следующие понятия.

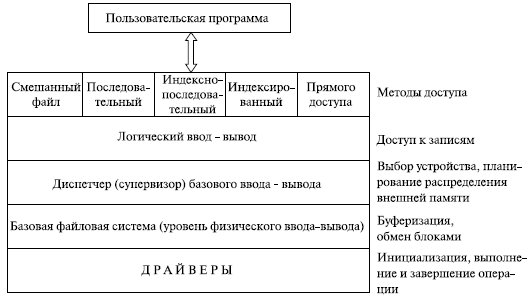
1. **Поле** – основной элемент данных. Поле содержит единственное значение, такое как имя служащего, дату, значение некоторого показателя и т.п. Поле характеризуется длиной и типом данных и может быть фиксированной или переменной длины, т.е. состоять из нескольких подполей: имя поля, значение, длина поля.
2. **Запись** – набор связанных между собой полей, которые могут быть обработаны как единое целое некоторой прикладной программой.
3. **Файл** – совокупность однородных записей.
4. **База данных** – набор связанных между собой данных, представленных совокупностью файлов одного или несколько типов. Обычно существует отдельная система управления базой данных (СУБД), независимая от операционной системы, но, тем не менее, она почти всегда использует некоторые программы управления файлами.

**Файловая система** – это часть операционной системы, включающая:

* совокупность всех файлов на носителе информации
* наборы структур данных, используемых для управления файлами, каталоги и дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске и др.);
* комплекс системных программных средств, реализующих различные операции над файлами (создание, уничтожение, чтение, запись и др.).

50.Архитектура файловой системы;

Файловая система играет роль промежуточного слоя, экранизирующего все сложности физической организации.



53.Физическая организация файловой системы;

Файл имеет образ цельного набора байт, но на самом деле разбросан своими частями по всему диску.

**Цилиндр -** совокупность дорожек одного радиуса на всех поверхностях пластин пакета. Каждая дорожка разбивается на фрагменты, называемые **секторами** или **блоками**, так что все дорожки имеют равное число секторов, в которые можно максимально записать одно и то же число байт. Сектор имеет фиксированный для данной системы размер, выражающийся степенью двойки (чаще всего 512 байт).

**Сектор** – наименьшая адресуемая единица обмена данными диска с оперативной памятью. Для того чтобы контроллер мог найти на диске нужный сектор, ему необходимо задать все составляющие адреса сектора: **номер цилиндра**, **номер поверхности** и **номер сектора**. Типичный запрос включает чтение (запись) нескольких секторов, содержащих наряду с требуемыми избыточные данные.

**Кластер** - единица дискового пространства, содержащая несколько секторов в числе, кратном степени двойки.

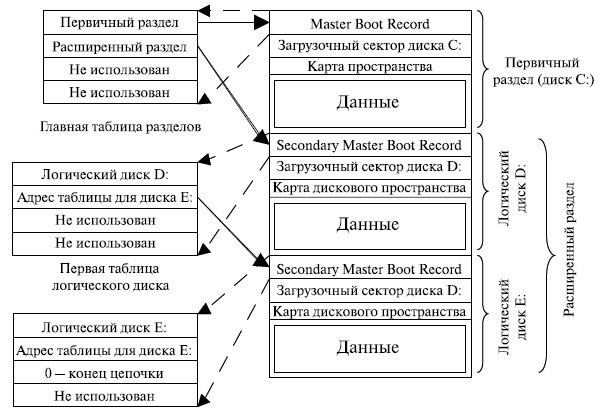
Дорожки и секторы создаются в результате выполнения процедуры низкоуровнего форматирования диска. Для определения *границ блоков* на диск записывается идентификационная информация.

Разметку диска под конкретный тип файловой системы выполняют процедуры **высокоуровнего** форматирования. При высокоуровневом форматировании определяется *размер кластера*, записываются информация, необходимая для работы файловой системы, и загрузчик ОС – небольшая программа, которая начинает процесс инициализации операционной системы после включения питания.

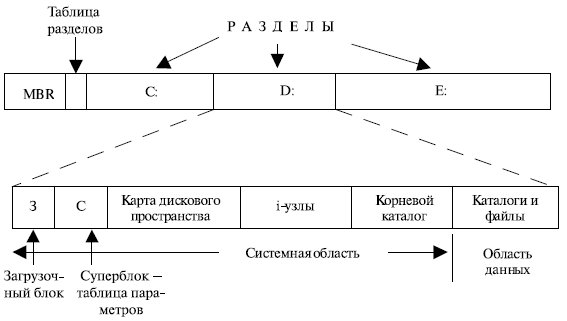
Прежде чем форматировать диск под определенную файловую систему, он может быть разбит на разделы. **Раздел** – это непрерывная часть физического диска, которую операционная система представляет пользователю как логическое устройство (логический диск или логический раздел). На каждом разделе может создаваться только одна файловая система.

**Главная загрузочная записью** (*MBR* – *Master Boot Record*) - сектор 1 диска, используется для загрузки компьютера. В конце *MBR* содержится **таблица разделов**. В ней хранятся начальные и конечные адреса (номера блоков) каждого раздела. Считывание и исполнение MBR, загрузчик в *MBR*-записи определяет активный раздел диска, считывает его первый блок (загрузчик) и исполняет его. Программа, находящаяся в загрузочном блоке, загружает операционную систему, содержащуюся в этом разделе.

Разделы дисков могут быть двух типов: **первичные** и **расширенные**. Максимальное число первичных разделов равно 4. Из них только один может быть активным. Согласно принятым спецификациям на одном жестком диске может быть только один ***расширенный*** *раздел*, который может быть разделен на логические диски. ***Расширенный раздел*** содержит вторичную запись *MBR*, в состав которой вместо таблицы разделов входит аналогичная ей таблица логических дисков (logical Disks Table, *LDT*). Эта таблица описывает размещение и характеристики раздела, содержащего единственный логический диск, а также может специфицировать следующую запись SMBR (Secondary *MBR*).



**Суперблок** – раздел который содержит ключевые параметры файловой системы и считывается в память при загрузке компьютера. Следом располагается информация о свободных блоках файловой системы. За этими данными может следовать информация об i-узлах, содержащих информацию о файлах. Следом может размещаться каталог и затем – остальные файлы и каталоги.

  
**Рис. 7.14.**  Структура раздела

На разных логических устройствах одного и того же физического диска могут располагаться файловые системы разного типа. Все разделы одного диска имеют одинаковый размер блока, определенный для данного диска в результате низкого уровневого форматирования. Однако в результате высокоуровневого форматирования в разных разделах одного и того же диска могут быть установлены различные файловые системы с различными разделами кластеров.

51.Организация файлов и доступ к ним;

Типов файлов:

* 1. обычные файлы, содержащие информацию произвольного характера (текст, графика, звук и др.). Информацию заносит пользователь или программы
  2. файлы-каталоги. это системные файлы, обеспечивающие поддержку структуры файловой системы. Они содержат системную справочную информацию о наборе файлов, сгруппированных пользователем по какому-либо неформальному признаку (договоры, рефераты, курсовые проекты и т.п.). Каталоги устанавливают соответствие между именами файлов и их характеристиками. В число таких характеристик входят тип файла, права доступа к файлу, его распоряжение на диске, размер, дата и время создания и др.
  3. специальные файлы. Специальные файлы – это фиктивные файлы, ассоциированные с устройствами ввода-вывода, которые используются для унификации механизма доступа к последовательным устройствам ввода-вывода, таким как терминалы, принтеры и др.
  4. Именованные конвейеры. представляют собой циклические буферы, позволяющие выходной файл одной программы соединить со входным файлом другой программы.
  5. отображаемые в память файлы и др - это обычные файлы, отображенные на адресное пространство процесса по указанному виртуальному адресу.

**Три типа имен файлов**: **простые, составные и относительные**.

**Файл** представляет собой набор однородных *записей*, которые могут быть обработаны как единое целое. Размер записи равен одному байту. Приложения оперируют записями, физический обмен с устройством ввода/вывода осуществляется блоками, поэтому записи объединяются в блоки.

1.**Последовательный файл** – *неструктурированная последовательность байтов*, обработка которых предполагает последовательное чтение записей от начала файла до конца. Способ последовательного чтения файла называется ***последовательным доступом***

2.**Файл прямого доступа** – файл, байты которого могут быть считаны в произвольном порядке. Содержимое файла может быть разбросано по разным блокам диска, которые можно считывать в произвольном порядке. Номер блока (относительный номер, специфицирующий блок) определяется позицией внутри файла. Для доступа к середине файла просмотр его с начала не обязателен.

3.В ранних операционных системах использовались другие формы организации файла и другие способы доступа к ним. В настоящее время они применяются в больших мэйнфреймах, ориентированных на коммерческую обработку данных.

**Способ 1**. Хранение файла в виде последовательности записей фиксированной длины, каждая из которых имеет внутреннюю структуру. Операция чтения производится над записью, а операция записи переписывает или добавляет запись целиком.

**Способ 2**. Хранение файла в виде последовательности записей переменной длины, каждая из которых содержит ключевое поле в фиксированной позиции внутри записи. Базисной операцией является считывание записи с каким-либо значением ключа. Записи могут располагаться в файле последовательно (например, отсортированные по значению ключевого поля) или в более сложном порядке. Метод доступа по значению ключевого поля к записям последовательного файла называется ***индексно-последовательным***.

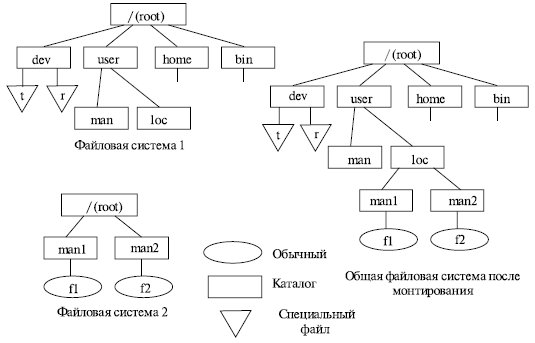
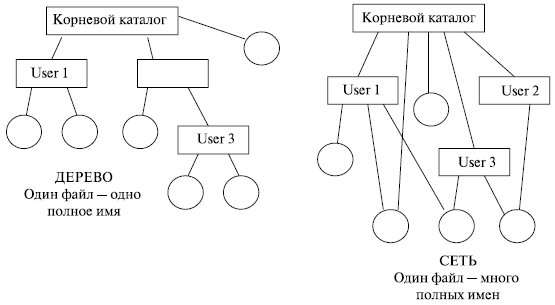
**Способ 3**. В некоторых системах ускорение доступа к файлу обеспечивает конструированием *индекса файла*. Индекс состоит из списка элементов, каждый из которых содержит идентификатор записи, за которым следует указание о местоположении данной записи. Для поиска записи вначале происходит обращение к индексу, где находится указатель на нужную запись. Такие файлы называются **индексированными**, а метод доступа к ним – *доступ с* **использованием индекса**.

Другим способом выделения дискового пространства при помощи индексных узлов (например,*Unix*) является **древовидная организация блоков**, при которой блоки, составляющие файл, являются листьями дерева, а каждыйвнутренний узел содержит указатели на множество блоков файла. Длябольших файлов индекс может оказаться громоздким, в этом случае создают индекс для индексного файла (блоки промежуточного уровня или блоки косвенной адресации).

52.Каталоговые системы;

**Файловый каталог** - служит связующим звеном между *системой управления файлами* и набором файлов. Простейшая форма системы каталогов состоит в том, что имеется один каталог, в котором содержатся все файлы. Каталог содержит информацию о файлах, включая атрибуты, местоположение, принадлежность. Пользователи обращаются к файлам по символьным именам. Иерархическая организация пространства имен.. Граф, описывающий иерархию каталогов, может быть деревом или сетью. Каталоги образуют дерево, если файлу разрешено входить только в один каталог, и сеть, если файл может входить в несколько каталогов.

Например, в Ms-Dos и Windows каталоги образуют древовидную структуру, а в UNIX – сетевую.



Первое решение состоит в том, что на каждом из устройств размещается автономная файловая система, т.е. файлы, находящиеся на этом устройстве, описываются деревом каталогов, никак не связанным с деревьями каталогов на других устройствах. В таком случае для однозначной идентификации файла пользователь вместе с составным символьным именем файла должен указывать идентификатор логического устройства. Примером такого автономного существования может служить MS-DOS, Windows 95/98/Me/XP.

**Монтирование -** такая организация хранения файлов, при которой пользователю предоставляется возможность объединить файловые системы, находящиеся на разных устройствах, в единую файловую систему, описываемую единым деревом каталогов.

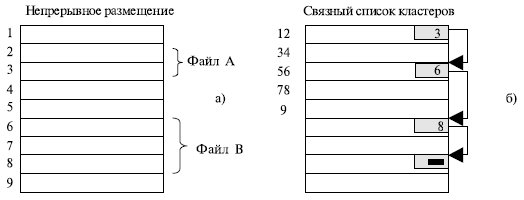
Файловая система, расположенная на системном диске, называется корневой. Для связи иерархий файлов в *корневой файловой системе* выбирается некоторый существующий каталог, в данном примере – каталог *loc*. После выполнения монтирования выбранный каталог *loc* становится корневым каталогом второй файловой системы. Через этот каталог монтируемая файловая система подсоединяется как поддерево к общему дереву.

54.Физическая организация и адресация файла;

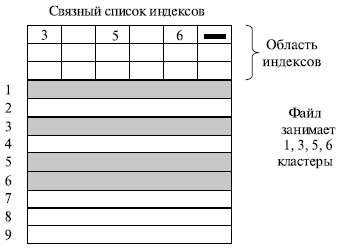
Физическая организация выделяет способ размещения файлов на диске и учет соответствия блоков диска файлам.

**Простейший вариант физической организации** – непрерывное размещение в наборе соседних кластеров. Достаточно хранить номер первого кластера и объем файла. Размер файла при такой организации не ограничивается.

**Второй** метод размещения файлов состоит в представлении файла **в виде связного списка кластеров** дисковой памяти. Первое слово каждого кластера используется как **указатель на следующий кластер**. Расположение файла задается номером его первого кластера.



Оба недостатка предыдущей схемы организации файлов могут быть устранены, если указатели на следующие кластеры хранить в отдельной таблице, загружаемой в память. Таким образом, образуется связный список не самих блоков (кластеров) файла, а индексов, указывающих на эти блоки.



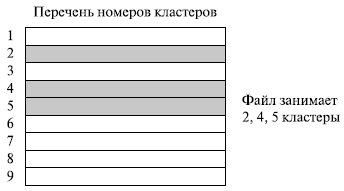
**Рис. 7.16.**  Вариант физической организации файлов

Такая таблица, называемая FAT-таблицей (File Allocation Table). Вообще индексы могут содержать следующую информацию о кластере диска (для FAT 32):

* не используется – 0000.0000;
* используется файлом – значение, отличное от 000.000, FFFF.FFFF и FFFF.FFF7;
* плохой кластер – FFFF.FFF7;
* последний кластер файла – FFFF.FFFF.

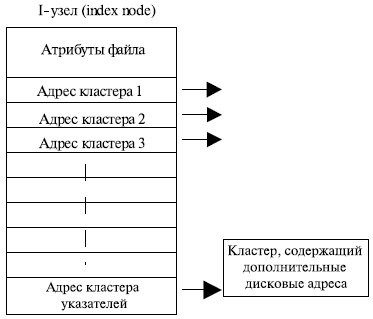
Для доступа к произвольному кластеру файла не требуется последовательно считывать его кластеры, достаточно прочитать FAT-таблицу, отсчитать нужное количество кластеров файла по цепочке и определить номера нужного кластера.

Еще один способ заключается в простом перечислении номеров кластеров, занимаемых файлом (рис. 7.17). Этот перечень и служит адресом файла. Недостаток такого подхода – длина адреса зависит от размера файла.



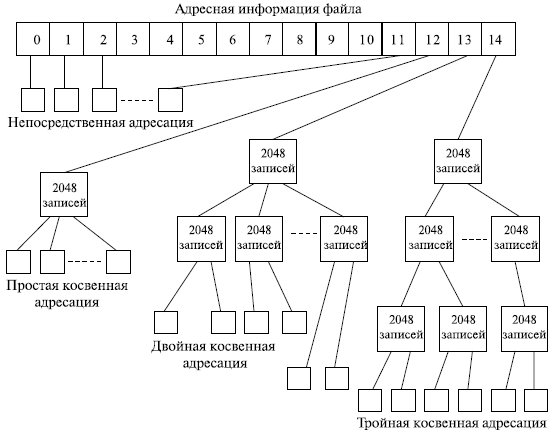
**Рис. 7.17.**  Вариант физической организации файлов

Эффективный метод организации файлов состоит в связывании с каждым файлом структуры данных, называемой i-узлами. Такой узел содержит атрибуты файла и адреса кластеров файла (рис. 7.18). Каждый конкретный i-узел должен находиться в памяти только тогда, когда открыт соответствующий ему файл.



**Рис. 7.18.**  вариант физической организации файлов

Сочетание прямой и косвенной адресации. Такой поход реализован в файловой системе *ufs*.



**Рис. 7.19.**  Файловая система ufs

Для хранения адреса файла выделено 15 полей, каждое из которых состоит из 4 байт. Если кластер имеет размер 8 Кбайт, то можно адресовать файл размеров до 8 Кбайт \* 12 = 98304 байт. Если *размер кластера* превышает 12 кластеров, то следующее 13 поле содержит адрес кластера, в котором могут быть расположены номера следующих кластеров, и размер файла может возрасти до 8192 \* (12 + 2048) = 16.875.520 байт.

Следующий уровень адресации, обеспечиваемый 14-м полем, позволяет адресовать до 8192 \* (12 + 2048 + 20482) = 3,43766\*1020 байт. Если и этого недостаточно, используется следующее 15-е поле. В этом случае максимальный размер файла может составить 8192 \* (12 + 2048 + 20482 + 20483) = 7,0403\*1013 байт.

При этом объеме самой адресной информации составит всего 0,05% от объема адресуемых данных (задачи!!!).

Метод перечисления адресов кластеров файла задействован и в файловой системе NTFS, применяемой в Windows NT/2000/2003. Для сокращения объема адресной информации в NTFS адресуются не кластеры файла, а непрерывные области, состоящие из смежных кластеров диска. Каждая такая область называется экстентом (extent) и описывается двумя числами: номером начального кластера и количеством кластеров.

55.Физическая организация FAT-системы;

Для обеспечения доступа приложений к файлам операционная система с файловой системой FAT использует следующие структуры:

* загрузочные секторы главного и *дополнительных разделов*;
* загрузочные секторы логических дисков (разделов);
* корневой каталог;
* область данных;
* цилиндр для выполнения диагностических операций чтения-записи.

Загрузочный сектор главного раздела (называемый главной загрузочной записью – *Master Boot Record* – *MBR*) является первым сектором на жестком диске (цилиндр 0, головка 0, сектор 1) и состоит из двух элементов [10]:

* таблица главного раздела, содержащая список разделов (максимум четыре) и расположение загрузочных секторов соответствующих логических дисков (первая и последняя головки, первый и последний цилиндры с соответствующими значениями секторов, а также количество секторов);
* главный загрузочный код – небольшая программа, которая выполняется системой BIOS. Основная функция этого кода – передача управления в раздел, который обозначен как активный (загрузочный).

Загрузочный сектор раздела содержит:

* блок параметров диска, в котором содержится информация о разделе (размер, количество секторов, *размер кластера*, метка тома и др.);
* загрузочный код – программу, с которой начинается процесс загрузки операционной.

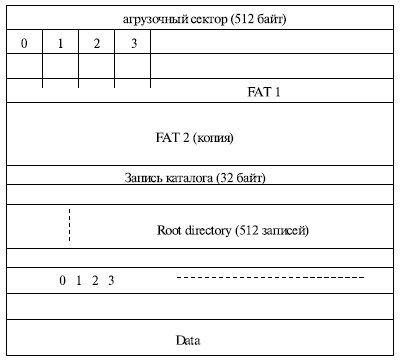
Загрузочные секторы логических дисков создаются программой Format. Они похожи на загрузочные диски разделов. Однако при загрузке выполняется код только того сектора, который находится в активном разделе.

Логический диск, отформатированный программой Fdisk, состоит из следующих областей (рис. 7.20):

* загрузочный сектор;
* основная FAT-таблица, содержащая информацию о размещении файлов и каталогов на диске;
* копия FAT-таблицы;
* корневой каталог – фиксированная область (16 Кбайт для жесткого диска), позволяющая хранить 512 записей о файлах и каталогах (каждая запись состоит из 32 байтов);
* область данных для размещения всех файлов и каталогов, кроме корневого каталога.

Первые две записи FAT зарезервированы и содержат информацию о самой FAT, все остальные указывают на соответствующие кластеры диска. Индексный указатель принимает значение, характеризующее состояние связанного с ним кластера (для FAT 16):

* кластер свободен (0000h);
* кластер используется (любое значение, кроме специальных);
* последний кластер файла (FFF8h – FFFFh);
* кластер поврежден (FFF7h);
* резервный кластер (FFF6h).



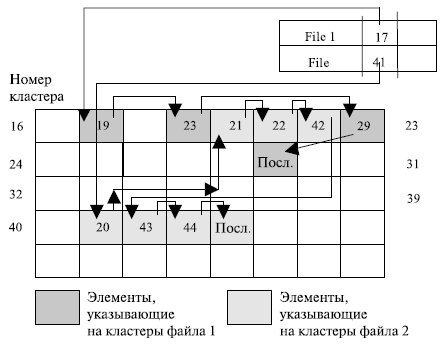
**Рис. 7.20.**  FAT-система

Размер FAT-таблицы определяется количеством кластера. Разрядность индексного указателя FAT-таблицы должна быть такой, чтобы можно было задать максимальный номер кластера диска определенного объема. В соответствии с разрядностью дискового указателя существуют несколько разновидностей FAT: FAT12, *FAT16*, FAT32 (соответственно 212, 216 и 232 кластеров). Тип используемой FAT определяется программой Fdisk, хотя и записываются они в процессе форматирования высокого уровня программы Format. На всех дискетах применяется FAT 12, на жестких дисках до 512 Мбайт – *FAT16*, на жестких дисках, имеющих большую емкость при использовании Windows 95 OSR2 и Windows98 – FAT 32 (вообще *размер кластера* может быть от 1 до 128 секторов или от 512 байт до 64 Кбайт). Максимальный размер раздела *FAT16* ограничен объемом 4 Гбайт (216 = 65536 кластеров по 64 Кбайт). Максимальный размер раздела FAT 32 практически не ограничен (232 кластеров по 32 Кбайт).

За копией FAT-таблицы следует корневой каталог – база данных, содержащая информацию о записанных на диске данных. Каждая запись в ней имеет длину 32 байта и содержит всю информацию о файле, которой располагает операционная система. Формат записи приведен ниже.

Информация о расположении файла, то есть о расположении оставшихся кластеров, содержится в FAT-таблице. В процессе работы системы кластеры файла могут оказаться не в смежных областях, а будут чередоваться с кластерами других файлов. Однако эту цепочку кластеров легко выделить, зная начальный кластер файлов. На рис. 7.21 показан пример размещения двух файлов.

В корневом каталоге имеются записи не только о файлах, но и подкаталогах. Эти записи имеют точно такую же структуру, что и записи корневого каталога. Признак подкаталогов указывается в атрибутах файла, т.е. можно считать, что подкаталог – это специальный файл. Структура *атрибутивного* байта показана ниже.



**Рис. 7.21.**  Пример размещения двух файлов

Файловые системы FAT 12 и *FAT16* оперируют с именами файлов, составленных по схеме 8.3 (имя, расширение). В Windows 95 с появлением 32-разрядной виртуальной FAT-*VFAT* (Virtual file allocation table) поддерживаются имена длиной 255 символов (заметим, что изменился лишь программный код, поддерживающий*FAT16*, он стал 32-м). Для обеспечения обратной совместимости ОС создает его псевдоним, удовлетворяющий стандарту 8.3. Делается это следующим образом.

1. Первые 3 символа после последней точки в длинном имени файла становятся расширением псевдонима.
2. Первые шесть символов длинного имени файла, за исключением пробелов, которые игнорируются, преобразуются в символы верхнего регистра и становятся шестью символами стандартного имени файла. Недопустимые символы (+ , ; = [28] ), которые могут использоваться в Windows 95, преобразуются в символы подчеркивания.
3. Добавляются символы ~1 (седьмой и восьмой) к псевдониму имени файла.

Если первые шесть символов нескольких файлов одни и те же, то добавляются символы ~2, ~3 и т.д.

*VFAT* хранит псевдонимы длинных имен в поле стандартных имен файлов записи каталога файла. Таким образом, все версии DOS и Windows могут получить доступ к файлу под длинным именем с помощью его псевдонима. Остается проблема: как хранить 255 символов имени файлов 32 байт записи каталога? Разработчики файловой системы решили эту проблему следующим образом: были добавлены дополнительные записи каталога для хранения длинных имен файлов. Чтобы предыдущие версии не повредили эти дополнительные записи каталога, *VFAT* устанавливает для них атрибуты, которые нельзя использовать для обычного файла: только для чтения, скрытый, системный и метка тома. Такие атрибуты DOS игнорирует, а следовательно, длинные имена файлов остаются нетронутыми. Подобным же образом решается проблема длинных имен в Windows NT/2000/2003/XP, применяющих для хранения имен двухбайтовый формат на каждый символ – Unicode.

Существует важное отличие FAT 32 от ее предшественниц – положение корневого каталога: он может располагаться в любом месте раздела и иметь любой раздел. Это обеспечивает динамическое изменение размера раздела.

56.Файловые операции;

Набор файловых операций

Файловая система ОС должна предоставлять пользователям набор операций для работы с файлами, оформленный в виде **системных вызовов**. В различных Часто встречающимися **системными вызовами** для работы с файлами являются.

1. Create (создание). Файл создается без данных. Объявляет о появлении нового файла;
2. Delete (удаление). Ненужный файл удаляется;
3. Open (открытие). Данный вызов позволяет прочитать атрибуты файла и список дисковых адресов для доступа к содержимому файла;
4. Close (закрытие). Файл следует закрыть, чтобы освободить пространство во внутренней таблице;
5. Read (чтение). Файл читается с текущей позиции. Открыть буфер и количество читаемых данных;
6. Write (запись). Данные записываются в файл в текущую позицию.
7. Append (добавление). Данные добавляются в конец файла;
8. Seek (поиск). Устанавливает указатель в определенную позицию;
9. Get attributes (получение *атрибутов)*;
10. Set attributes (установка атрибутов). установить атрибуты файлу после его создания;
11. Rename (переименование).
12. Execute (выполнить). Используя этот системный вызов, файл можно запустить на выполнение.

Microsoft определила набор функциональных вызовов, называемый Win 32 API (Win 32 Application Programming Interface). Они представляют собой библиотечные процедуры, которые либо обращаются к системным вызовам, чтобы выполнить требуемую работу, либо выполняют ее прямо в пространстве пользователя.

Многие вызовы API создают объекты ядра того или иного типа (файлы, процессы, потоки, каналы и т.д.). Каждый вызов, создающий объект, возвращает вызывающему процессу результат, называемый дескриптором (небольшое целое число). Дескриптор используется впоследствии для выполнения операций с объектами. Дескриптор может быть дублирован и передан другому процессу защищенным способом, что предоставляет второму процессу контролируемый доступ к объекту, принадлежащему первому процессу. С каждым объектом ассоциирован дескриптор безопасности, описывающий, кто и какие действия может, а какие не может выполнять с данным объектом.

**Способы выполнения файловых операций**

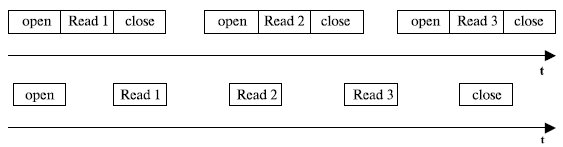
ОС необходимо выполнить ряд постоянных для всех операций действий.

1. По символьному имени файла найти его характеристики, которые хранятся в файловой системе на диске.
2. Скопировать характеристики в оперативную память
3. На основании характеристик файла проверить права пользователя на выполнение запрошенной операции.
4. Очисть область памяти, отведенную под временное хранение характеристик файла.

Кроме того, каждая операция включает ряд уникальных для нее действий, например, чтение определенного набора кластеров диска, удаление файла, изменение его атрибутов и т.п.

ОС может выполнить последовательность действий над файлами двумя способами.

1. Для каждой операции выполняются как универсальные, так и уникальные действия. Такая схема иногда называется **схемой без заполнения состояния операции**.
2. Все универсальные действия выполняются в начале и конце последовательности операций, а для каждой промежуточной операции выполняются только уникальные действия.



При втором способе в ФС вводится два специальных системных вызова: open и close.

Преобразование символьного имени файла в его уникальное числовое имя, копирование характеристик файла из дисковой области в буфер оперативной памяти и проверка прав пользователя на выполнение запрошенной операции. Вызов close освобождает буфер с характеристиками файла.

57.Контроль доступа к файлам;

Файлы – один из видов разделяемых ресурсов, доступ к которым ОС должна контролировать. Пользователи являются **субъектами доступа**, а разделяемые ресурсы – **объектами**. Пользователь осуществляет доступ к объектам не непосредственно, а c помощью **прикладных процессов**, которые запускаются от его имени.

У каждого объекта доступа существует **владелец**. Во многих ОС существует особый пользователь – администратор "superuser", который имеет все права по отношению к объектам системы, не обязательно являясь их владельцем.

Различают **два** основных подхода к определению прав доступа.

1. Избирательный доступ – владелец объекта определяет допустимые операции с объектом. Позволяет администратору и владельцам объекта определить права доступа произвольным образом, по их желанию.
2. Мандатный доступ (принудительный) – администратор наделяет пользователя или группу определенными правами по отношению к каждому разделяемому ресурсу. В этом случае группы пользователей образуют строгую иерархию, причем каждая группа пользуется всеми правами группы более низкого уровня иерархии.

Каждый пользователь (группа) имеет символьное имя, а также уникальный числовой идентификатор. Все идентификационные данные, а также сведения о вхождении пользователя в группы хранятся в специальном файле (UNIX) или базе данных (Windows NT).

Вход пользователя в систему порождает процесс – оболочку, который поддерживает диалог с пользователем и запускает для него другие процессы. Любой порождаемый процесс наследует идентификаторы пользователя и групп от процесса родителя.

Набор файловых операций может включать всего несколько укрупненных операций, например, для файлов и каталогов: читать, писать и выполнять.

Для каждого файла и каталога создается **список управления доступом** (Access Control List, ACL), в котором описываются права на выполнение операций пользователей и групп пользователей по отношению к этому файлу или каталогу.

**ACL** можно представить в виде набора идентификаторов пользователей и групп пользователей, в котором для каждого идентификатора указывается набор разрешенных операций над объектом. Сам список ACL состоит из элементов управления доступом (Access Control Element, ACE), которые соответствуют одному идентификатору. Список ACL с добавлением идентификатора владельца называют характеристиками безопасности.

Проверки прав доступа для объектов любого типа выполняются централизованно с помощью **монитора безопасности** (Security Reference Monitor), работающего в привилегированном режиме.