А. Ф. Губкин

Конспект лекций по дисциплине: «Операционные системы и сети»

2007г.

АННОТАЦИЯ ДИСЦИПЛИНЫ

Дисциплина «Операционные системы и сети» является базовой для специальностей, связанных с информатикой, программированием и вычислительной техникой. В дисциплине изучаются понятие термина «операционная система», типы операционных систем и принципы их построения. Основное внимание уделяется изучению управляющей программы. Рассматриваются методы управления основной памятью, управление процессами и ресурсами, управление устройствами, вводом-выводом и данными. Изложение иллюстрируется примерами из реальных операционных систем. Также в дисциплине затрагиваются основные вопросы организации компьютерных сетей. Рассматриваются уровни модели OSI, сетевые протоколы, маршрутизация в сетях.

Тема 1. Определение понятия ОС

2.1. Состав программного обеспечения вычислительной системы

Многие авторы отмечают, что трудно дать всеобъемлющее и точное определение понятия ОС. В существующей литературе сложилось два понимания этого термина. Для того чтобы разобраться в различных смыслах, который вкладывается в понятие ОС, следует классифицировать весь спектр программ, функционирующих в среде вычислительной системы (ВС). На рис. 1 представлена такая классификация.

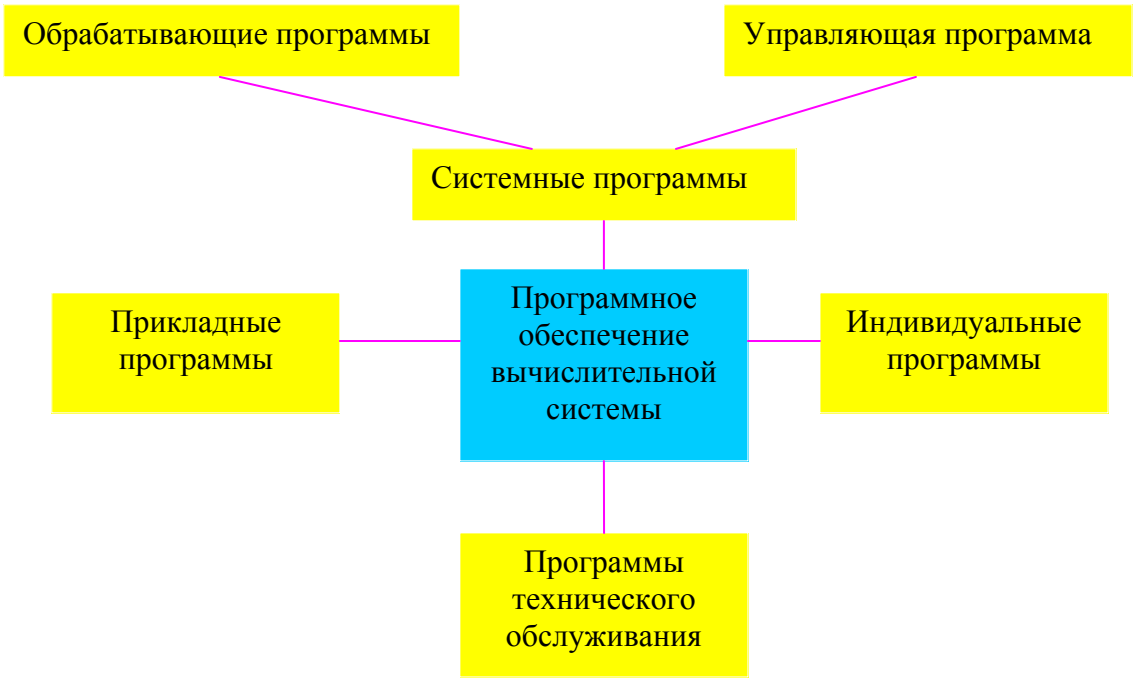


Рис. 1. Классификация программного обеспечения ВС

Индивидуальные программы – это программы, которые пишут для личных целей. Это может быть проверка или иллюстрация какого-то метода, расчет каких-то значений и так далее. Основная цель – получение быстрого результата. Индивидуальные программы разрабатываются без использования технологического процесса разработки, как правило, сразу пишется код, который по мере проникновения в задачу корректируется. Отсутствует какая-либо документация. В лучшем случае создается текстовый файл ReadMe, в котором описывается обращение к программе. Исходный код комментируется в зависимости от вкусов автора. Этот тип программ не может существовать без участия разработчика.

Прикладные программы – это рыночный продукт, который производится и распространяется в соответствии с существующими законами и нормами. Как и всякий продукт, прикладная программа отчуждена от своего разработчика, то есть пользователь может эксплуатировать эту программу без его участия. Для этого она должна быть хорошо документирована, проведены определенные испытания с целью определения ее свойств и осуществляется ее сопровождение. Все это требует соблюдения определенного технологического процесса разработки. Интересно заметить, что впервые продажа программы была осуществлена в 1980 году фирмой IBM. Этот год можно считать первым появлением рыночных отношений в сфере программного обеспечения. До этого ПО

поставлялось совместно с компьютером и было привязано к этой модели, а порой и к поставляемому экземпляру.

Программы технического обслуживания–тесно связаны с аппаратурой. Они служат для управления и обслуживания аппаратной части ВС. Широко известны такие программы как fdisk, пакет для анализа и тестирования SiSoft Sandra, тест для проверки и настройки монитора Nokia Test. Программы технического обслуживания распространяются как прикладные программы, а также включаются в состав ОС. Это программа дефрагментации диска или реализация в Windows2000 и WindowsXP управления дисками вместо ранее применяемой fdisk.

Обрабатывающие программы – это программы, которые поставляются в дистрибутиве версии ОС, инсталлируются в определенные каталоги и при обращении к ним загружаются в основную память и выполняются. Состав этих программ может меняться от версии к версии. Иногда программные модули распространяются как отдельные прикладные программы, в других версиях они включаются в состав дистрибутива. Таким образом, граница между прикладными и обрабатывающими программами в определенной степени размыта. Разумеется, большую часть дистрибутива составляют модули, которые отдельно не поставляются и являются системными программами данной версии ОС.

Управляющая программа – представляет собой набор функций и данных, которые находятся в дистрибутиве, при инсталляции загружаются в специальные файлы, а при запуске системы загружаются в определенную область основной памяти и находятся там резидентно. Управляющую программу часто называют ***ядром*** системы. Ядро ОС как правило реализует функции управления основной памятью, процессами, устройствами, вводом/выводом, файлами. Следует заметить, что в современных системах существует тенденция сократить размер ядра. Так в ОС QNX ядро реализует только функции управления основной памятью, сообщениями, сигналами и прокси. В этом случае резидентную часть ОС называют микроядром. Остальные функции реализуются в виде системных процессов - менеджеров ресурсов. Так существует Task manager, File manager и другие.

Теперь можно вернуться к вопросу о двух пониманиях термина ОС. В общей литературе, посвященной ИТ, под ОС понимают ту совокупность ПО, которая поставляется в дистрибутивах. Это понятие ОС в **широком** смысле. В специальной литературе, посвященной проблематике ОС, рассматриваются функции, составляющие управляющую программу, то есть ОС рассматривается в более **узком** смысле. Это функции по управлению процессами и ресурсами. Такое понимание ОС в широком и узком смыслах следует учитывать при чтении литературы.

Задание для самостоятельной работы!

Задание 1) Рассмотреть состав версий ОС WindowsXP (Home Edition, Professional, Media Center Edition 2005). Чем они отличаются?

Задание 2) Рассмотреть состав версий Linux.

2.1. Определение ОС

Согласно ГОСТ 15971-78 «Системы обработки данных. Термины и определения.»: **«ОС – система программ, предназначенная для обеспечения определенного *уровня эффективности* цифровой вычислительной системы за счет *автоматизированного управления* ее работой и предоставляемого пользователям *набора услуг*.»**

В этом определении существует несколько ключевых понятий. Рассмотрим их.

1) «уровень эффективности» - характеризует качество работы системы. Известны два основных показателя уровня эффективности: пропускная способность и время реакции системы на события.

Показатель «пропускная способность» используется для так называемых пакетных систем, которые предназначены для решения пакета задач. Например, это может быть обработка поступающих статистических данных, обработка изображений, поступающих с космических аппаратов. Пропускная способность измеряется в количестве решенных задач в единицу времени.

Показатель «время реакции на события» характерен для систем, которые обрабатывают поток событий и на каждое событие должны выдавать определенную реакцию. Это может быть система заказа и покупки билетов (железнодорожных или авиа), система управления ядерным реактором и огромное количество других систем.

2) «автоматизированное управление работой» ВС состоит в управлении прохождением программ и управлении ее ресурсами. Этим занимается управляющая программа.

3) «набор услуг, предоставляемый пользователям» определяется назначением и версией ОС. Эта функция возложена на обрабатывающие программы. Однако для полной сатисфакции не обойтись без прикладных программ.

Рассмотренное выше определение говорит об ОС в широком смысле.

2.1. Типы ОС

Однопрограммные ОС

В качестве примеров однопрограммных (однозадачных) ОС можно привести выпущенную в 1964 году OS360 PCP (prime control program). Эта ОС предшествовала серии мультипрограммных ОС и была выпущена для обеспечения возможности продаж системы IBM360. Дело в том, что ВМ была запущена в производство в 1962 году, а разработка программного обеспечения задержалась на два года. Поэтому для поставки потребителям этой системы и была выпущена более простая версия PCP. Историю ВС фирмы IBM можно посмотреть здесь и здесь.

Для работы с этой системой программист должен был составить задание на языке управления заданиями JCL (Job Control Language). Задание состояло из шагов. На каждом шаге выполнялась какая-то программа, например, типовое задание. Задание могло состоять из следующих шагов:

1. компилировать программу
2. собрать загрузочный модуль
3. выполнить загрузочный модуль.

Исходный текст программы и данные для ее выполнения могли быть помещены в задание. Система последовательно вызывала программы, описанные в шагах задания, и выполняла их. Структура такой системы представлена на рис. 2.

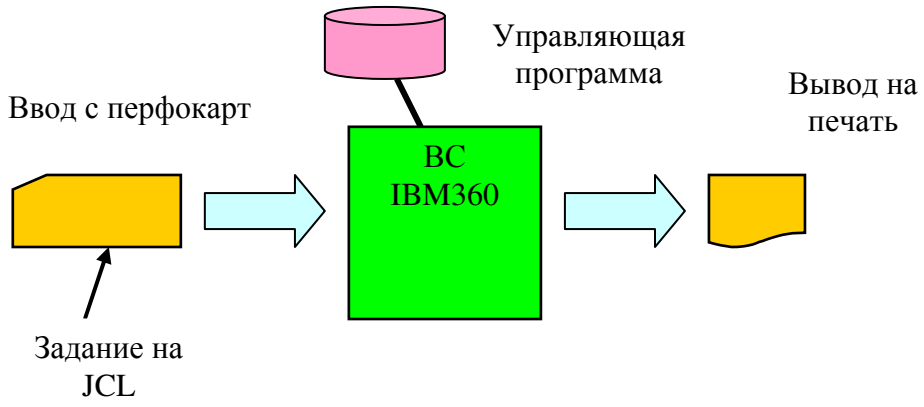


Рис. 2. Структура однопрограммной системы

Другой пример однопрограммной ОС – MS DOS v.6.22, которая поддерживается и сейчас и эмулируется версиями Windows. Разумеется, на смену перфокартам и печати

пришли ввод с клавиатуры и вывод на экран монитора, то есть общение стало диалоговым. Вместо ввода всего задания, написанного на JCL, пользователь вводит команды динамически, командный процессор разбирает очередную команду и организует ее выполнение, хотя возможность задать выполнение последовательности команд осталась в виде командных файлов.

Состав однопрограммной системы:

- процессор командного языка, который выполняет команды и командные файлы,
- система управления внешними устройствами,
- файловая система.

Характерные свойства однопрограммной ОС:

- - в системе выполняется одна задача, поэтому все ресурсы отдаются этой задаче и когда она заканчивается, ресурсы освобождаются,
- - переключение между задачами осуществляет пользователь.

Пакетные ОС

Производительность ВМ росла, и на них решались все более сложные задачи. Характер этих задач таков, что требуется осуществление однотипных расчетов для потока данных или результаты расчетов могут быть получены с определенной задержкой. Такие задачи называются пакетными в отличие от диалоговых или задач реального времени.

При обработке задач такого типа существенным является повышение производительности, измеряемое в числе решенных задач в единицу времени. И первым средством повышения производительности было исключение человека из процесса обработки, то есть переключение между задачами происходит автоматически.

Пример такой системы был реализован еще до появления системы IBM360 на более ранних машинах этой фирмы. Весь процесс обработки осуществлялся системой подготовки данных и расчетной системой.

Система подготовки данных (рис. 3) была реализована на маломощной машине IBM1401. Задания подготавливались на перфокартах и вводились в машину, а система подготавливала поток заданий на магнитной ленте (МЛ). Затем эта МЛ переносилась на более мощную расчетную систему. Результаты обработки также писались расчетной системой на МЛ и переносились для распечатки на расчетную систему.

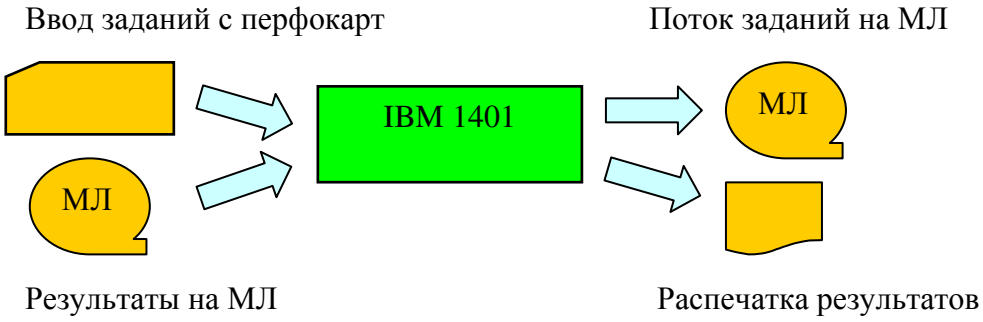


Рис. 3. Система подготовки данных

Расчетная система (рис. 4) принимала поток заданий на МЛ, считывала по очереди задания и последовательно шаг за шагом выполняла их. Результаты выполнения заданий писались на МЛ.

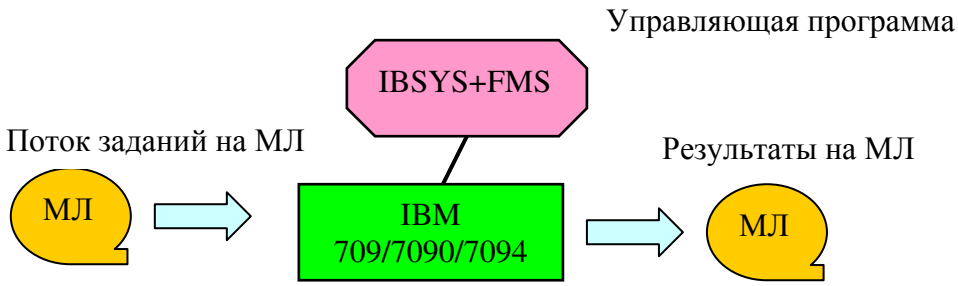


Рис. 4. Расчетная система

Состав пакетной системы:

- управление потоком заданий,
- система управления внешними устройствами,
- файловая система.

Характерные свойства пакетных ОС:

- - в системе выполняется одна задача, поэтому все ресурсы отдаются этой задаче и когда она заканчивается, ресурсы освобождаются,
- - переключение между задачами осуществляет управляющая программа.

Мультипрограммные пакетные ОС

Примерами мультипрограммных ОС могут служить серия OS360 для IBM360. Это версии MFT, MVT. Системы виртуальной памяти для IBM370: SVS, VS1, VS2, MVS, а также мэйнфреймы с MVS.

Создание мультипрограммных пакетных ОС преследовало цель еще более повысить производительность систем, выполняющих пакетную обработку. Следующим шагом после исключения человека из процесса обработки стало создание ВС, способной обеспечить одновременное выполнение нескольких задач, разделяющих устройства системы. Для осуществления этой идеи была спроектирована IBM360.

В ходе своего выполнения каждая задача занимает процессор, а также осуществляет операции ввода/вывода. По сравнению со скоростью выполнения команд процессором операции ввода/вывода достаточно медленные. Если поручить управление этими операциями периферийному процессору, освободив, таким образом, центральный процессор (ЦП), то на ЦП может выполняться другая задача. На рис. 5 показана структура такой системы.

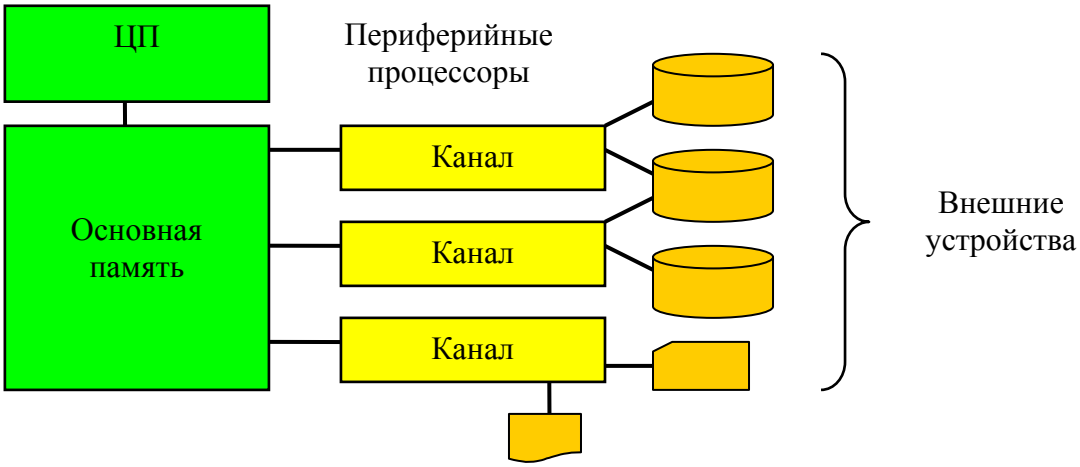


Рис. 5. Структура аппаратной части мультипрограммной пакетной ВС

Цель, которая преследуется при создании ОС для такой архитектуры, состоит в построении механизмов управления, которые обеспечивают **максимальную загрузку устройств ВМ и, прежде всего, ЦП.**

Аппаратные средства такой системы, кроме периферийных процессоров и соответствующих интерфейсов, имели развитую систему прерываний и защиту ОП и устройств от проникновения одних задач в области полномочий других задач, а также в системную область памяти. На базе таких аппаратных средств и были разработаны мультипрограммные пакетные системы.

Алгоритм управления задачами в системах такого типа основан на **принципе невытесняющей многозадачности**. Этот принцип состоит в следующем: если какая-либо задача (процесс) выполняются на ЦП, то эта задача освободит процессор либо когда она закончится, либо когда она обратиться к управляющей программе с каким-либо запросом, например, запросом на ввод/вывод. Иногда в литературе для обозначения такого принципа управления используют запись $\Delta t \to \infty$. Это следует понимать так: управляющая программа отводит задаче квант процессорного времени Δt бесконечно большой, она не собирается прерывать эту задачу.

Диалоговые мультипрограммные ОС

Идея создания диалоговых ОС возникла примерно в то же время что и пакетной обработки, но отсутствие долгое время внешних устройств для визуального отображения информации сдерживало появление систем этого типа. К диалоговым мультипрограммным ОС следует отнести UNIX, Linux и систему виртуальных машин VM370 фирмы IBM.

Цель создания диалоговых ОС – облегчить доступ пользователей к ВС в начальный период существования таких систем, сделать его комфортным и эффективным в дальнейшем. Поэтому при разработке диалоговых мультипрограммных ОС необходимо обеспечить приемлемую реакцию системы на запросы пользователей. У каждого пользователя должно создаваться впечатление, что он один владеет ресурсами ВС, он не должен чувствовать присутствие других пользователей.

Учитывая, что диалоговые ОС должны обеспечивать хорошую реакцию системы, управляющая программа в отличие от мультипрограммных пакетных ОС, определяет квант времени Δt. Для этого используется таймер, в который перед запуском приложения загружается значение интервала времени, на которое задаче предоставляется процессор. По истечению этого интервала возникает прерывание от таймера, и управляющая программа переключает процессор на другую задачу, также определяя для этой задачи значение интервала времени. Такой способ управления, когда управляющая программа определяет промежуток времени для задач, в течение которого они занимают процессор, называется вытесняющей многозадачностью.

К числу таких систем относятся системы коллективного пользования (рис. 6). Такие системы представляли собой достаточно производительную ВС, к которой подключены удаленные терминалы пользователей. Каждый пользователь, имеющий доступ к системе имел имя и пароль, и мог с любого терминала использовать ВС.

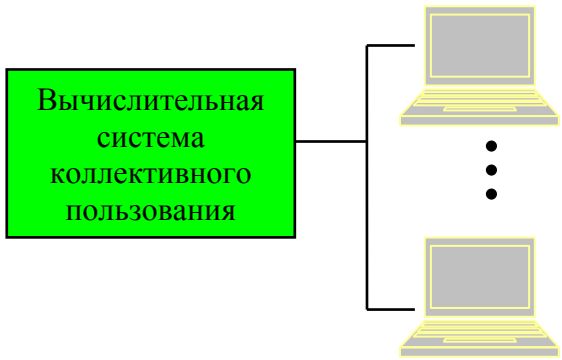


Рис. 6. Диалоговая система коллективного пользования.

С развитием средств вычислительной техники фирма IBM использовала подобную схему в разработке систем продаж для торговых фирм. Несколько десятков тысяч торговых агентов разъезжают по всему миру, рекламируя и

продавая продукцию фирмы. Каждый агент должен был иметь каталоги товаров и прочую атрибутику для своей деятельности. Фирма IBM предложила использовать так называемых «тонких клиентов», которые представляли собой, говоря современным языком, ноутбуки не имеющие жесткого диска. Тонкий клиент подключался по сети к серверам на базе мощных мэйнфреймов и отображал необходимую информацию. Кроме того, таким образом можно было, и оформить сделку. Такая технология продаж позволила не только существенно повысить качество, но и ускорить время осуществления сделок, что позволило резко увеличить объем продаж.

В современном состоянии к диалоговым мультипрограммным ОС относятся ОС серверов. Действительно, под их управлением выполняются web-сервера, интерпретаторы PHP и PERL, которые выполняют скрипты, написанные разработчиками сайтов, а также CGI-приложения.

ОС реального времени

ОС реального времени отличаются от других мультипрограммных систем тем, что число задач, выполняемых под их управлением, **фиксировано**. Системы реального времени (СРВ) предназначены для управления различными физическими объектами. В качестве таких объектов могут выступать ядерные и термоядерные реакторы, радиолокационные станции обнаружения и слежения, системы взлета и посадки самолетов или других летательных объектов и множество других объектов. Даже когда вы покупаете билеты на поезд или самолет, вы сталкиваетесь с СРВ, которая занимается резервированием билетов.

Общую схему СРВ можно представить (рис. 7) в виде взаимодействия физического объекта и ВС.

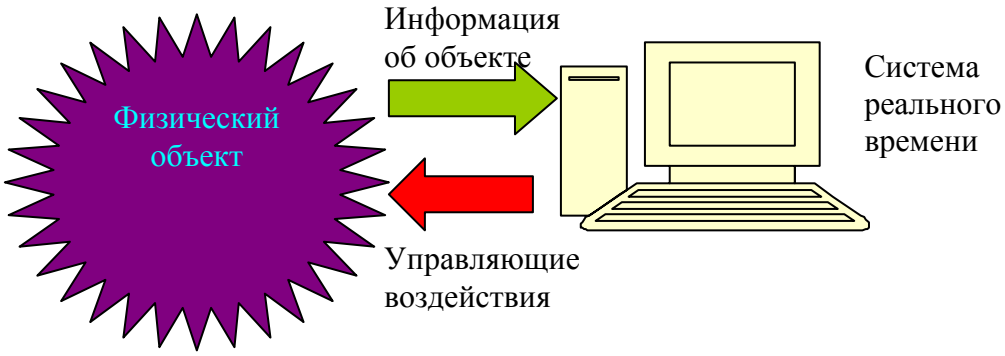


Рис. 7. Общая схема системы реального времени

Информация о физическом объекте поступает в ВС. Там она обрабатывается, и формируются управляющие воздействия. Для СРВ важно, чтобы время ответа на определенное воздействие не превышало определенной величины, то есть время реакции системы ограничено и не может превышать некоторого значения. Поэтому в СРВ используется такое понятие как масштаб времени, которое характеризует допустимое время реакции системы. Ясно, что требования ко времени реакции СРВ, управляющей летательными объектами существенно жестче, чем в системе резервирования билетов.

С точки зрения масштаба времени ОС реального времени (ОСРВ) можно разделить на два типа:

- Нарушение масштаба времени недопустимо и приводит к фатальным последствиям.
- Нарушение масштаба времени не приводит к фатальным последствиям, а снижает их эффективность.

В первом случае речь идет о системах управления сложными техническими объектами. В качестве примера систем второго типа можно привести ту же систему резервирования билетов.

Характерные свойства ОС РВ.

1. Множество внешних событий известно и можно иметь полный набор функциональных программ с изученными характеристиками.
2. Поток событий, приходящих на вход системы, частично детерминирован во времени. Существуют объекты, требующие периодического обслуживания с постоянным или мало колеблющимся периодом обслуживания.
3. Входные и выходные данные имеют относительно простую структуру, а периферийные устройства простые, но специфические методы доступа.
4. Все функциональные программы считаются отлаженными. Известны их характеристики по быстродействию и занимаемой памяти.

С точки зрения конструкции СРВ можно выделить также два типа систем:

- Встроенные системы (embedded system), которые специально разрабатываются для определенной задачи. Как правило, это связано с управлением в жестких временных ограничениях, либо в жестких условиях эксплуатации на борту летательных, плавающих объектов.
- Универсальные СРВ, которые используют серийно выпускаемые ВМ и ОС. Прежде всего, следует отметить специализирующуюся в этом направлении фирму Digital Equipment Co. (DEC), которая выпускала целую серию ОСРВ для шестнадцати разрядных ВМ PDP11, затем для 32 разрядной VAX ОС VMS, которая существует и в настоящее время на процессорах фирмы Motorolla. Кроме того, следует отметить сетевую ОСРВ QNX.

Тема 2. Понятие мультипрограммирования в ОС

2.1. Разделение времени центрального процессора в мультипрограммных системах

Мультипрограммирование – такой способ управления задачами в ОС, который позволяет двум или более задачам в процессе выполнения занимать определенную часть ОП, разделять время ЦП и другие устройства ВС. На рис. 8 показана структура ОП, в которую загружены три задачи и разделение времени ЦП.

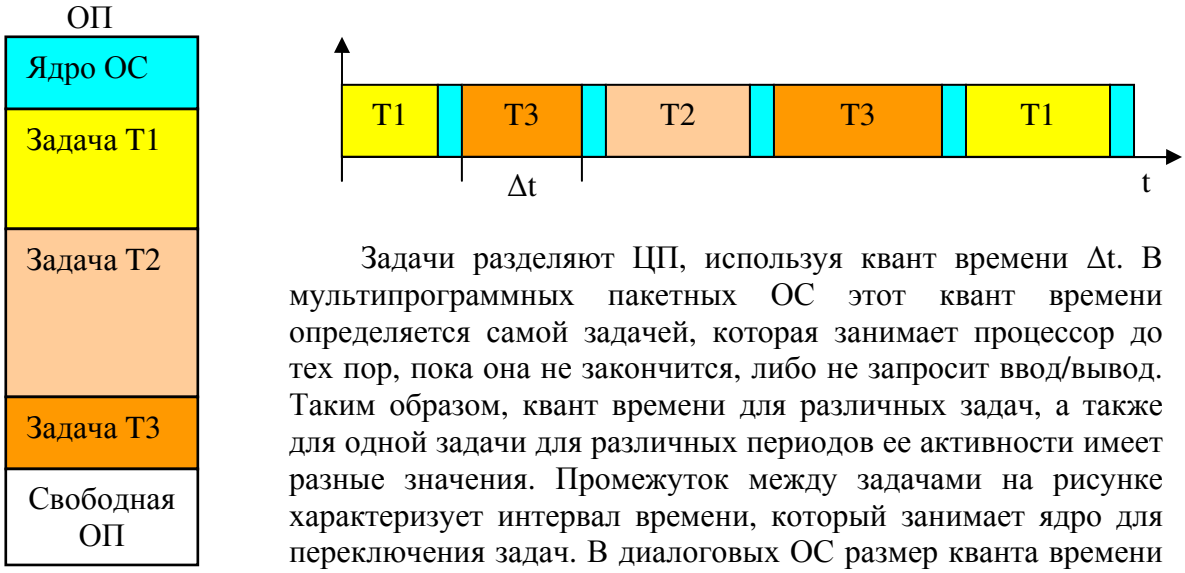
С мультипрограммным режимом работы связаны такие понятия как уровень мультипрограммирования и смесь задач.

Уровень мультипрограммирования – число задач, выполняющихся одновременно в ВС.

Смесь задач – качественная характеристика состава задач, выполняющихся одновременно, которая влияет на загрузку ЦП. Эта характеристика связана со свойствами выполняемых задач. Это свойство определяется процентом времени выполнения задачи, отводимого на операции с внешними устройствами, то есть все время выполнения задачи можно разделить на ту часть, когда задача занимает ЦП, и часть, когда она занимается вводом/выводом и освобождает ЦП. Понятие смеси задач поясним на примере двух задач, которые выполняются в мультипрограммном режиме. Пусть одна из этих задач занимается расчетами, то есть мало времени тратит на ввод/вывод, а много времени занимает ЦП. Назовем эту задачу счетной. Другая задача наоборот, занимает мало времени ЦП, но активно работает с внешними устройствами. Назовем эту задачу информационной. Выполнение этих двух задач позволит загрузить и ЦП, и внешние устройства, поскольку счетная задача будет иметь возможность занимать ЦП в те интервалы времени, когда информационная задача осуществляет работу с внешними устройствами. Главное обеспечить информационной задаче возможность занимать центральный процессор своевременно для запуска очередных операций с внешними устройствами.

Если бы в смеси присутствовали только счетные задачи, то внешние устройства бы простаивали, а процессор был бы загружен. Если бы в смеси присутствовали только

информационные задачи, то ЦП бы простаивал. Таким образом, формирование оптимальной смеси задач улучшает производительность ВС.



Задачи разделяют ЦП, используя квант времени Δt. В мультипрограммных пакетных ОС этот квант времени определяется самой задачей, которая занимает процессор до тех пор, пока она не закончится, либо не запросит ввод/вывод. Таким образом, квант времени для различных задач, а также для одной задачи для различных периодов ее активности имеет разные значения. Промежуток между задачами на рисунке характеризует интервал времени, который занимает ядро для переключения задач. В диалоговых ОС размер кванта времени определяется алгоритмом диспетчеризации управляющей программы.

Рис. 8. Структура ОП и разделение времени ЦП

2.2 Структура и организация управляющей программы

Структура управляющей программы мультипрограммной пакетной ОС

Программа начальной загрузки IPL (Initial program loader) присутствует во всех системах и служит для загрузки ОС при старте компьютера.

Планировщик заданий состоит из трех компонентов. Это задачи системного ввода, которых может быть до трех, в зависимости от количества устройств ввода заданий. Задачи инициации/завершения управляют созданием и завершением прикладных задач. Их может быть до 15. Задачи системного вывода управляют выводом информации на различные устройства, и их может быть до 36. Все задачи планировщика заданий являются системными и размещаются в той же области ОП, что и прикладные задачи. Планировщик заданий используется только в пакетных системах. В других типах ОС его использование не имеет смысла, поскольку нет потока заданий, который он обрабатывает.

Следующие компоненты управляющей программы являются функциями ядра и располагаются резидентно в ОП. Это функции управления ОП, задачами, внешними устройствами, вводом/выводом и данными. Подобные функции существуют и в других типах ОС, но реализуют другие алгоритмы.

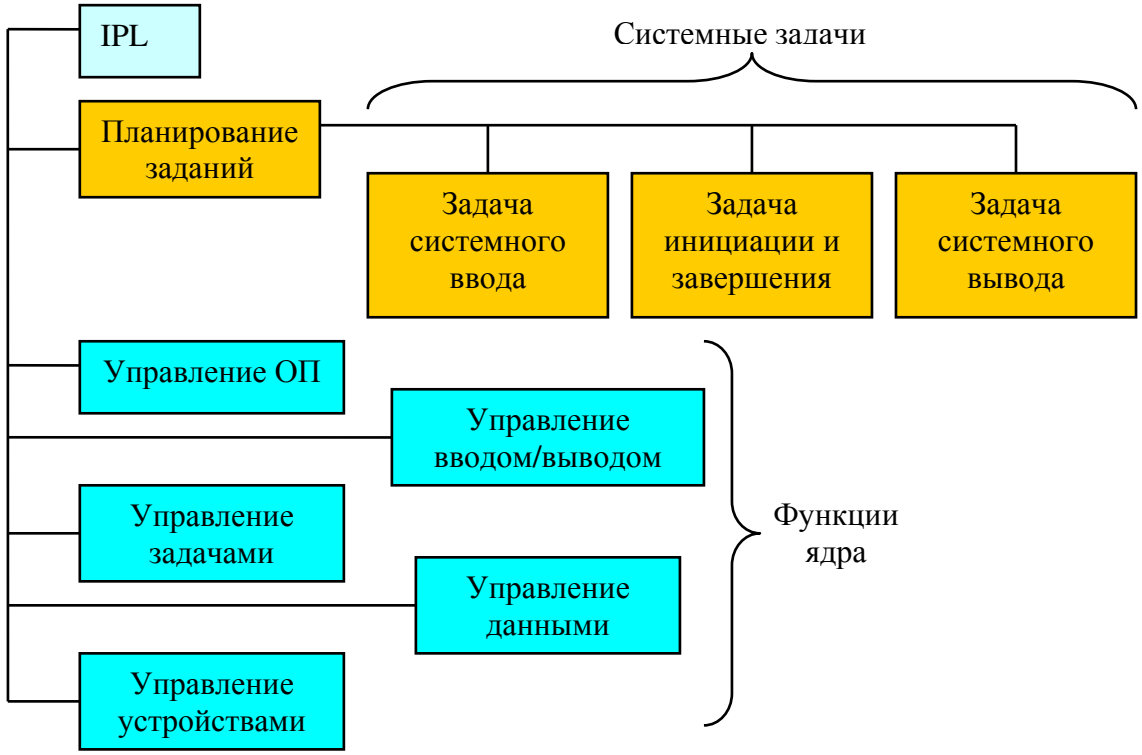


Рис. 9. Структура управляющей программы мультипрограммной пакетной ОС

Планировщик заданий.

Планировщик заданий обрабатывает входной пакет заданий и управляет прохождением заданий через ВС. Пакет заданий состоит из отдельных последовательно поступающих заданий и заканчивается признаком конца – два символа "//". Пример текста на языке управления заданиями (JCL – job control language) приведен ниже.

```
//task1                 JOB   <параметры оператора JOB>
//                      STEP  PGM=FORTRAN <другие параметры оператора STEP>
//<имя DD оператора>    DD    <параметры DD оператора>
. . . . . .. . . . . .. . . . . .
//<имя DD оператора>    DD    <параметры DD оператора>
//                      STEP  PGM=LINK <другие параметры оператора STEP>
//<имя DD оператора>    DD    <параметры DD оператора>
. . . . . .. . . . . .. . . . . .
//<имя DD оператора>    DD    <параметры DD оператора>
//                      STEP  GO=*
//SYSIN                 DD    *
            <Входные данные программы>
//task2                 JOB   <параметры оператора JOB>
 . . . . . .<описание  второго задания>. . . . . . . . . . . . .
//                                                    . . . . . .
```

Оператор JOB определяет начало описания очередного задания. Этот оператор содержит параметры для всего задания. Например, оператор CLASS=A описывает класс, в котором будет выполняться это задания.

Оператор STEP задает программу, которая будет выполняться на этом шаге, а также параметры для этого шага. Так на первом шаге будет выполняться программа FORTRAN – компилятор с языка FORTRAN. На втором шаге выполняется LINK, а на третьем выполняется изготовленный загрузочный модуль. Далее идут другие задания, входящие в пакет.

Пакет заданий мог размещаться на перфокартах, на магнитной ленте или на магнитном диске. Когда появились средства диалогового ввода, то пользователь стал вводить задание с клавиатуры, а далее задание поступало на обработку стандартным образом.

Стандартная обработка задания состоит в следующем. Задача Системного Ввода вводит задание в ВС, производит разбор текста задания и записывает **список таблиц задания** (СТЗ), в которых содержатся параметры из операторов JCL. Список таблиц для одного задания выглядит следующим образом (рис. 10).

Такие СТЗ всех заданий, поступающих в ВС составляют **очередь входных работ**, которая имеет следующую структуру. Каждое задание принадлежит определенному классу, который задается параметром CLASS в операторе JOB. Классов заданий в ВС может быть до 15. Классы обозначаются латинскими буквами от A до O. Каждый класс входных работ характеризуется определенным объемом ОП, предоставляемым заданиям и временем выполнения. Таким образом, задачи шагов задания различных классов формируют смесь задач, выполняемых в мультипрограммном режиме. Задачи одного задания, а также задания одного класса выполняются последовательно. Структура очереди входных работ представлена на рис. 11. Как уже упоминалось, задач системного ввода может быть до трех, и каждая задача читает описания заданий на соответствующем устройстве и пополняет очередь входных работ.

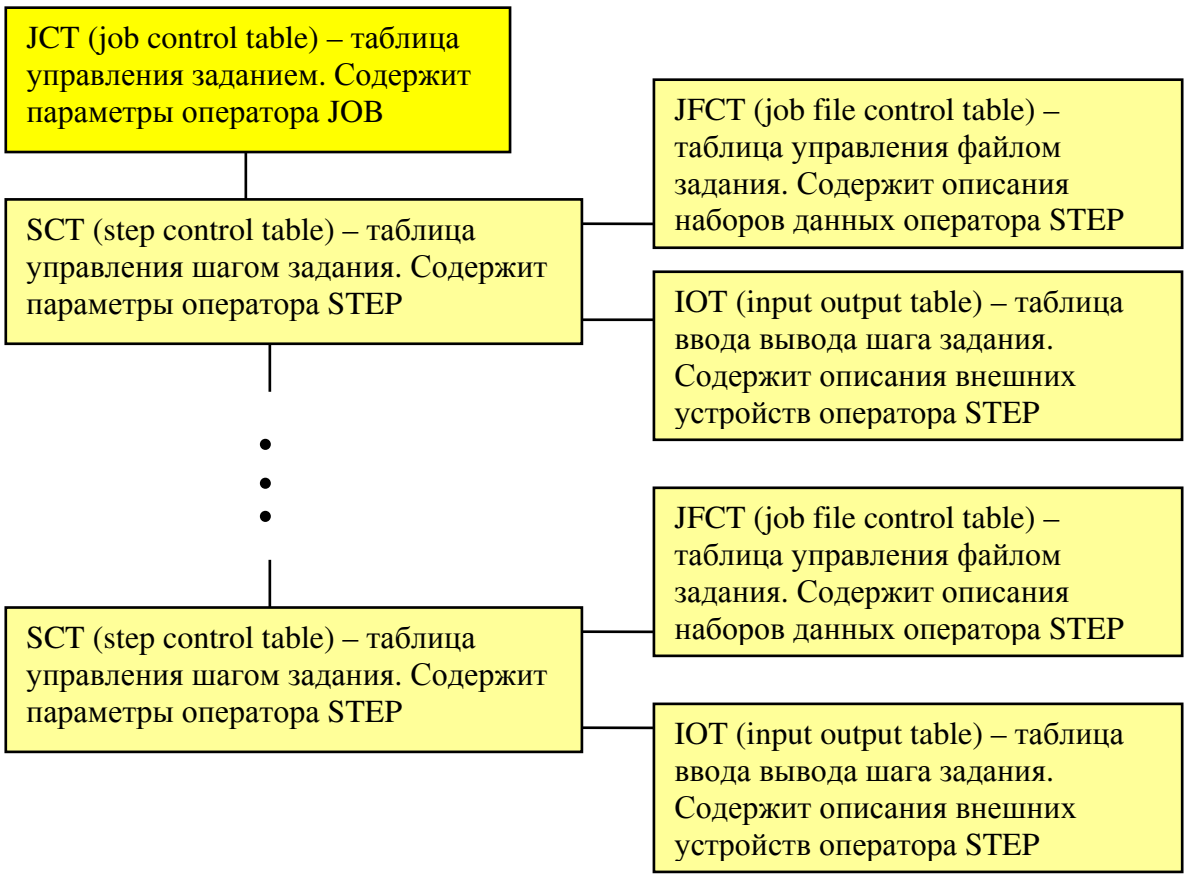


Рис. 10. Структура списка таблиц для одного задания

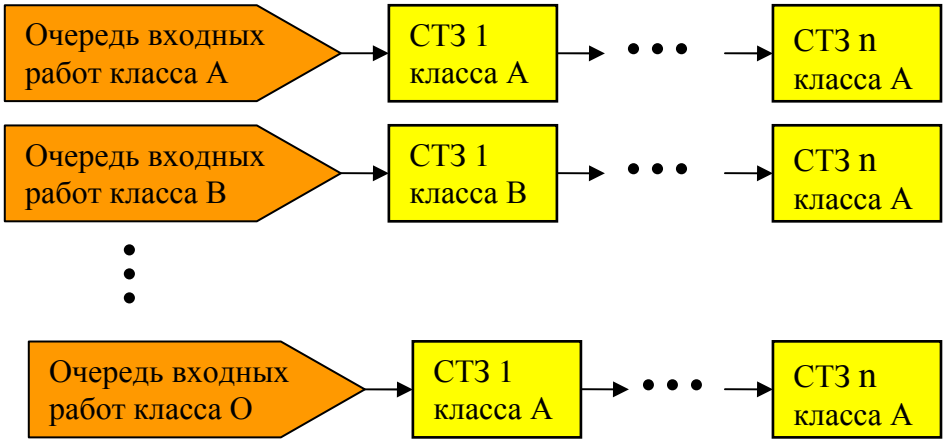


Рис. 11. Структура очереди входных работ

Очередь входных работ каждого класса обрабатывает задача инициации/завершения этого класса. Таким образом, может быть до 15 задач инициации/завершения.

Задача инициации/завершения выбирает очередное задание из очереди входных работ данного класса и последовательно обрабатывает шаги этого задания, подготавливая для очередного шага выполнение программы шага задания. Подготовка состоит в запросе раздела ОП и требуемых ресурсов для задачи шага задания. После того, как функции ядра отработают эти запросы и выделят требуемые ресурсы, задача инициации/завершения создаст задачу для шага задания, в рамках которой выполняется программа шага, и передаст ей управление. После завершения задачи шага задания, управление опять получает задача инициации/завершения, которая закрывает незакрытые программой шага задания файлы, уничтожает предписанные файлы, освобождает использованные ресурсы, освобождает участок ОП и выводит информацию о выполнении задачи шага задания. Схема функционирования задач планировщика при обработке очереди входных работ показана на рис. 12. Разумеется, в реальных системах все 15 классов использовались в редких случаях.

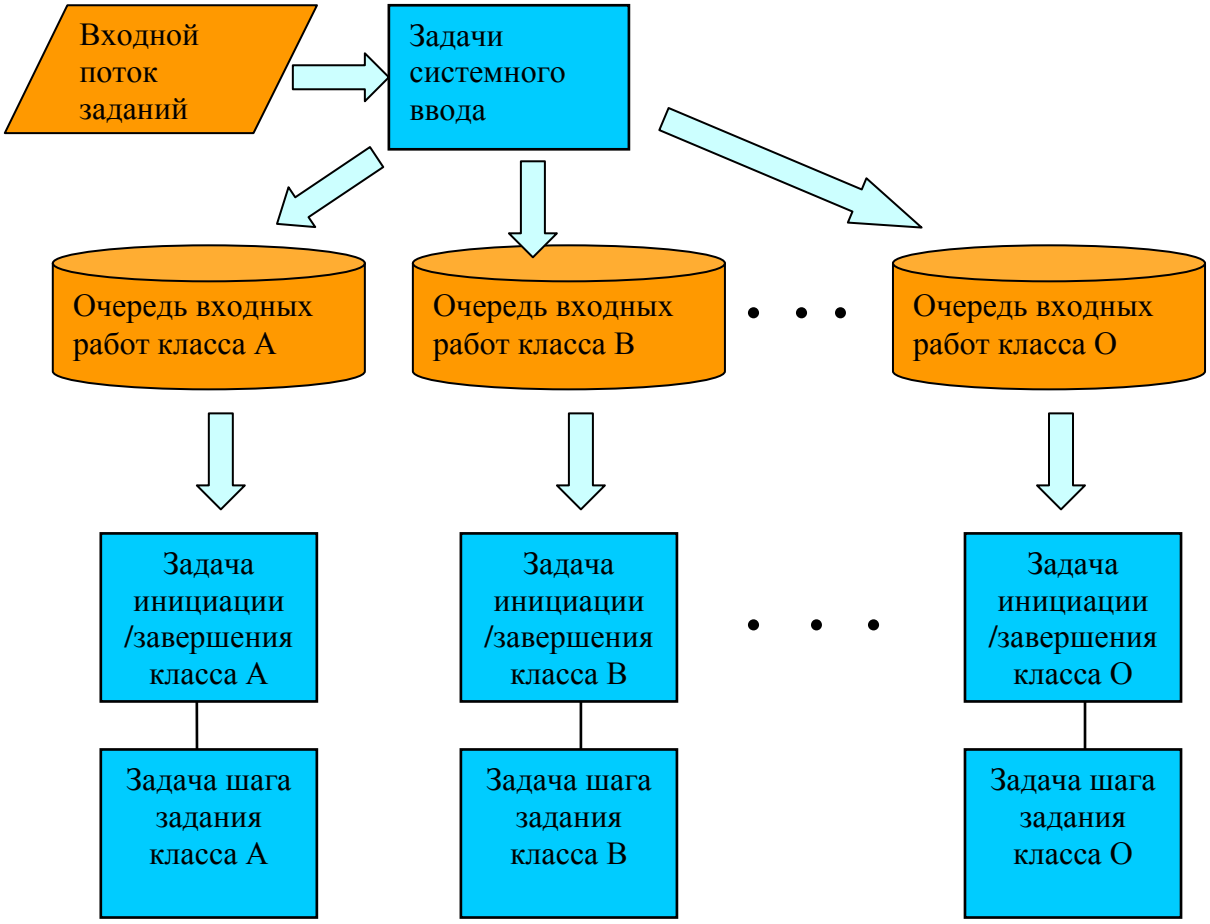


Рис. 12. Структура обработки входного потока заданий

Задачи системного ввода, инициации/завершения и шага задания выводят различную информацию в процессе работы. Задачи выполняются в различные промежутки времени, разделяя время ЦП, а информация, выводимая в процессе выполнения определенного задания, должна выводится слитно. Для решения этой проблемы используется виртуальное устройство вывода, которое называется **очередью выходных работ**.

В системе могло быть до 36 очередей выходных работ, которые обозначались двадцатью шестью латинскими буквами (A,…,Z) и десятью арабскими цифрами (0,…,9). Так, например, очередь A связывалась с принтером, а очередь B с выводом на перфокарты. С появлением систем телеобработки, удаленных систем, вывод на эти системы связывался с очередями выходных работ.

Каждая очередь выходных работ представляет собой последовательность списков вывода. Каждый список вывода состоит из управляющих блоков и относится к одному заданию (рис. 13). Выходные данные для каждого задания формируются задачей системный ввод, а также задачами инициации/завершения и шага задания для каждого шага задания. Задания различных классов **входных** работ выполняются параллельно и могут выводить информацию в один и тот же класс **выходных** работ. Поэтому вначале формируется список вывода для задания, а затем он вставляется в очередь класса выходных работ (рис. 14).

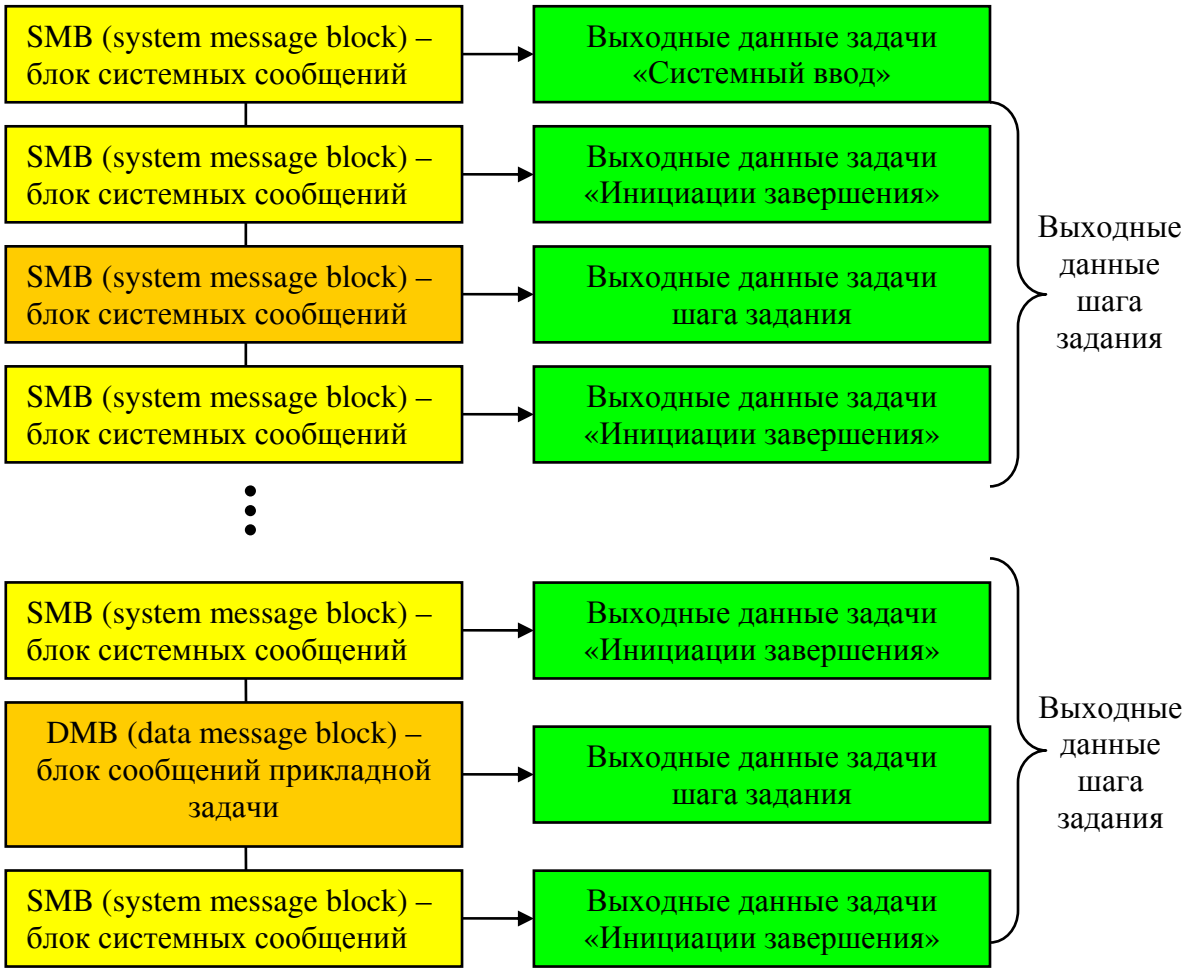


Рис. 13. Список управляющих блоков вывода для одного задания.

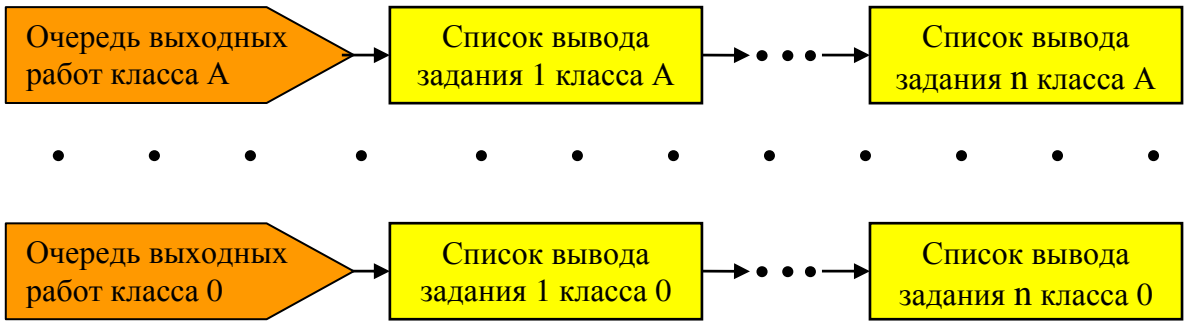


Рис. 14. Очереди выходных работ.

Все задачи как системные, так и прикладные используют функции ядра, для того чтобы получить участок ОП, осуществить ввод/вывод, работать с файлами. Поэтому функции ядра называют системой управления задачами. Обращение к функциям ядра осуществляется посредством программного интерфейса (API – application program interface), который базируется на синхронных программных прерываниях, реализуемых соответствующими командами процессора.

Программный интерфейс (API – application program interface) предназначен для программистов и служит для использования в разрабатываемых программах функций ОС. Для реализации API используется синхронное (программное) прерывание. Вызов из программы функции ядра ОС состоит в выполнении прерывания. Предварительно в

регистры загружаются параметры и в определенный регистр код функции. При выполнении прерывания управление передается ядру ОС. Функция ядра, обрабатывающая прерывание, анализирует код функции и передает управление функции ОС, определяемой этим кодом (рис. 15).

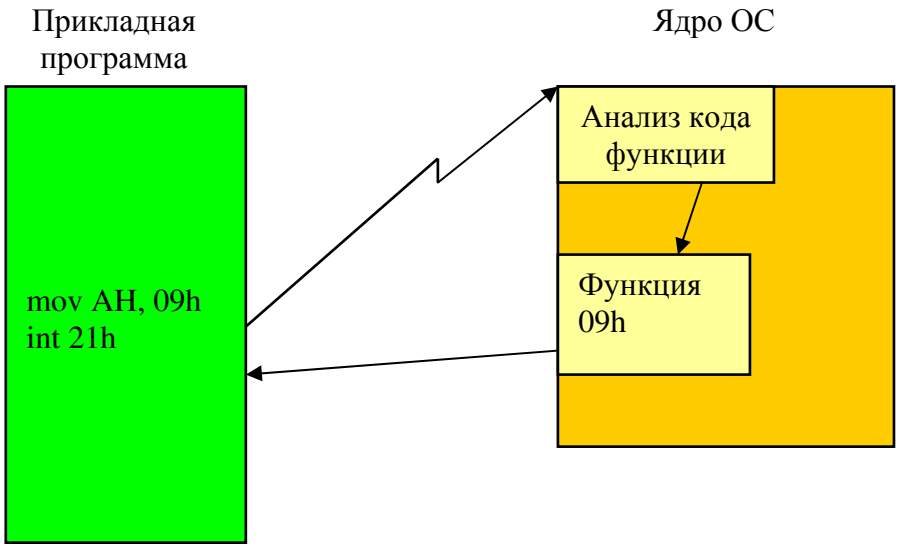


Рис. 15. Реализация программного интерфейса

Структура управляющей программы мультипрограммной диалоговой ОС

Структура управляющей программы мультипрограммной диалоговой ОС показана на рис. 16**.** В диалоговых ОС, в отличие от пакетных систем, неизвестна информация о последовательности выполняемых пользователем программ и их описание отсутствует. В каждый момент времени запускается некоторая задача, и ее описание известно ОС в этот момент времени. Поэтому надобность в планировщике заданий отсутствует, а необходимо присутствие командного процессора, который обрабатывает и исполняет команды пользователя.

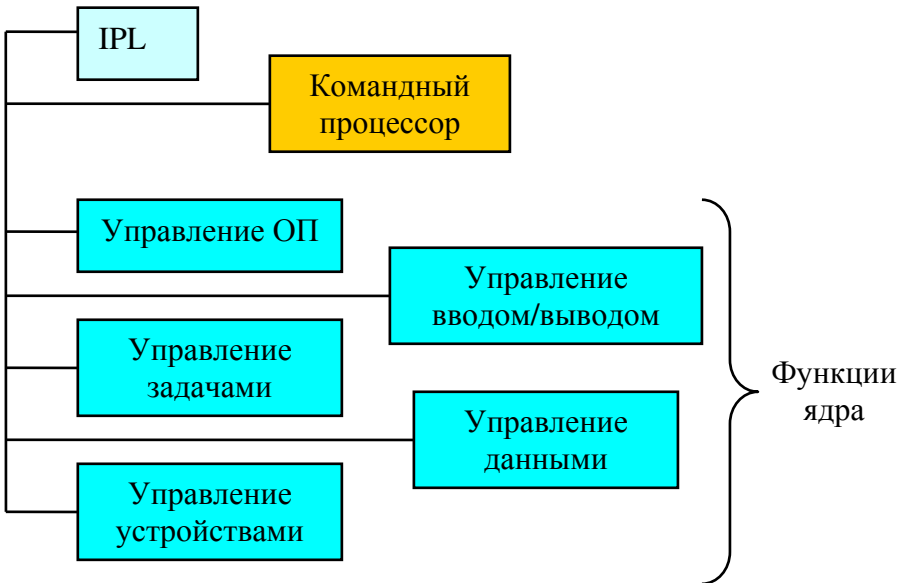


Рис. 16. Структура управляющей программы мультипрограммной диалоговой ОС

Естественно, что алгоритмы, реализованные в управляющих программах диалоговых систем отличны от тех, что используются в пакетных ОС. Так, например, алгоритмы

диспетчеризации используют вытесняющую многозадачность и стараются отделить задачи, требующие большого количества процессорного времени (пакетные задачи), от задач, использующих незначительное количество процессорного времени (диалоговых).

Тема 3. Управление основной памятью в ОС

Организация управления ОП в ВС тесно связана с архитектурой аппаратных средств и поддерживается специальными механизмами, реализованными в аппаратуре. На организацию системы управления ОП влияют:

- организация памяти в компьютере,
- механизмы защиты памяти,
- механизмы адресации.

Функции управления ОП располагаются резидентно в ядре ОС. Их используют как пользовательские и системные программы, так и другие модули ядра. Реализация управления ОП для каждой ВС весьма специфична и зависит от возможностей аппаратуры и *стратегии распределения памяти*, то есть принципа, в соответствии с которым выделяется память под различные задачи.

Система управления ОП выполняет следующие функции:

1) учет ОП в ОС,
2) планирование запросов на выделение свободных участков ОП,
3) выделение и освобождение ОП.

Учет состояния памяти заключается в отслеживании занятых и свободных областей памяти в каждый момент времени посредством изменения информации в системных структурах данных.

Планирование запросов на предоставление памяти состоит в определении задачи или программы, получающей память, момента времени, в который выделяется память, в определении объема выделяемой памяти.

Выделение и освобождение памяти связаны с работой алгоритмов поиска свободных участков памяти требуемого объема, коррекцией системных структур данных и предоставлением информации о выделенном участке памяти в ответ на запрос.

Реализация этих функций существенно зависит от аппаратной части, а также от назначения ОС. Рассмотрению стратегий управления памятью, способов реализации функций посвящена эта тема.

3.1 Связывание загрузочного модуля с адресами ОП

Логическое адресное пространство - совокупность программных адресов, которые занимают программы и данные задачи.

Физическое адресное пространство задачи - совокупность абсолютных адресов физической памяти, в которых располагается задача.

Система управления ОП осуществляет отображение логического адресного пространства задачи в физическое. Этот процесс состоит из поиска и выделения свободных участков физической памяти, загрузки копии программы с диска в ОП и связывания программных адресов с физическими. Загрузку копии осуществляет модуль ядра - загрузчик. Он же настраивает программные адреса на физическую память в случае статического связывания. При использовании динамического связывания преобразование программного адреса в абсолютный адрес осуществляется аппаратными средствами.

Статическое связывание – назначение корректных физических адресов в командах загруженной программы, использующих прямую адресацию. Лучше всего это рассмотреть на весьма популярном примере. Когда на ассемблере для MS DOS создается загрузочный модуль типа **.exe** , то необходимо в первых командах кода написать

```
mov    AX, seg DATA
mov    DS, AX
```

Второй операнд первой команды задает прямой адрес – сегментный адрес сегмента DATA. Этот адрес в момент компиляции неизвестен и в файле копии загрузочного модуля на диске корректное значение адреса отсутствует. В начале файла копии загрузочного модуля на диске помещается таблица, которая содержит указания на поля операндов в

командах, которые требуют настройки при перемещении копии загрузочного модуля в ОП, а также описание этих адресов. Используя эту информацию, загрузчик корректирует адреса, поскольку их значения ему известны. Все эти действия загрузчик выполняет до передачи управления загрузочному модулю, поэтому подобный способ связывания называется статическим.

Динамическое связывание – вычисление действительных адресов в процессе выполнения загрузочного модуля аппаратными средствами процессора. Такой способ применяется в случае относительной, косвенной, страничной, сегментной и сегментно-страничной способах адресации.

Логическое адресное пространство может быть односегментным и многосегментным. Под **логическим сегментом** понимается логически законченный объем информации (программа, подпрограмма, структура данных, объект класса и т.д.). Широко используется понятие **физического сегмента**, под которым понимают некоторый объем физической памяти.

Физическое адресное пространство может быть одноуровневым и многоуровневым. Многоуровневое физическое адресное пространство представляет собой иерархию уровней памяти. Как правило, выделяют сверхоперативную, оперативную, массовую и внешнюю памяти. Уровни памятей различаются по своим временным, емкостным и стоимостным характеристикам. Размещение задач и адресного пространства одной задачи, динамическое управление этим размещением позволяют существенно повысить пропускную способность ВС.

Компьютерные системы, имеющие одноуровневую организацию физического адресного пространства ОП, называются не страничными. При многоуровневой организации физического адресного пространства используются системы виртуальной памяти. Иногда такой способ называют страничной памятью. В настоящем изложении под страничной памятью будет пониматься вполне определенный способ организации физической памяти.

3.2 Управление ОП в нестраничных системах

Общей **особенностью** стратегий управления ОП в нестраничных системах является то, что **все логическое адресное пространство** задачи загружается в физическую память. В результате чего, мало используемые функции, такие, например, как функции обработки ошибок, загружаются в ОП и память используется нерационально. Однако, эти методы управления памятью требуют меньших затрат на реализацию системы адресации процессора, более простые в реализации. К методам управления нестраничной памятью относятся:

одиночное непрерывное распределение,
распределение статическими разделами,
распределение динамическими разделами.

Одиночное непрерывное распределение

Стратегия одиночного непрерывного распределения используется в однопрограммных ОС. Она состоит в следующем.

После загрузки ОС вся оставшаяся ОП используется для задач, выполняющихся по командам пользователя. В каждый момент времени выполняются программы только одной задачи. Поэтому вся свободная память предоставляется этой задаче.

В таких системах управление ОП на уровне задач практически отсутствует, но предоставляются возможности управления памятью на уровне программ. Для этого в ядре содержатся функции выделения и освобождения памяти, которые используются в программах при их разработке. Таким образом, функции управления памятью осуществляет программист. На рис. 17 представлен пример одиночного непрерывного

распределения памяти. Область памяти управляющей программы находится в младших адресах. Остальная память отводится задаче.

Рис. 17 Одиночное непрерывное распределение памяти

Основные функции управления памятью выполняются на уровне программ, а не задач. Так функция учета заключается в построении **списка блоков управления памятью MCB (memory control block)**. Адрес первого блока этого списка находится в области управляющей программы. В начале каждого участка памяти находится MCB, в котором содержится описание этого участка: размер, занят или свободен, какой программой или данными занят участок.

Управление этим списком осуществляется функциями выделения и освобождения памяти. Планирование запросов на выделение и освобождение памяти реализует программист в момент разработки программы.

Конечно, даже в однозадачной ОС необходимо защищать область памяти управляющей программы от вмешательства программ пользователя. Это можно делать с помощью регистра защиты, в котором хранится адрес начала области памяти, предоставляемый задаче. При вычислении абсолютного адреса аппаратные средства должны сравнивать его значение с содержимым регистра защиты. Если значение абсолютного адреса лежит в области управляющей программы, то возникает прерывание по защите памяти. Однако в реальных ОС, использующих такую стратегию управления памятью, как правило, защита области УП не осуществляется. Это объясняется тем, что порча области памяти УП происходит в момент выполнения задачи пользователя. Особенно часто это случается при отладке программ. Поскольку в однопрограммных ОС программы других пользователей в этот момент времени не содержатся в ОП, то страдает от несанкционированных действий сам нарушитель. Восстановление системы, как правило, осуществляется ее перезагрузкой.

Иное дело в мультипрограммных системах. Несанкционированный доступ к области УП или областям памяти задач других пользователей может отразиться на их работе, поэтому такое вмешательство недопустимо и области памяти должны быть защищены.

Достоинством стратегии одиночного непрерывного распределения ОП является простота реализации.

Недостатки состоят в следующем:

1) невозможно реализовать мультипрограммирование. При достаточно мощном процессоре и больших объемах ОП применять эту стратегию нецелесообразно, так как память будет использоваться плохо, а процессор будет простаивать. Данный способ пригоден для маломощных небольших ВС;

2) этой стратегии присуще плохое использование памяти. Часть памяти не используется совсем, так как объем загружаемой задачи обычно меньше объема свободной ОП. Кроме того, поскольку в физическую память загружается все логическое адресное пространство задачи, то загружаются и редко исполняемые участки кода. К ним можно отнести, например функции обработки ошибок;

3) отсутствие защиты областей ОП программных модулей, присущее стратегии одиночного непрерывного распределения приводит к существенным трудностям при реализации сложных программных проектов. В процессе отладки одни модули портят память других, и такие ошибки весьма трудно отыскиваются.

В качестве примера управления ОП, организованной по такой стратегии, рекомендуется разобрать систему управления ОП MS DOS.

Распределение разделами

Стратегия распределения памяти разделами позволяет реализовать мультипрограммный режим работы. Задаче выделяется участок памяти – раздел.

Поскольку принцип распределения памяти разделами используется в мультипрограммных системах, и задачи различных пользователей выполняются в разных разделах, то должен быть исключен доступ задачи к другим разделам, то есть аппаратные средства должны обеспечивать защиту разделов и области памяти управляющей программы. Существуют способы защиты регистрами и ключами.

Защита регистрами заключается в реализации в процессоре пар регистров: базового регистра и регистра защиты. В базовый регистр загружался базовый адрес раздела памяти, отведенного задаче, а в регистр защиты загружался размер раздела. Таким образом, определялось адресное пространство раздела. При вычислении исполнительного адреса в программе, размещенной в этом разделе, этот адрес проверялся на принадлежность адресному пространству раздела. Если адрес не принадлежал разделу, то возникало прерывание по защите памяти. Так в процессоре DEC PDP11 было 16 пар таких регистров. Восемь из них могли использоваться управляющей программой, а восемь прикладной задачей. Таким образом, каждая задача могла размещаться в восьми разделах, каждый из которых был защищен своей парой регистров.

Защита ключами основана на том, что ОП выделяется определенными порциями. Так в компьютерах IBM память выделялась блоками по 2Кб. С каждым блоком был связан **четырех байтный регистр защиты**. При создании задачи ей присваивался **код защиты**, который записывался в **Слово Состояния Программы** (PSW) и регистры защиты тех блоков памяти, которые принадлежали этой задаче. При вычислении исполнительного адреса, код защиты в PSW сравнивался с кодом защиты в регистре защиты блока памяти, которому принадлежал исполнительный адрес. Если коды не совпадали, то возникало прерывание по защите памяти. Поскольку на хранение кода защиты отводилось 4 разряда, то коды защиты задач могли принимать значения от 1 до 15. Нулевой код приписывался системным задачам, и этот код обеспечивал доступ к любому блоку памяти.

Распределение статическими разделами

Принцип распределения ОП статическими разделами состоит в том, что вся свободная память после загрузки ОС делится на разделы фиксированной длины. В эти разделы и загружаются задачи для выполнения. Это наиболее простой способ управления памятью, позволяющий организовать мультипрограммный режим работы. Данная стратегия управления ОП использовалась в первых мультипрограммных системах. Также она популярна в управляющих ВС, поскольку характеристики задач, порядок их выполнения, способы загрузки в память частично определяются в процессе проектирования.

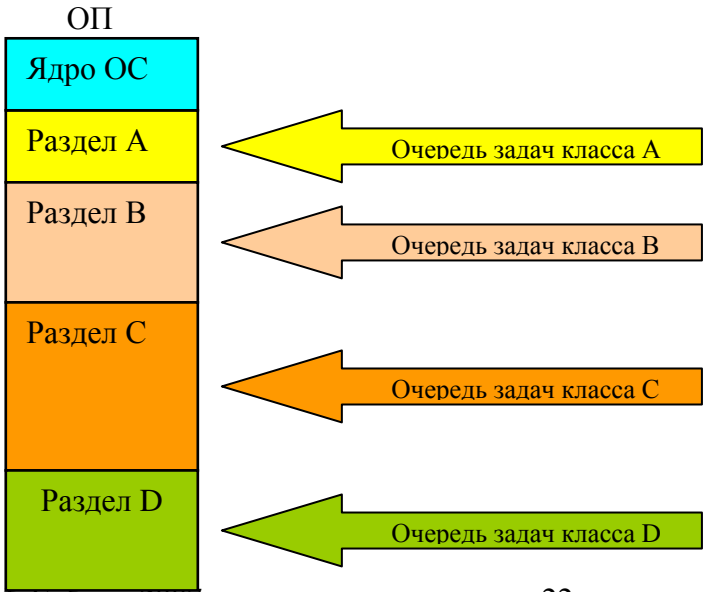


Рис. 18. Распределение ОП статическими разделами

Рассмотрим в качестве примера пакетную мультипрограммную ОС IBM OS360 MFT, в которой

была использована подобная стратегия управления памятью. ОП делилась на разделы фиксированного размера, которые для данной версии, установленной на конкретном компьютере, были постоянны (рис. 18). Каждый раздел соответствовал классу входных работ и все задачи всех заданий, принадлежащих этому классу, выполнялись в этом разделе ОП.

Каким же образом задания попадали в соответствующий класс?

В операторе JOB программист указывал параметр CLASS=A и в, соответствии со значением этого параметра, задача Системного ввода помещала это задание в очередь класса А. Программист мог указать и любой другой класс, в зависимости от характеристик своей задачи и информации о параметрах классов, которые существовали на используемом компьютере. Администрация вычислительного центра предоставляла такую информацию, характеризуя каждый класс размером раздела основной памяти и временем счета.

Такая организация системы позволяла сформировать хорошую смесь задач, поскольку задачи одного класса выполнялись последовательно, а задачи разных классов и создавали мультипрограммную смесь задач.

Стратегия распределения ОП статическими разделами также использовалась в ОС реального времени фирмы Digital Equipment Corporation (DEC), таких как RSX11 для компьютеров PDP11. В этом случае разделы имели имена и задачи загружались в соответствующие разделы. При создании задачи программист указывал, в какой раздел и как должна загружаться разрабатываемая задача. Эта информация хранилась в заголовке задачи и использовалась управляющей программой.

В настоящее время такой подход к управлению ОП используется во встроенных системах типа телефонов и другой аппаратуры.

Реализация функций управления ОП в таких системах достаточно проста.

Функция учета свободных и занятых разделов может быть реализована с помощью битового вектора. Каждый бит соответствует разделу и показывает, свободен раздел или занят.

Функция планирования выделения памяти для задачи связано с определением возможности загрузки задачи в раздел. В системах общего пользования к каждому разделу формируется очередь задач. Все задачи выполняются в порядке очереди. Момент времени выполнения определяется готовностью задачи, то есть задаче должны быть предоставлены необходимые для выполнения ресурсы.

Объем памяти, предоставляемый задаче, соответствует размеру раздела. Освобождается также вся память раздела.

Блокам памяти раздела, который отводился задаче, присваивался ключ задачи. Этот же ключ присваивался и другим ресурсам, выделенным задаче.

Основные **достоинства** стратегии распределения статическими разделами состоят в возможности реализации мультипрограммного режима работы и простоте реализации.

К **недостаткам** можно отнести неэффективное использование памяти, обусловленное различными размерами задач и неизвестной частотой их поступления. Если задачи одного класса поступают редко, то раздел, предназначенный для задач этого класса, будет использоваться плохо. Кроме того, для решения задач большого объема требуется соответствующий раздел. Но большие задачи встречаются более редко, чем маленькие задачи. Следовательно, раздел большого объема будет пустовать. Чтобы компенсировать этот недостаток, разрешается одной очереди задач приписывать два раздела. Однако память в разделе большого объема будет использоваться неэффективно при загрузке в него маленькой задачи.

Способ распределения статическими разделами можно использовать, если заранее известны размеры задач и частота их поступления в компьютерную систему.

Поскольку в раздел загружается все адресное пространство задачи, то редко используемые программы будут занимать память. Это также не улучшает использование памяти.

Распределение динамическими разделами

Стратегия управления ОП, распределяемой динамическими разделами, является весьма распространенной в нестраничных системах. При распределении динамическими разделами память выделяется по запросам. В запросе содержится объем требуемого участка памяти. Запросы выдаются программой управления задачами при создании очередной задачи. Система управления памятью при удовлетворении запроса находит свободный участок требуемого объема и создает раздел задачи. При завершении задачи этот раздел уничтожается.

Такой способ управления ОП использовался в ОС IBM OS360 MVT, в версиях UNIX для DEC PDP11. Также в DEC RSX11M можно было создать раздел, в котором использовался этот метод, что позволяло использовать эту ОС как диалоговую многопользовательскую систему.

На рис. 20 показано изменение состояния памяти в процессе поступления запросов.

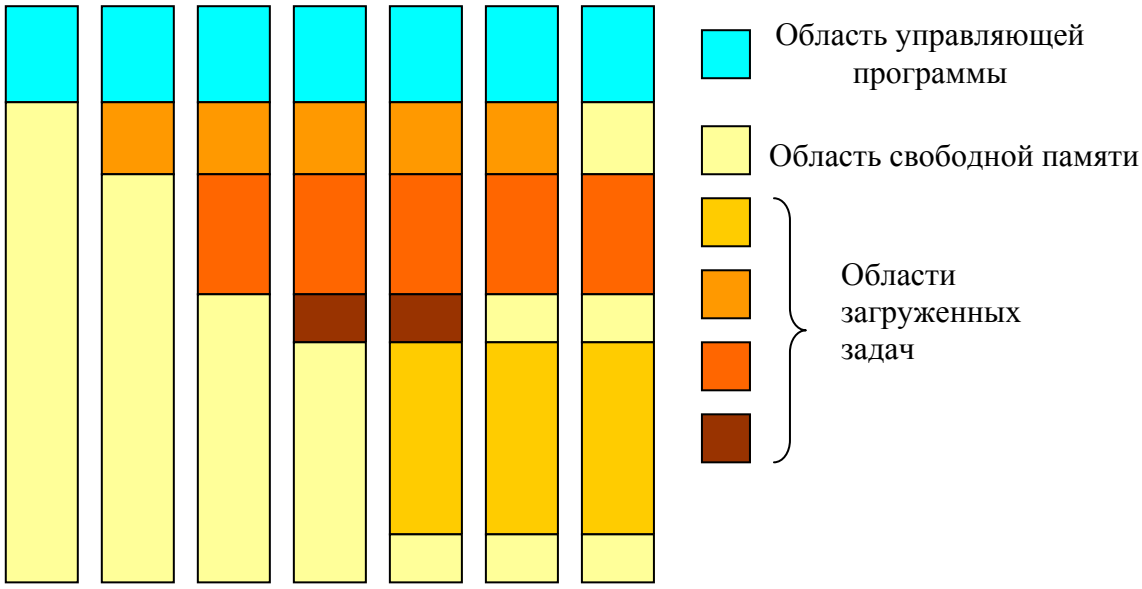


Рис. 20 Изменение состояния памяти при загрузке и удалении задач

При способе управления памятью, распределяемой динамическими разделами, необходимо учитывать свободные и занятые участки, то есть хранить информацию о **базовых адресах** и **размерах** этих участков памяти. Во многих ОС для этой цели используется **список блоков управления памятью**. В области управляющей программы по определенному адресу размещается указатель начала списка, который содержит ссылку на первый блок.

В начало каждого свободного участка помещается блок управления памятью (MCB). В простейшем случае этот блок должен содержать адрес следующего свободного участка и объем участка, которым он управляет. MCB последнего свободного участка содержит 0 в поле адреса следующего участка.

С этим списком работают функции ядра, занимающиеся выделением и освобождением памяти. Для освобождения памяти необходимо знать адрес и размер занятого участка. Если в ОС принят способ учета только свободных участков, то информация о занятых участках хранится в блоке управления задачей. При завершении задачи освобождается занятый участок, и эта информация используется для реорганизации списка блоков свободной памяти. Можно с помощью списка учитывать как

свободные, так и занятые участки памяти. Однако поиск свободного участка по такому списку занимает больше времени. На рис. 21 представлен список свободных блоков.

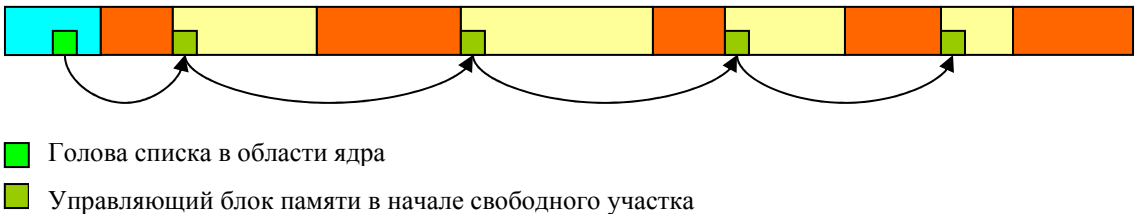
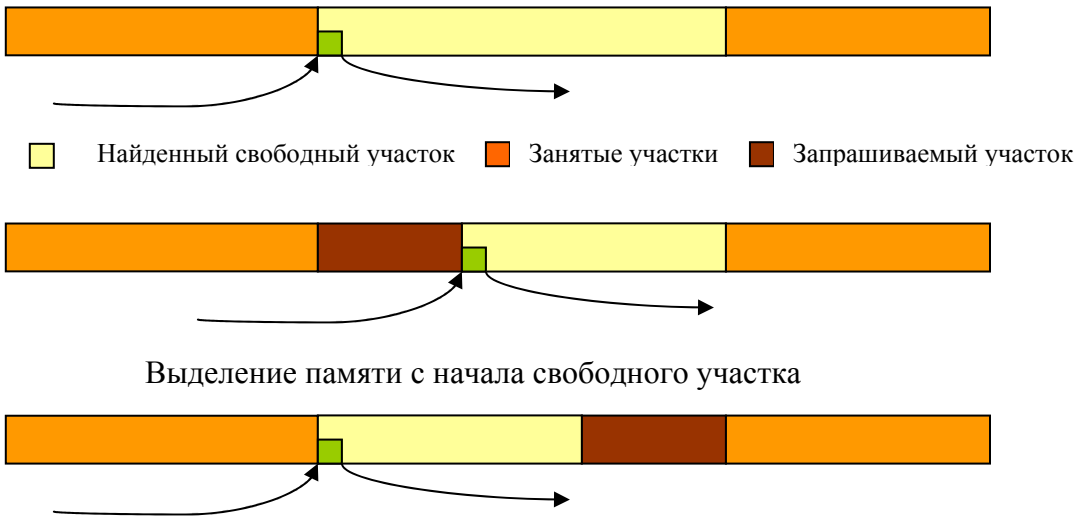


Рис. 21 Список свободных блоков

Планирование запросов на выделение памяти осуществляется функциями планирования задач в соответствии с реализованным в ОС алгоритмом. Так в первых версиях ОС UNIX право на загрузку в ОП получала задача дольше всех находившаяся в области свопинга. Функции планирования задач обращаются к загрузчику, который и выдает запрос на выделение участка памяти для очередной созданной задачи или загружаемой из области свопинга. Выполнение функций выделения и освобождения памяти связано с работой со списком.

Алгоритмы выделения памяти.

Для того, чтобы выделить участок памяти по запросу необходимо найти свободный участок, размер которого больше или равен размеру запроса. На рис. 22 показаны два способа выделения участка.



Выделение памяти с начала свободного участка

Выделение памяти с конца свободного участка

Рис. 22 Способы выделения участка

Можно выделять память с начала свободного участка. В этом случае придется изменять адрес предыдущей ссылки на новый свободный участок и формировать новый управляющий блок. Если отводить память под запрашиваемый раздел в конце свободного участка, то придется лишь изменить размер измененного свободного участка.

Существует три наиболее популярных алгоритма поиска свободного участка.

1. Алгоритм «Первый подходящий» (first fit)

Исходными данными для алгоритма служит размер запрашиваемого участка памяти Vp. Алгоритм просматривает список управляющих блоков свободных участков памяти и выбирает первый участок, чей размер Vi ≥ Vp.

Равенство размеров запрашиваемого и выделяемого участков памяти реализуется редко, поэтому остается свободный участок. В начале списка группируются мелкие свободные участки, а свободные участки больших размеров сдвигаются в конец списка.

С одной стороны это положительный фактор, поскольку запросы на малые участки удовлетворяются в одной области памяти, а на большие – в другой. Это уменьшает дробление памяти на мелкие участки – **фрагментацию**. С другой стороны увеличивается время поиска свободного участка.

2. Алгоритм «Самый подходящий» (the best fit)

Алгоритм просматривает список управляющих блоков свободных участков памяти и выбирает участок, чей размер $Vi \geq Vp$ и разность $Vi - Vp$ минимальна, то есть выбирается наиболее близкий по размеру свободный участок, чей размер превосходит размер запроса.

Список свободных участков может быть упорядочен по возрастанию, тогда алгоритм «Самый подходящий » будет работать как «Первый подходящий» с этим списком. Но в этом случае при образовании нового свободного участка требуется сортировать список по возрастанию размеров разделов.

Достоинства этого алгоритма состоят в следующем:

- если есть свободный участок, чей размер в точности равен размеру запроса, то этот участок будет выделен,
- свободные участки больших размеров остаются нетронутыми.

Недостатки метода:

- увеличивается вероятность образования свободных участков маленьких размеров, то есть увеличивается тенденция к фрагментации ОП,
- при образовании новых свободных участков необходимо сортировать список, меняя ссылки.

3. Алгоритм «Наименее подходящий» (the worst fit)

Алгоритм просматривает список управляющих блоков свободных участков памяти и выбирает участок, чей размер $Vi \geq Vp$ и разность $Vi - Vp$ максимальна, то есть выбирается максимальный по размеру свободный участок.

Если список организован таким образом, что первым в списке находится **максимальный свободный участок**, то удовлетворение запроса осуществляется чрезвычайно просто – выбирается первый участок в списке. Однако при образовании нового свободного участка необходимо в списке отыскивать максимальный свободный участок.

Достоинство этого алгоритма состоит в том, что остаток после выделения участка по запросу остается большим.

Освобождение памяти.

Освобождение занятого участка памяти происходит при удалении задачи из ОП в результате ее окончания или перемещения ее в область свопинга на диске. В этом случае возможны следующие ситуации:

- освобождаемый участок находится между свободными и в результате образуется большой свободный участок (рис. 23),
- освобождаемый участок находится между занятым и свободным участками (рис. 24),
- освобождаемый участок находится между занятыми участками (рис. 25).

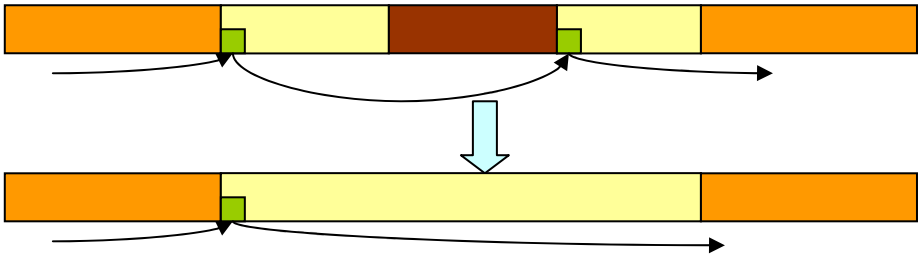


Рис. 23 Освобождение участка находящегося между двумя свободными

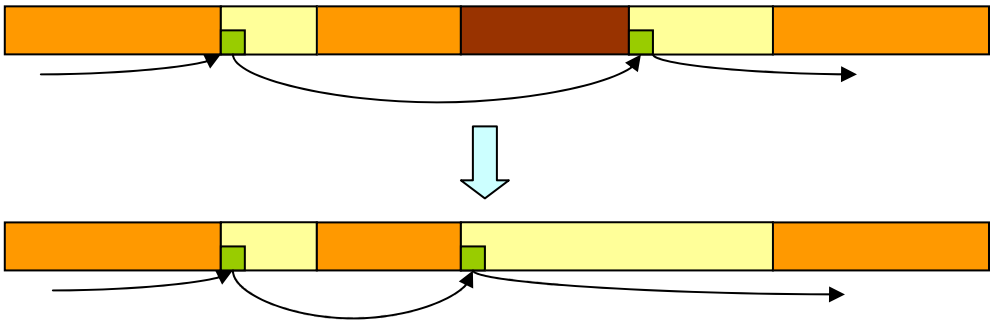


Рис. 24 Освобождение участка находящегося между свободным и занятым

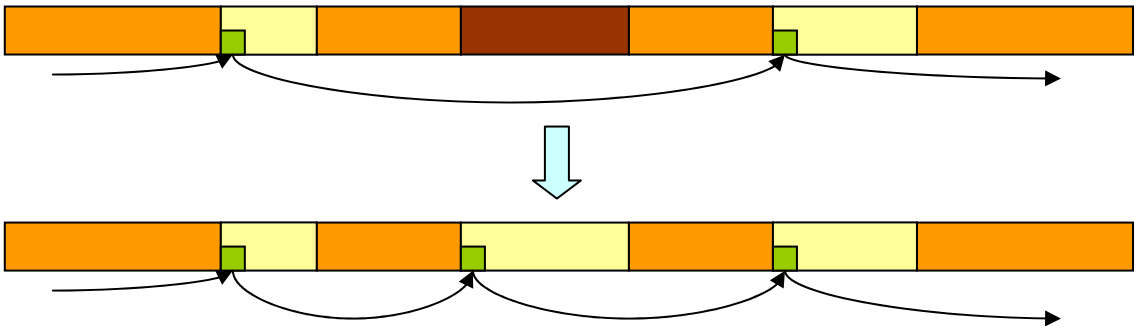


Рис. 25 Освобождение участка находящегося между двумя занятыми

Достоинства и недостатки динамического распределения памяти.

К достоинствам распределения памяти, управляемой динамическими разделами, относится следующее:

- обеспечивает мультипрограммный режим работы ОС,
- более эффективное использование памяти по сравнению со статическим распределением,
- алгоритмы достаточно просты,
- аппаратные затраты невелики.

Недостатки динамического распределения памяти:

- основной недостаток – **фрагментация памяти**,
- объем памяти, требуемый задаче, может быть настолько большим, что ее будет недостаточно, даже в отсутствии фрагментации,
- в память загружаются те части кода, которые используются весьма редко, например, функции обработки ошибок. Это характерно для всех методов, использующих загрузку всего логического адресного пространства в физическую память.

Проблема фрагментации при распределении динамическими разделами

Под фрагментацией понимают такое состояние памяти, при котором образуется много мелких свободных участков и создается ситуация, когда не существует непрерывного участка памяти, удовлетворяющего запросу задачи, но общий объем свободной памяти достаточно большой для загрузки задачи в ОП.

Различают **внешнюю** и **внутреннею** фрагментации.

Внешняя фрагментация создается в ОП между задачами.

Внутренняя фрагментация образуется в области задачи. Дело в том, что в область задачи загружаются кодовые сегменты, сегменты данных, образуется стек и куча, в некоторых системах в эту область грузятся системные таблицы задачи, методы доступа. Средства борьбы с внутренней фрагментацией располагаются в библиотечных функциях, которые работают с кучей, либо используется загрузка с противоположных концов раздела системной информации и сегментов прикладной задачи.

Для борьбы с внешней фрагментацией разрабатывались специальные методы управления ОП. Рассмотрим два из них: распределение перемещаемыми разделами и использование страничной памяти.

Распределение перемещаемыми разделами памяти.

Суть этого метода состоит в том, что все занятые разделы смещаются в одну область и образуется одна непрерывная область свободной памяти (рис. 26).

Однако в программах существуют адресно-зависимые элементы, чьи значения зависят от места расположения в ОП. При перемещении программного кода задач эти адреса должны быть откорректированы. Примерами таких адресно-зависимых элементов являются:

- значения базовых регистров,
- команды, содержащие прямые адреса памяти,
- адреса в списках параметров,
- указатели в структурах данных.

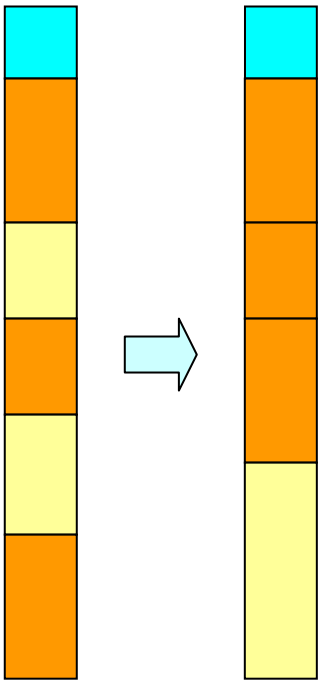


Рис. 26 Перемещение разделов в памяти

Замена значений адресно-зависимых элементов является достаточно трудоемкой операцией. Для этого необходимо хранить информацию об адресно-зависимых элементах, просматривать коды задач и корректировать их значения.

Поэтому количество типов адресно-зависимых элементов было сокращено и остались лишь значения базовых регистров. Это можно было осуществить, если сделать все адреса в области памяти задачи относительными. Тогда при перемещении задачи можно изменить только значения базовых регистров.

Такой метод использовался в системах DEC PDP10, UNIVAC 1108, HoneyWell 6000. Однако перемещение кода также является длительной операцией.

Достоинством метода является ликвидация фрагментации между задачами, что позволяет лучше использовать память, увеличить уровень мультипрограммирования и увеличить производительность системы.

Недостатками этого метода являются следующие:

- перекомпоновка памяти требует существенных накладных расходов,
- хотя фрагментация исключается, но часть памяти

теряется, так как даже после перемещения освободившаяся память может быть недостаточна для загрузки адресного пространства задачи,

- память занимают те части кода, которые используются редко.

Управление памятью, организованной страницами.

Идея отказа от загрузки логического адресного пространства задачи в непрерывный участок физической памяти эксплуатировалась достаточно давно, но свою завершенность она обрела с созданием страничной памяти. Все логическое адресное пространство делилось на логические страницы – участки адресов одинаковой длины. Физическая память также делилась на физические страницы. Размер логической страницы равен размеру физической страницы.

Этот метод требовал и существенной аппаратной поддержки в виде механизма страничного преобразования адресов. Адрес представляется в этом случае в виде двух полей: поля номера страницы и поля смещения на странице (рис. 27). Для каждой задачи, загруженной в ОП, создается **Таблица страниц задачи**. Каждый элемент таблицы со смещением P соответствует логической странице с номером P. Элемент страницы содержит базовый адрес физической страницы, в которой размещена логическая страница с номером P. Для того, чтобы получить **действительный адрес** для заданного **программного адреса** (P,D) аппаратура процессора по номеру страницы в поле адреса выбирает элемент таблицы и суммирует базовый адрес физической страницы, находящийся в этом элементе со смещением из поля смещения адреса.

Кроме того, необходимо учитывать свободные и занятые физические страницы ОП. Это можно реализовать с помощью битового вектора, где каждый бит соответствует физической странице.

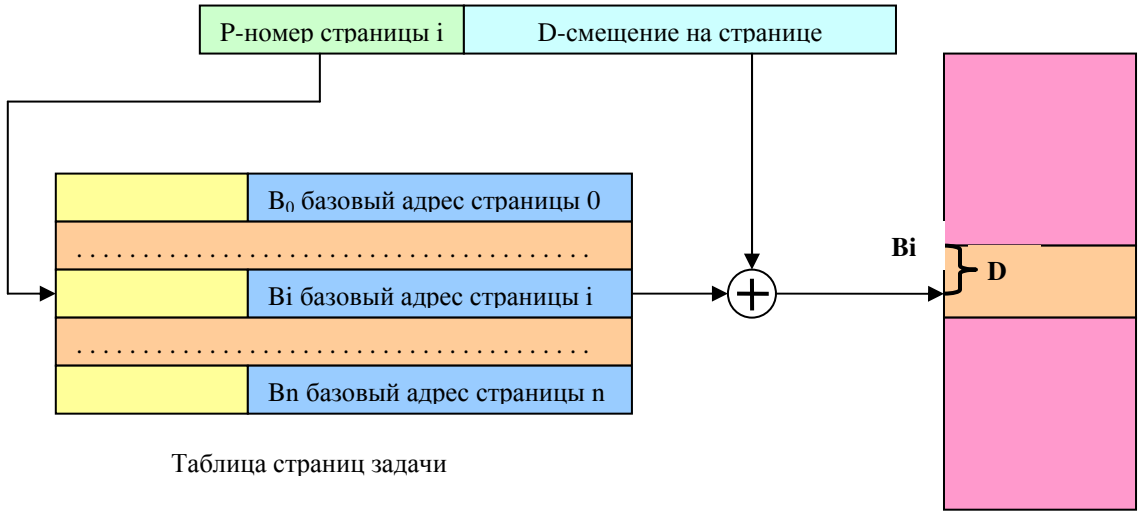


Рис. 27 Схема страничной адресации

Алгоритм работает следующим образом:

Отыскивается необходимое количество свободных физических страниц.

ЕСЛИ число свободных страниц меньше числа логических страниц задачи, ТО задача переводится в состояние ожидания.

ЕСЛИ свободных физических страниц достаточно, ТО логические страницы размещаются в ОП и формируется таблица страниц задачи. Физические страницы помечаются как занятые.

Достоинства метода:

- исключается внешняя фрагментация,
- отсутствуют накладные расходы, связанные с перекомпоновкой разделов.

Недостатки метода:

- остается внутренняя фрагментация, поскольку логическое адресное пространство не кратно размеру страницы и остается незаполненная последняя страница,
- часть физических страниц остается нераспределенной, так как после загрузки нескольких задач оставшаяся часть свободных физических страниц не удовлетворяет требованиям запросов других задач,
- в памяти размещается часть программного кода, которая используется крайне редко, поскольку загружается все логическое пространство задачи.

3.3 Управление виртуальной памятью

В случае не страничной организации памяти в ОП загружалось все логическое адресное пространство задачи. Однако нет необходимости иметь весь программный код в ОП, так как не все функции используются при решении конкретной задачи, существуют функции, которые используются крайне редко. Если загружать только ту часть кода, которая необходима в данный момент, то использование памяти будет более эффективным, увеличиться уровень мультипрограммирования, повышается производительность системы.

Идея загрузки только части адресного пространства задачи предполагает, что другая часть адресного пространства хранится во внешней памяти. Таким образом, требуется двухуровневая физическая память и основная память, в которой выполняются задачи, состоит из оперативной памяти и внешней памяти, в качестве которой используется память на диске. Для функций управляющей программы, системных и прикладных задач ОП выступает как **виртуальная память** большого размера. Функции ядра, которые реализуют управление памятью в этом случае и образуют тот слой программного обеспечения, который обеспечивает виртуализацию памяти.

Существуют следующие способы управления виртуальной памятью:

- управление виртуальной страничной памятью по запросам,
- управление сегментной виртуальной памятью,
- управление виртуальной сегментно-страничной памятью.

Управление виртуальной страничной памятью по запросам

Этот способ управления памятью очень популярен. Он использовался в системах

- MULTICS;
- IBM370 VS1, VS2, MVS, VM370;
- UNIVAC70/46 VMOS;
- MS WINDOWS;
- Unix, Linux.

Физическая память – двухуровневая, делится на физические страницы. Адресное пространство задачи также делится на логические страницы. Размеры логической и физической страниц совпадают. Реально использовались размеры страниц от 256 байт до 4 Кбайт.

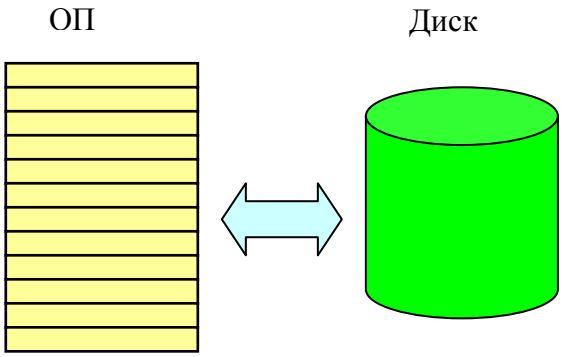


Рис. 28 Обмен страницами ОП и внешней памятью

При выполнении задачи только часть страниц находится в ОП. Остальные страницы находятся во внешней памяти. Если при выполнении программного кода на странице в ОП происходит обращение к программному коду на странице во внешней памяти,

то возникает **страничное** прерывание. Это прерывание происходит во время выполнения команды, при вычислении адреса операнда. Управление по этому прерыванию передается функциям ядра, которые находят нужную страницу во внешней памяти, отыскивают свободную страницу в ОП и подкачивают туда найденную страницу из внешней памяти. Затем повторяется прерванная команда. Если в ОП нет свободной страницы, то функции управления виртуальной страничной памятью выполняют замещение некоторой страницы в ОП. Для этого отыскивается в соответствии с **алгоритмом замещения** удаляемая страница и на ее место загружается страница из внешней памяти.

Таким образом, при вычислении действительного адреса может возникнуть обмен между ОП и внешней памятью. Поэтому организации внешней памяти для свопинга страниц должно уделяться определенное внимание. В первых ОС с виртуальной страничной памятью использовались специально разработанные внешние накопители информации, но с появлением быстродействующих и надежных жестких дисков стали использовать стандартные устройства компьютера. Однако в различных ОС организация области свопинга различна. Так в ОС Unix, Linux во время установки ОС на компьютер выделяется специальный раздел на диске, а в ОС MS WINDOWS используется системный файл **pagefile.sys**, который располагается на диске С. Конечно его можно переопределить на другой диск, а также задать минимальный и максимальный размеры, которые должны быть кратны 8. Под минимальный размер система выделит определенное количество смежных блоков памяти, а в случае необходимости будет выделять дополнительные блоки до максимального размера. Разумеется, в этом случае файл будет фрагментирован, что увеличивает время доступа к внешним страницам. Чтобы смягчить этот недостаток можно создать раздел в виде логического диска размером 1.5-2 Гбайта с именем отличным от “С”. Диск С многие программы используют по умолчанию. А можно просто изменить минимальный размер файла, сделав его равным максимальному.

Разумеется, такая организация управления памятью требует усложнения аппаратуры процессора, системных таблиц и алгоритмов функций управления.

Аппаратура процессора должна поддерживать механизм страничного преобразования адресов (стр. 27) и механизм страничного прерывания.

Учет состояния памяти.

Необходимы следующие системные таблицы:

1) **таблица физических страниц** – одна на систему. В таблице существует запись для каждой страницы в ОП. В записи отмечается состояние физической страницы. Возможны следующие состояния:

- физическая страница свободна и в нее может быть загружена страница из внешней памяти,
- физическая страница занята,
- физическая страница находится в **транзитном** состоянии.

Транзитное состояние физической страницы необходимо для того, чтобы предотвратить конфликты при выборе страницы. Предположим, что некоторая страница свободна и выбрана для загрузки в нее логической страницы задачи из внешней памяти. Операция обмена страниц достаточно длительная операция по сравнению со скоростью процессора, поэтому сделать ее непрерываемой невозможно. Это существенно увеличит накладные расходы, работа управляющей программы будет занимать слишком много времени. Если операция обмена страницами прерываема, то процессор на это время будет занят выполнением другой задачи и возможно для этой задачи также придется загружать страницу из внешней памяти. Состояние памяти к этому моменту не измениться, поэтому для другой задачи будет выбрана та же свободная страница.

Предположим, что страница занята и выбрана алгоритмом замещения для удаления, а на место этой страницы будет загружена страница из внешней памяти. В этом случае возникает такая же конфликтная ситуация выбора одной и той же физической страницы для двух задач, как и в предыдущем случае.

Чтобы избежать этой конфликтной ситуации, необходимо ввести транзитное состояние, которое будет присваиваться физической странице, когда страница выбрана для загрузки как свободная, или выбрана для выгрузки и последующей загрузки внешней страницы как занятая. Алгоритмы выбора страниц и замещения не рассматривают страницы в транзитном состоянии.

Также в записи таблицы физических страниц находится **бит обращения** к странице, который устанавливается, если к странице было обращение. Это необходимо для реализации аппроксимации алгоритма замещения LRU. Для удаления выбирается страница, к которой не было обращения, то есть бит обращения не установлен.

2) **Таблица страниц задачи** – одна на задачу. Таким образом, сколько создано в системе задач, столько будет создано и таблиц страниц.

Каждая запись в таблице страниц задачи содержит:

- бит состояния, который показывает, находится ли страница в ОП или во внешней памяти. Когда при обращении к странице соответствующая запись таблицы загружается на управляющий регистр процессора и бит состояния показывает отсутствие страницы в основной памяти, возникает страничное прерывание.
- базовый адрес страницы физической памяти, в которую загружена логическая страница.
- бит использования страницы, который устанавливается, если на страницу была произведена запись. В этом случае, при замещении страницу необходимо сохранить во внешней памяти. Если запись не производилась на страницу, то копия страницы хранится во внешней памяти и операция записи не нужна.

4) **Таблица внешних страниц задачи** – одна на задачу. Содержит адреса внешних страниц во внешней памяти.

Планирование запросов на выделение памяти.

Запрос на выделение памяти поступает в момент выполнения задачи и осуществляется по страничному прерыванию. Количество страниц, выделяемое задаче, зависит от алгоритма управления. Некоторые алгоритмы не загружают задачу, пока в ОП не будет достаточного количества страниц, называемого **рабочим множеством** страниц задачи.

Выделение памяти.

Если существуют свободные страницы, то они выделяются либо при начальной загрузке задачи в ОП, либо по страничному прерыванию.

Если свободные страницы отсутствуют, то необходимо выгрузить какую-либо страницу во внешнюю память, а на ее место загрузить требуемую страницу. Кандидат на выгрузку определяется в этом случае **алгоритмом замещения**.

Освобождение памяти.

В случае окончания задачи освобождается вся память, как внешние страницы, так и страницы в ОП. В процессе выполнения, страницы задачи могут быть выгружены во внешнюю память по алгоритму замещения.

Алгоритмы замещения страниц.

В случае возникновения страничного прерывания и отсутствия свободных физических страниц в ОП, выполняется алгоритм замещения, который выбирает страницу для удаления. В результате исследований выяснилось, что от применяемого алгоритма зависит производительность системы.

Число страничных прерываний служит оценкой эффективности работы алгоритмов управления памятью. Действительно, при возникновении страничного прерывания выполняется операция обмена с внешней памятью, которая требует существенных временных затрат. Поэтому лучше будет тот алгоритм, который обеспечивает минимальное число страничных прерываний.

Модель производительности алгоритмов замещения позволяет проанализировать свойства различных алгоритмов. Модель включает в себя:

- Траекторию страниц P – список страниц, в которых расположены используемые при выполнении программы адреса памяти.
- Размер ОП M, измеряемый в физических страницах.
- Алгоритм замещения.

Оценка производительности алгоритма, измеряемая как отношение f = F/|P|, где f – функция неудач, F – число страничных прерываний, |P| - число страниц в траектории P.

Естественно задать вопрос: «А существует ли оптимальный алгоритм? То есть такой, который для любой траектории страниц обеспечивал бы минимальное значение функции неудач»

1) Оптимальный алгоритм.

Оптимальный алгоритм заключается в том, что для удаления следует выбирать страницу, к которой дольше всего не будет обращения.

Пример применения оптимального алгоритма.

P	4	3	2	1	4	3	5	4	3	2	1	5
M	4	4	4	4	4	4	4	4	4	2	2	2
		3	3	3	3	3	3	3	3	3	1	1
			2	1	1	1	5	5	5	5	5	5
F	+	+	+	+			+			+	+	

F = 7 |P| =12 f=7/12

Цветом в таблице выделены страницы, которые удаляются алгоритмом замещения. Единственным недостатком оптимального алгоритма является **невозможность его реализации**. Узнать траекторию страниц до выполнения задачи **невозможно**. Как пишут некоторые авторы, если обладать такими возможностями, то лучше играть на бирже. То есть знал бы прикуп, жил бы в Сочи.

2) Алгоритм FIFO (first in – first out первый пришел – первым ушел).

По этому алгоритму выбирается страница, которая была загружена в ОП первой из всех присутствующих страниц.

Пример применения алгоритма FIFO.

P	4	3	2	1	4	3	5	4	3	2	1	5
M	4	4	4	1	1	1	5	5	5	5	5	5
		3	3	3	4	4	4	4	4	2	2	2
			2	2	2	3	3	3	3	3	1	1
F	+	+	+	+	+	+	+			+	+	

F=9 |P| =12 f=9/12

Этот алгоритм действует неразумно, поскольку удаляет страницы, которые тут же будут использоваться. Кроме того, он обладает и принципиальным недостатком – **аномалией страниц**.

Если увеличить объем ОП, то естественно ожидать уменьшения числа страничных прерываний. Однако для алгоритма FIFO можно подобрать пример траектории, для которой при увеличении размера ОП число страничных прерываний возрастает.

P	4	3	2	1	4	3	5	4	3	2	1	5
M	4	4	4	4	4	4	5	5	5	5	1	1
		3	3	3	3	3	3	4	4	4	4	5
			2	2	2	2	2	2	3	3	3	3
				1	1	1	1	1	1	2	2	2
F	+	+	+	+			+	+	+	+	+	+

F=10 |P| =12 f=10/12

3) Алгоритм LRU (the least recently used самую давно используемую страницу).

По этому алгоритму выбирается страница для удаления, которая дольше всего не использовалась из всех присутствующих страниц, то есть к этой странице дольше всего не было обращений.

Пример применения алгоритма LRU.

P	4	3	2	1	4	3	5	4	3	2	1	5
M	4	4	4	1	1	1	5	5	5	2	2	2
		3	3	3	4	4	4	4	4	4	1	1
			2	2	2	3	3	3	3	3	3	5
F	+	+	+	+	+	+	+			+	+	+

F=10 | P | =12 f=10/12

Для примера используемой траектории страниц на начальном участке работа этого алгоритма совпадает с работой алгоритма FIFO, но в дальнейшем алгоритм ведет себя иначе. Результат получился хуже для этой траектории, но в алгоритме LRU отсутствует аномалия страниц.

P	4	3	2	1	4	3	5	4	3	2	1	5
M	4	4	4	4	4	4	4	4	5	5	1	1
		3	3	3	3	3	3	3	4	4	4	5
			2	2	2	2	2	2	3	3	3	3
				1	1	1	5	5	1	2	2	2
F	+	+	+	+			+			+	+	+

F=8 | P | =12 f=8/12

Если сравнить соответствующие столбцы этих двух таблиц, то можно заметить, что множество страниц столбца i при трех физических страницах включается в множество страниц столбца i при четырех физических страницах. Следовательно, невозможна ситуация, когда для страницы i траектории для варианта с тремя физическими страницами не возникало страничного прерывания, а для той же страницы i траектории с четырьмя физическими страницами страничное прерывание возникло.

Алгоритм замещения страниц LRU также трудно реализовать в чистом виде. Для того чтобы находить страницу, к которой дольше всего не было обращения, необходимо хранить время последнего обращения к странице. Затем при поиске кандидата на удаление следует сравнивать времена всех страниц. Ясно, что это связано с большими накладными расходами. Учитывая, что страничные прерывания возникают при вычислении действительного адреса в команде, а выполняется в среднем лишь одна пятая часть всех команд, расположенных на странице, то прерывания возникают довольно часто. Поэтому используется в реальных системах некоторая аппроксимация алгоритма LRU.

Бит обращения в таблице физических страниц позволяет отличить страницы, к которым было обращение от страниц, к которым не было обращения за определенный отрезок времени. В начальный момент биты обращений всех страниц устанавливаются в 0. В процессе работы если к странице было обращение, то бит выставляется в 1. Если все биты обращений выставлены в 1, то они сбрасываются в 0. Таким образом, за отрезок времени, пока все биты обращений не выставлены в 1, для удаления выбирается одна из страниц с нулевым битом обращения.

Пробуксовка (thrashing).

В страничных системах наблюдается еще одна неприятная проблема, которая получила название «пробуксовка». Это такое состояние системы, когда идет интенсивный обмен страницами между ОП и внешней памятью, а процессор простаивает, так как все задачи находятся в состоянии ожидания завершения обменом страниц. Каждая задача ждет, когда же будет закачана необходимая ей страница.

Эта ситуация не зависит от применяемого алгоритма замещения страниц, а связана с неограниченным уровнем мультипрограммирования. Количество задач может быть достаточно велико, а объем страниц, выделяемый для одной задачи, соответственно, небольшим. В этом случае часто происходят обращения к страницам, находящимся во внешней памяти. Это и приводит к экспоненциальному росту страничных прерываний.

Стратегия рабочего множества позволяет преодолеть этот принципиальный недостаток. В основе этой стратегии лежит следующая мысль: если в ОП присутствуют те страницы, к которым задача обращается часто, то обмен страницами будет слабым.

Рабочее множество – это такое множество страниц, которое нужно держать в ОП, чтобы в течение достаточно большого промежутка процессорного времени не произошло страничного прерывания.

Определенное количество страниц выделяется в системе в качестве рабочего множества. Если количество свободных страниц меньше рабочего множества, то задача не загружается в ОП. Это приводит к снижению уровня мультипрограммирования и потерям в использовании памяти, если число свободных страниц меньше, чем рабочее множество страниц какой-либо задачи. Система может корректировать рабочие множества страниц задач, оценивая использование страниц в процессе выполнения. Но это приводит к накладным расходам.

При разработке таких систем руководствуются **принципами локальности обращений к памяти в программах**. Различают локальность во **времени** и локальность в **пространстве**.

Локальность во времени состоит в том, что если к данному слову в памяти произошло обращение, то в ближайшее время к этому же слову в памяти опять произойдет обращение.

Локальность в пространстве состоит в том, что если к данному слову в памяти произошло обращение, то в ближайшее время с большой вероятностью произойдет обращение к соседним словам.

Системы программирования должны стараться повысить уровень локальности при построении загрузочного модуля.

Достоинства метода управления виртуальной страничной памятью по запросам следующие:

- большая виртуальная память,
- более эффективное использование ОП, за счет загрузки только части адресного пространства задачи,
- неограниченное мультипрограммирование.

Недостатки метода:

- накладные расходы возрастают,
- пробуксовка ограничивает уровень мультипрограммирования, и ухудшает использование памяти.

Метода управления виртуальной страничной памятью по запросам очень популярен и используется во многих ОС.

Управление виртуальной сегментной памятью по запросам

В ранее рассмотренных методах логическое адресное пространство задачи рассматривалось как односегментное, то есть не разделялась память под программы и данные, не выделялись отдельные структурные единицы данных и кода, такие как функции, массивы, другие структуры данных, объекты классов, то есть те, с которыми имеет дело программист. Эти единицы данных и кода представляются как сегменты. То есть сегмент – это **логическая** группа информации. Обмен между ОП и внешней памятью осуществляется сегментами. Сегменты имеют **переменную длину** и это обстоятельство вызывает большие сложности при управлении виртуальной сегментной памятью.

Такой способ управления памятью использовался в OS/2.

Программный адрес при сегментной организации логического адресного пространства имеет вид (S, D), где S – номер сегмента, а D – смещение на сегменте. Сегмент имеет переменную длину, поскольку смещение D занимает определенное число разрядов **n** в адресе, то размер сегмента может быть не более чем 2^n-1, то есть размер сегмента ограничен.

Организация физической памяти также двухуровневая. Так как сегмент имеет переменную длину, то управление ОП и внешней памятью усложняются.

Аппаратные средства процессора должны поддерживать механизм динамического сегментного преобразования адресов (рис. 29) и защиту сегментов.

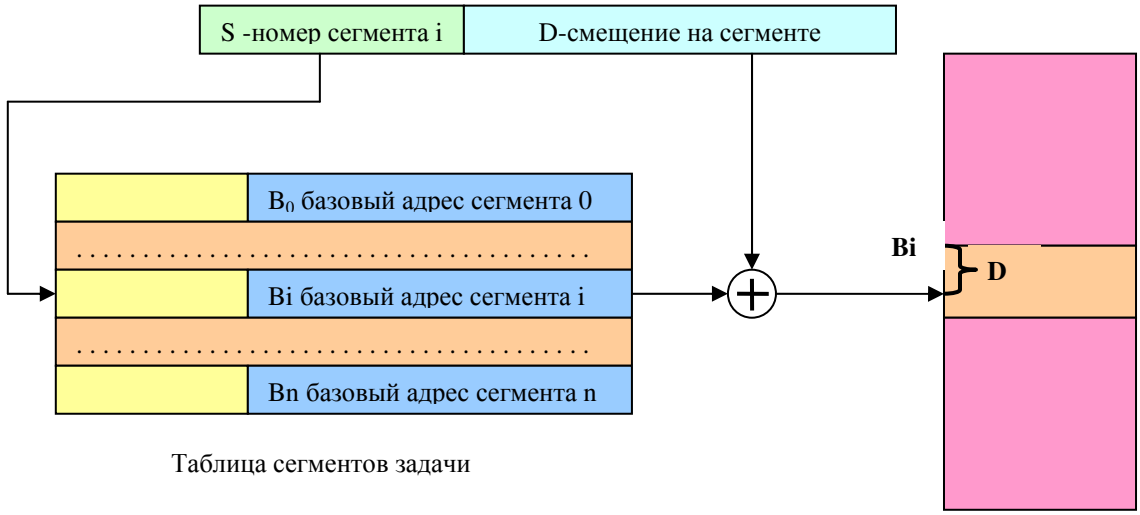


Рис. 29 Схема сегментной адресации

Учет состояния памяти.

1) Для каждой задачи в момент ее создания образуется **таблица сегментов**. Номеру сегмента в программном адресе соответствует **элемент** таблицы сегментов, который содержит следующие поля:

- базовый адрес сегмента **Bi** в ОП,
- бит состояния, показывающий находится сегмент в ОП или во внешней памяти,
- бит использования, показывающий производилась ли запись в сегмент,
- длина сегмента.

Бит состояния вызывает **сегментное** прерывание, если сегмент, содержащий адрес, к которому произошло обращение, находится во внешней памяти. Бит использования позволяет не переписывать сегмент во внешнюю память, если в сегмент не происходило записи. Длина сегмента позволяет контролировать адресное пространство сегмента.

2) Учет **свободных участков ОП** может быть выполнен с помощью списка свободных блоков (рис. 21), который использовался в методах управления ОП, выделяемой динамическими разделами. Начало списка находится в ядре ОС. Поскольку сегменты имеют переменную длину, то возникает аналогичная проблема фрагментации памяти. Методом борьбы с фрагментацией в этом случае является откачка подходящего сегмента во внешнюю память.

3) Учет **внешних сегментов задачи** также выполняется для каждой задачи и в таблице внешних сегментов содержатся адреса сегментов во внешней памяти.

Планирование запросов на выделение памяти.

Запрос на выделение памяти поступает либо в момент активизации задачи, либо в момент выполнения задачи и осуществляется по сегментному прерыванию. Для задачи выделяется участок ОП размером, достаточным для загрузки сегмента.

Выделение и освобождение памяти.

Для выделения памяти необходимо найти свободный участок, достаточный для размещения сегмента задачи. Для этого могут быть использованы те же алгоритмы, что и для управления памятью, выделяемой динамическими разделами. Если свободного участка достаточного размера нет, то выбирается сегмент внешней памяти для замещения.

Освобождение памяти осуществляется по окончанию задачи и при замещении сегмента.

Достоинства метода управления виртуальной сегментной памятью состоят в следующем:

- ликвидация фрагментации,
- большая виртуальная память,
- более эффективное использование памяти за счет загрузки только используемых сегментов,
- понятие сегмента более близко к логической структуре программы, следовательно, это упрощает разработку систем программирования.

Недостатки метода:

- большие накладные расходы и сложности связанные с управлением ОП и внешней памятью, поскольку сегменты имеют переменную длину,
- размеры сегмента ограничены,
- наблюдается явление пробуксовки,
- большие размеры сегментов препятствуют эффективному использованию памяти.

Управление виртуальной сегментно-страничной памятью

Комбинация методов управления виртуальной сегментной и страничной памятью позволяет устранить основной недостаток сегментной памяти – переменный размер сегментов и сохранить преимущества сегментной организации. Это достигается тем, что логическое адресное пространство состоит из сегментов, а обмен между ОП и внешней памятью осуществляется страницами.

Для того чтобы реализовать этот способ необходимо разбить логическое адресное пространство задачи на сегменты, а каждый сегмент на страницы. Обмен между ОП и внешней памятью осуществляется страницами таким же способом, как и в виртуальной страничной памяти. Поэтому в ОП может быть загружена только часть сегмента задачи. Таким образом, многосегментное адресное пространство задачи отображается на страничную память.

Программный адрес в этом случае состоит из трех полей (S,P,D), где S – **номер сегмента** задачи, P – **номер страницы сегмента** и D – смещение на странице. Аппаратные средства процессора по программному адресу динамически вычисляют действительный адрес памяти. Схема динамической переадресации представлена на рис.30. Как видно из этой схемы при вычислении действительного адреса по таблице сегментов задачи находится базовый адрес таблицы страниц сегмента, а затем по таблице страниц сегмента определяется базовый адрес страницы в ОП. Таким образом, накладные расходы возрастают.

Процессор поддерживает два прерывания: **сегментное** и **страничное**. Сегментное прерывание возникает, когда **таблица страниц** данного сегмента отсутствует в памяти, а страничное – когда **страница** отсутствует в ОП.

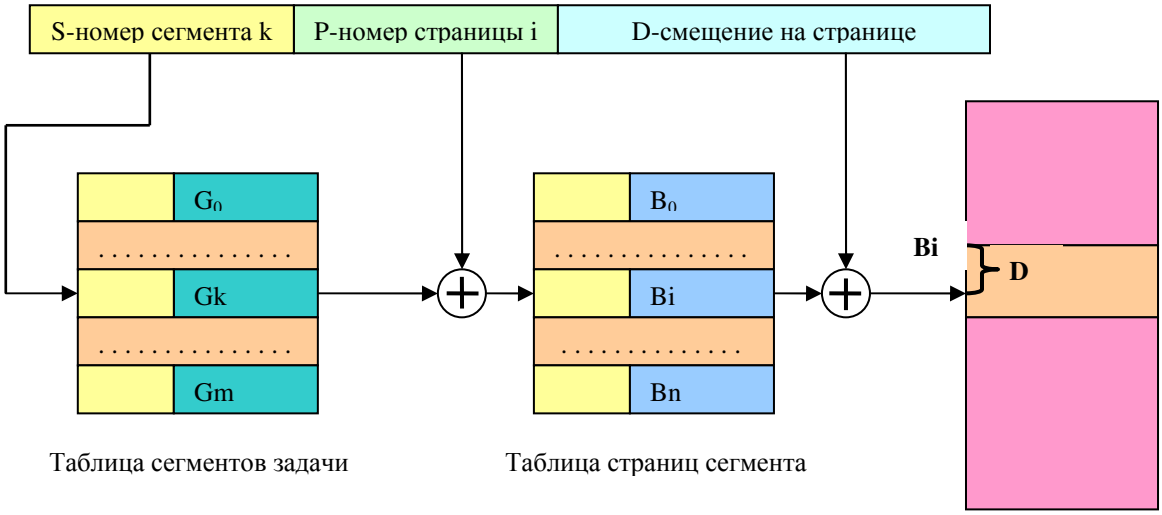


Рис. 30 Схема сегментно-страничной адресации

Учет состояния памяти.

1) **Таблица сегментов задачи** связывает сегменты и страницы, принадлежащие сегментам. Каждый элемент таблицы содержит базовый адрес таблицы страниц данного сегмента. Кроме того, элемент таблицы сегментов должен содержать бит состояния, который показывает, загружена или нет таблица страниц данного сегмента. Если таблица страниц не загружена в ОП, то возникает сегментное прерывание. Обработка этого прерывания завершается загрузкой таблицы страниц сегмента. Таблицы страниц сегментов могут также располагаться на страницах и участвовать в страничном обмене.

Кроме того, элемент таблицы сегментов может содержать поля, связанные с использованием таблицы страниц сегмента и полномочиями доступа к сегменту, то есть структура элемента таблицы сегментов аналогична структуре таблицы сегментов при стратегии управления с сегментным распределением, но вся информация относится к таблице страниц сегмента.

2) **Таблица страниц сегмента** устроена так же, как и таблица страниц при стратегии управления страничной памятью с замещением страниц. Таким образом, в системе могут возникать два типа прерываний при обращении к памяти: страничное и сегментное.

Поскольку обмен на физическом уровне осуществляется страницами, то необходимы таблица физических страниц и таблицы внешних страниц для каждого сегмента.

Планирование запросов при выполнении задачи связано с обработкой страничных прерываний, когда требуемая страница отсутствует, и обработкой сегментных прерываний, когда отсутствует в ОП таблица страниц сегмента. Планирование запросов и выделение и освобождение памяти осуществляется, так же как и в случае управления виртуальной страничной памятью. Используются те же алгоритмы замещения и стратегия рабочего множества.

Достоинства метода управления виртуальной сегментно-страничной памятью состоят в следующем:

1) Большая память, предоставляемая пользователям за счет реализации идеи виртуальной памяти;
2) эффективное использование ОП за счет загрузки только необходимой части адресного пространства;
3) нет ограничений на размер сегмента;
4) отсутствуют недостатки, вызванные использованием сегментов переменной длины;
5) как и в стратегии управления с сегментным распределением, можно поддерживать различные виды доступа к сегментам и защиты сегментов.

Таким образом, основные недостатки стратегии управления с сегментным распределением удалось преодолеть, сохранив преимущества.

Однако недостатки стратегии управления страничной памятью с замещением страниц остались. К ним относится явление "пробуксовки". Другие недостатки, связанные с усложнением аппаратного обеспечения, затратами памяти под системные таблицы, усложнением программного обеспечения, возросли. Тем не менее, эта стратегия использовалась в нескольких ОС, разработанных для IBM 370, которые нашли широкое применение. Использование этой стратегии возможно только в больших и мощных компьютерных системах.

Тема 4. Управление процессами и ресурсами в ОС

(в разработке)

Тема 5. Управление устройствами в ОС

(в разработке)

Тема 6. Управление вводом-выводом в ОС

(в разработке)

Тема 7. Управление данными в ОС

(в разработке)

Тема 8. Компьютерные сети

Коммуникационные сети могут быть разделены на два основных типа:

- с коммутацией каналов
- коммутацией пакетов.

Сети **с коммутацией каналов** работают, образуя выделенное соединение(канал) между двумя точками. Телефонная сеть США использует технологию с коммутацией каналов - телефонный вызов устанавливает канал от вызывающего телефона через локальную АТС, по линиям связи, к удаленной АТС, и, наконец, к отвечающему телефону. Пока существует канал, телефонное оборудование постоянно опрашивает микрофон, кодирует полученное значение в цифровой форме, и передает его по этому каналу к получателю. Отправителю гарантируется, что опросы будут доведены и воспроизведены, так как канал обеспечивает скорость 64 Кбит/с, которой достаточно для передачи оцифрованного голоса. Преимущество коммутации каналов заключается в ее гарантированной пропускной способности: как только канал создан, ни один сетевой процесс не уменьшит пропускной способности этого канала. Недостатком при коммутации каналов является ее стоимость: платы за каналы являются фиксированными и независимыми от траффика. Например, можно заплатить за телефонный вызов, даже если две разговаривающие стороны вообще ничего не говорили.

Сети с **коммутацией пакетов**, тип обычно используемый при соединении компьютеров, используют совершенно другой подход. В сетях с коммутацией пакетов траффик сети делится на небольшие части, называемые пакетами, которые объединяются в высокоскоростных межмашинных соединениях. Пакет, который обычно содержит только несколько сотен байтов данных, имеет идентификатор, который позволяет компьютерам в сети узнавать, предназначен ли он им, и если нет, то помогает им определить, как послать его в указанное место назначения. Например, файл, передаваемый между двумя машинами, может быть разбит на большое число пакетов, которые посылаются по сети по одному. Оборудование сети доставляет пакеты к указанному месту назначения, а сетевое программное обеспечение собирает пакеты опять в один файл. Главным преимуществом коммутации пакетов является то, что большое

число соединений между компьютерами может работать одновременно, так как межмашинные соединения разделяются между всеми парами взаимодействующих машин. Недостатком ее является то, что по мере того как возрастает активность, данная пара взаимодействующих компьютеров получает все меньше сетевой пропускной способности. То есть, всякий раз, когда сеть с коммутацией пакетов становится перегруженной, компьютеры, использующие сеть, должны ждать, пока они не смогут послать следующие пакеты.

Модель взаимодействия открытых систем OSI

OSI (Open Systems Interconnection) - *моделью взаимодействия открытых систем.*

Разработана *Международной Организацией по Стандартам* ISO (International Organization for Standardization). Модель описывает средства сетевого взаимодействия, которые делятся на семь уровней, для которых определены стандартные названия и функции.

Основные функции уровней модели OSI.

Физический уровень выполняет передачу битов по физическим каналам, таким, как коаксиальный кабель, витая пара или оптоволоконный кабель. На этом уровне определяются характеристики физических сред передачи данных и параметров электрических сигналов.

Канальный уровень обеспечивает передачу кадра данных между любыми узлами в сетях с типовой топологией либо между двумя соседними узлами в сетях с произвольной топологией. В протоколах канального уровня заложена определенная структура связей между компьютерами и способы их адресации. Адреса, используемые на канальном уровне в локальных сетях, часто называют MAC-адресами.

Сетевой уровень обеспечивает доставку данных между любыми двумя узлами в сети с произвольной топологией, при этом он не берет на себя никаких обязательств по надежности передачи данных.

Транспортный уровень обеспечивает передачу данных между любыми узлами сети с требуемым уровнем надежности. Для этого на транспортном уровне имеются средства установления соединения, нумерации, буферизации и упорядочивания пакетов.

Сеансовый уровень предоставляет средства управления диалогом, позволяющие фиксировать, какая из взаимодействующих сторон является активной в настоящий момент, а также предоставляет средства синхронизации в рамках процедуры обмена сообщениями.

Уровень представления. В отличии от нижележащих уровней, которые имеют дело с надежной и эффективной передачей битов от отправителя к получателю, уровень представления имеет дело с внешним представлением данных. На этом уровне могут выполняться различные виды преобразования данных, такие как компрессия и декомпрессия, шифровка и дешифровка данных.

Прикладной уровень - это в сущности набор разнообразных сетевых сервисов, предоставляемых конечным пользователям и приложениям. Примерами таких сервисов являются, например, электронная почта, передача файлов, подключение удаленных терминалов к компьютеру по сети.

Физический уровень	Транспортный	Сеансовый уровень
Канальный уровень		Уровень представления
Сетевой уровень		Прикладной уровень
Аппаратура	**Связь между протоколами**	**Операционная система и прикладные программы**

При построении транспортной подсистемы наибольший интерес представляют функции физического, канального и сетевого уровней, тесно связанные с используемым в данной сети оборудованием: **сетевыми адаптерами**, **концентраторами**, **мостами**, **коммутаторами**, **маршрутизаторами**. Функции *прикладного и сеансового* уровней, а также уровня *представления* реализуются операционными системами и системными приложениями конечных узлов. Транспортный уровень выступает посредником между этими двумя группами протоколов.

Аппаратные средства компьютерных сетей

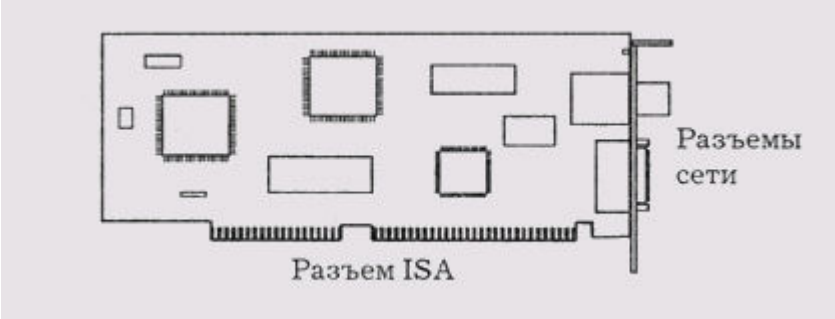
Как физически соединить компьютеры, чтобы можно было с одного на другой передавать информацию?

Сетевые адаптеры (сетевые карты, сетевая плата Network Interface Card, NIC*)* - физическое устройство, которое сопрягается по интерфейсам на системной плате с компьютером, с другой стороны со средой передачи. Функции сетевого адаптера:

1) Оформление передаваемой информации в виде кадра определенного формата. Кадр включает несколько служебных полей, среди которых имеется *адрес компьютера назначения* и *контрольная сумма кадра*, по которой сетевой адаптер станции назначения делает вывод о корректности доставленной по сети информации.
2) Получение доступа к среде передачи данных. В локальных сетях в основном применяются разделяемые между группой компьютеров каналы связи (общая шина, кольцо), *доступ к которым предоставляется по специальному алгоритму* (наиболее часто применяются метод случайного доступа или метод с передачей маркера доступа по кольцу). В последних стандартах и технологиях локальных сетей наметился переход от использования разделяемой среды передачи данных к использованию индивидуальных каналов связей компьютера с коммуникационными устройствами сети, как это всегда делалось в телефонных сетях, где телефонный аппарат связан с коммутатором АТС индивидуальной линией связи. Технологиями, использующими индивидуальные линии связи, являются 100VG-AnyLAN, ATM и коммутирующие модификации традиционных технологий - switching Ethernet, switching Token Ring и switching FDDI. При использовании индивидуальных линий связи в функции сетевого адаптера часто входит установление соединения с коммутатором сети
3) Кодирование последовательности бит кадра последовательностью электрических сигналов при передаче данных и декодирование при их приеме. Кодирование должно обеспечить передачу исходной информацию по линиям связи с определенной *полосой пропускания* и определенным *уровнем помех* таким образом, чтобы принимающая сторона смогла распознать с высокой степенью вероятности посланную информацию. Так как в локальных сетях используются широкополосные кабели, то сетевые адаптеры не используют модуляцию сигнала, необходимую для передачи дискретной информации по узкополосным линиям связи (например, телефонным каналам тональной частоты), а передают данные с помощью импульсных сигналов. Представление же двоичных 1 и 0 может быть различным.
4) Преобразование информации из параллельной формы в последовательную и обратно. Эта операция связана с тем, что для упрощения проблемы синхронизации сигналов и удешевления линий связи в вычислительных сетях информация

передается в последовательной форме, бит за битом, а не побайтно, как внутри компьютера.

5) Синхронизация битов, байтов и кадров. Для устойчивого приема передаваемой информации необходимо поддержание постоянного синхронизма приемника и передатчика информации. Сетевой адаптер использует для решения этой задачи специальные методы кодирования, не использующие дополнительной шины с тактовыми синхросигналами. Эти методы обеспечивают периодическое изменение состояния передаваемого сигнала, которое используется тактовым генератором приемника для подстройки синхронизма. Кроме синхронизации на уровне битов, сетевой адаптер решает задачу синхронизации и на уровне байтов, и на уровне кадров.

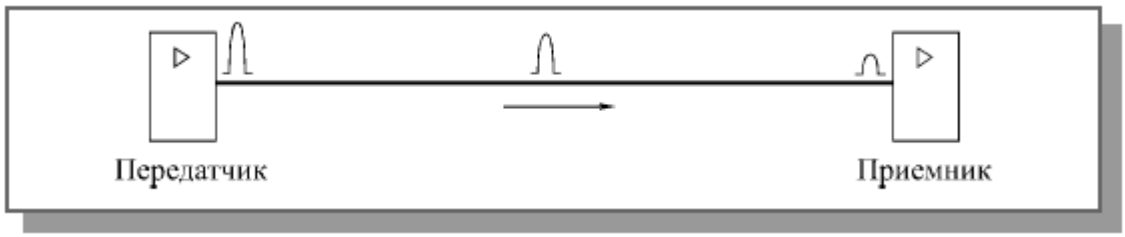


Среда передачи:

- коаксиальный кабель,
- витая пара,
- оптоволоконный кабель,
- радиотракт.

Драйвер сетевого адаптера – программа, которая управляет сетевым адаптером.
Как всякое устройство, сетевой адаптер подключен к определенным портам компьютера и ему отведено определенное прерывание. Когда приходит кадр, предназначенный для данного компьютера, сетевой адаптер инициирует прерывание и управление передается программе обработки этого прерывания, которая и инициирует работу драйвера. Именно благодаря *драйверу* ПО компьютера может не знать никаких аппаратных особенностей адаптера. Сетевые *драйверы*, поставляемые вместе с сетевыми адаптерами, позволяют сетевым программам одинаково работать с платами разных поставщиков и даже с платами разных локальных сетей (Ethernet, Arcnet, Token-Ring и т.д.). Если говорить о стандартной модели OSI, то *драйверы*, как правило, выполняют функции канального уровня, хотя иногда они реализуют и часть функций сетевого уровня

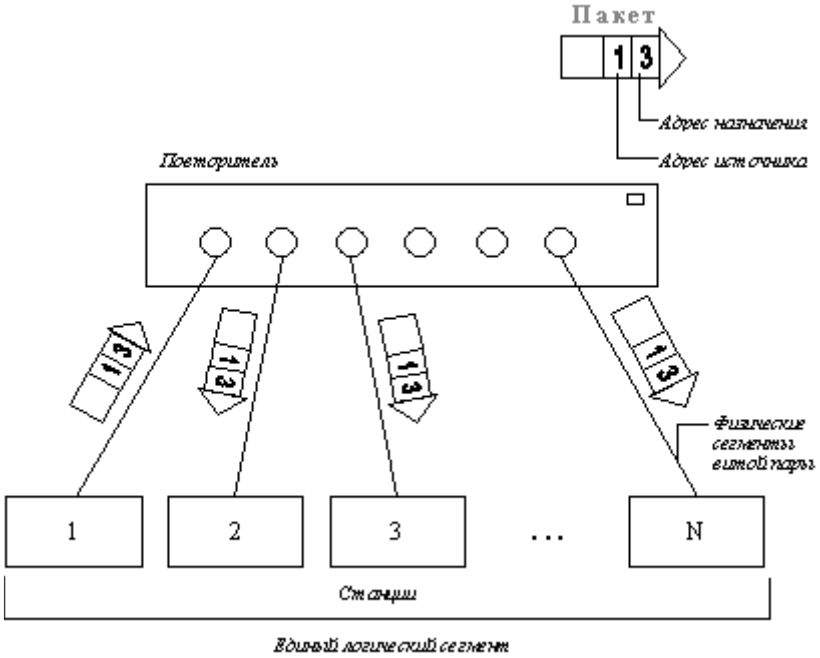
Физическая структура локальной сети.



Затухание сигнала при распространении по сети.

Повторители и концентраторы.

Основная функция *повторителя* (repeater), как это следует из его названия - повторение сигналов, поступающих на один из его портов, на всех остальных портах (Ethernet) или на следующем в логическом кольце порте (Token Ring, FDDI) синхронно с сигналами-оригиналами. Повторитель улучшает электрические характеристики сигналов и их синхронность, и за счет этого появляется возможность увеличивать общую длину кабеля между самыми удаленными в сети станциями.



Повторитель Ethernet синхронно повторяет биты кадра на всех своих портах

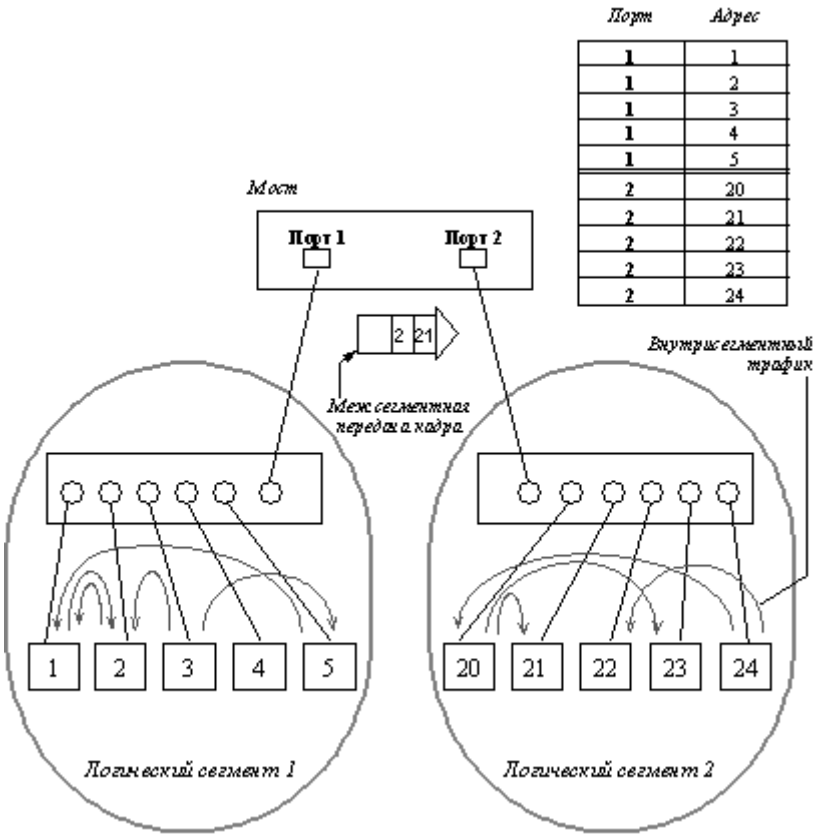
Многопортовый повторитель часто называют *концентратором* (hub, concentrator), что отражает тот факт, что данное устройство реализует не только функцию повторения сигналов, но и концентрирует в одном центральном устройстве функции объединения компьютеров в сеть. Практически во всех современных сетевых стандартах концентратор является необходимым элементом сети, соединяющим отдельные компьютеры в сеть.

Коллективное использование многими компьютерами общей кабельной системы в режиме разделения времени приводит к существенному снижению производительности сети при интенсивном трафике. Общая среда перестает справляться с потоком передаваемых кадров и в сети возникает очередь компьютеров, ожидающих доступа. Это явление характерно для всех технологий, использующих разделяемые среды передачи данных, независимо от используемых алгоритмов доступа (хотя наиболее страдают от перегрузок трафика сети Ethernet с методом случайного доступа к среде).

Поэтому сети, построенные на основе концентраторов, не могут расширяться в требуемых пределах - при определенном количестве компьютеров в сети или при появлении новых приложений всегда происходит насыщение передающей среды, и задержки в ее работе становятся недопустимыми. Эта проблема может быть решена путем логической структуризации сети с помощью мостов, коммутаторов и маршрутизаторов.

Мосты и коммутаторы

Мост (bridge), а также его быстродействующий функциональный аналог - *коммутатор* (switching hub), делит общую среду передачи данных на логические сегменты. Логический сегмент образуется путем объединения нескольких физических сегментов (отрезков кабеля) с помощью одного или нескольких концентраторов. Каждый логический сегмент подключается к отдельному порту моста/коммутатора (рис. 1.10). При поступлении кадра на какой-либо из портов мост/коммутатор повторяет этот кадр, но не на всех портах, как это делает концентратор, а только на том порту, к которому подключен сегмент, содержащий компьютер-адресат.



Разница между мостом и коммутатором состоит в том, что мост в каждый момент времени может осуществлять передачу кадров только между одной парой портов, а коммутатор одновременно поддерживает потоки данных между всеми своими портами. Другими словами, мост передает кадры последовательно, а коммутатор параллельно.

Маршрутизаторы.

Маршрутизатор (router) имеет в своем распоряжении базу топологической информации, которая говорит ему, например, о том, между какими подсетями общей сети имеются связи и в каком состоянии (работоспособном или нет) они находятся. Имея такую карту сети, маршрутизатор может выбрать один из нескольких возможных маршрутов доставки пакета адресату.

В отличии от моста/коммутатора, который не знает, как связаны сегменты друг с другом за пределами его портов, маршрутизатор видит всю картину связей подсетей друг с другом, поэтому он может выбрать правильный маршрут и при наличии нескольких альтернативных маршрутов. Решение о выборе того или иного маршрута принимается каждым маршрутизатором, через который проходит сообщение.

Для того, чтобы составить карту связей в сети, маршрутизаторы обмениваются специальными служебными сообщениями, в которых содержится информация о тех связях между подсетями, о которых они знают (эти подсети подключены к ним непосредственно или же они узнали эту информацию от других маршрутизаторов).

Функции:

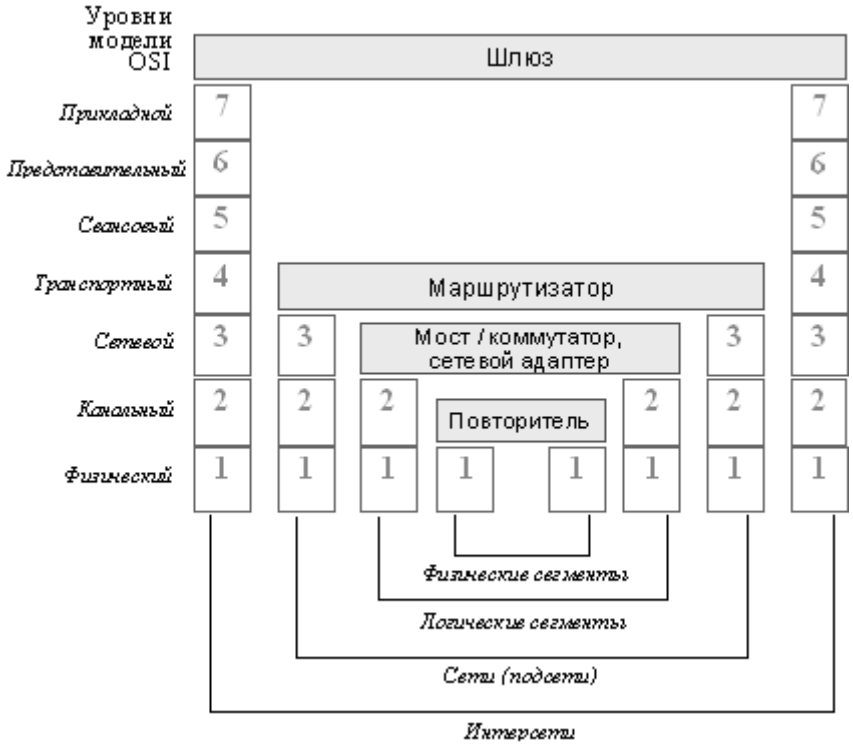
1) Маршрутизаторы позволяют объединять сети с различными принципами организации в единую сеть, которая в этом случае часто называется *интерсеть*. Маршрутизаторы могут объединять не только локальные сети с различной технологией, но и локальные сети с глобальными.
2) Маршрутизаторы надежно защищают сети друг от друга. Причем эта изоляция осуществляется гораздо проще и надежнее, чем с помощью

мостов/коммутаторов. Например, при поступлении кадра с неправильным адресом мост/коммутатор обязан повторить его на всех своих портах, что делает сеть незащищенной от некорректно работающего узла. Маршрутизатор же в таком случае просто отказывается передавать "неправильный" пакет дальше, изолируя дефектный узел от остальной сети.

3) Маршрутизатор предоставляет администратору удобные средства фильтрации потока сообщений за счет того, что сам распознает многие поля служебной информации в пакете и позволяет их именовать понятным администратору образом.

4) Маршрутизатор может обеспечивать приоритетный порядок обслуживания буферизованных пакетов, когда на основании некоторых признаков пакетам предоставляются преимущества при выборе из очереди

Шлюзы – (gateway) это устройства для соединения сетей с сильно отличающимися протоколами, например, для соединения локальных сетей с большими компьютерами или с глобальными сетями. Это самые дорогие и редко применяемые сетевые устройства. Шлюзы реализуют связь между абонентами на верхних *уровнях модели OSI* (с четвертого по седьмой). Соответственно, они должны выполнять и все функции нижестоящих *уровней*.

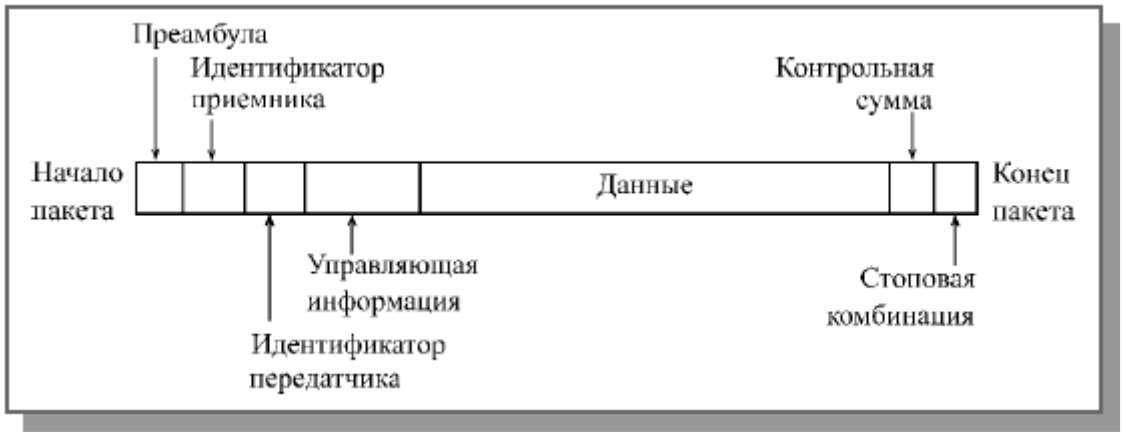


Соответствие функций коммуникационного оборудования модели OSI

Пакеты

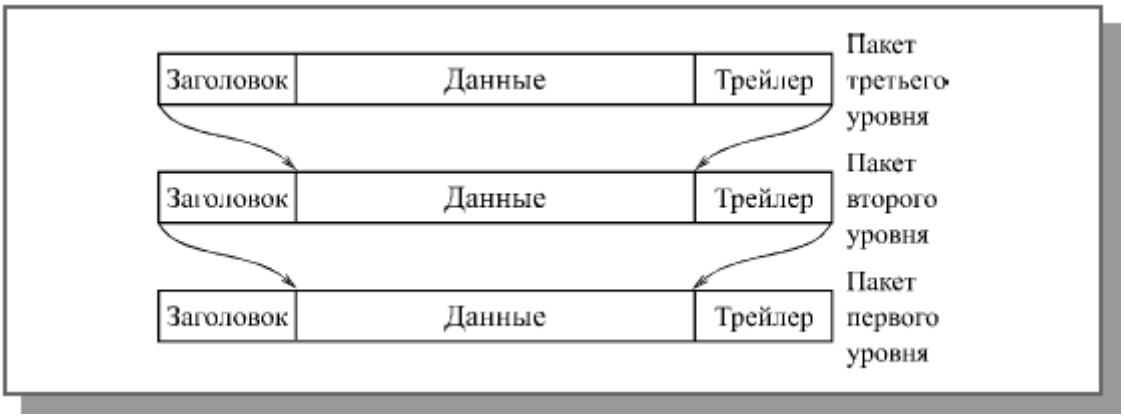
Структура и размеры *пакета* в каждой сети жестко определены стандартом на данную сеть и связаны, прежде всего, с аппаратурными особенностями данной сети, выбранной топологией и типом среды передачи информации. Кроме того, эти параметры зависят от используемого протокола (порядка обмена информацией).
Но существуют некоторые общие принципы формирования структуры *пакета*, которые учитывают характерные особенности обмена информацией по любым локальным сетям. *Пакет* содержит в себе следующие основные *поля:*

- Стартовая комбинация битов или преамбула, которая обеспечивает предварительную настройку аппаратуры адаптера или другого сетевого устройства на прием и обработку *пакета*. Это *поле* может полностью отсутствовать или же сводиться к единственному стартовому биту.
- Сетевой адрес (идентификатор) принимающего абонента, то есть индивидуальный или групповой номер, присвоенный каждому принимающему абоненту в сети. Этот адрес позволяет приемнику распознать *пакет*, адресованный ему лично, группе, в которую он входит, или всем абонентам сети одновременно (при широком вещании).
- Сетевой адрес (идентификатор) передающего абонента, то есть индивидуальный номер, присвоенный каждому передающему абоненту. Этот адрес информирует принимающего абонента, откуда пришел данный *пакет*. Включение в *пакет* адреса передатчика необходимо в том случае, когда одному приемнику могут попеременно приходить *пакеты* от разных передатчиков.
- Служебная информация, которая может указывать на тип *пакета*, его номер, размер, формат, маршрут его доставки, на то, что с ним надо делать приемнику и т.д.
- Данные (*поле* данных) – это та информация, ради передачи которой используется *пакет*. В отличие от всех остальных *полей пакета поле* данных имеет переменную длину, которая, собственно, и определяет полную длину *пакета*. Существуют специальные управляющие *пакеты*, которые не имеют поля данных. Их можно рассматривать как сетевые команды. *Пакеты*, включающие *поле* данных, называются информационными *пакетами*. Управляющие *пакеты* могут выполнять функцию начала и конца сеанса связи, подтверждения приема информационного *пакета*, запроса информационного *пакета* и т.д.
- Контрольная сумма *пакета* – это числовой код, формируемый передатчиком по определенным правилам и содержащий в свернутом виде информацию обо всем *пакете*. Приемник, повторяя вычисления, сделанные передатчиком, с принятым *пакетом*, сравнивает их результат с контрольной суммой и делает вывод о правильности или ошибочности передачи *пакета*. Если *пакет* ошибочен, то приемник запрашивает его повторную передачу. Обычно используется циклическая контрольная сумма (CRC). Подробнее об этом рассказано в главе 7.
- Стоповая комбинация служит для информирования аппаратуры принимающего абонента об окончании *пакета*, обеспечивает выход аппаратуры приемника из состояния приема. Это *поле* может отсутствовать, если используется самосинхронизирующийся код, позволяющий определять момент окончания передачи *пакета*.



В процессе сеанса обмена информацией по сети между передающим и принимающим абонентами происходит обмен информационными и управляющими *пакетами* по установленным правилам, называемым протоколом обмена. Это позволяет обеспечить надежную передачу информации при любой интенсивности обмена по сети.

При реальном обмене по сети применяются многоуровневые протоколы, каждый из уровней которых предполагает свою структуру *пакета* (адресацию, управляющую информацию, формат данных и т.д.). Ведь протоколы высоких уровней имеют дело с такими понятиями, как файл-сервер или приложение, запрашивающее данные у другого приложения, и вполне могут не иметь представления ни о типе аппаратуры сети, ни о методе управления обменом. Все *пакеты* более высоких уровней последовательно вкладываются в передаваемый *пакет*, точнее, в *поле* данных передаваемого *пакета*. Этот процесс последовательной упаковки данных для передачи называется также ***инкапсуляцией пакетов***.



Обратный процесс последовательной распаковки данных приемником.

Адресация пакетов

Каждый абонент (узел) локальной сети должен иметь свой уникальный адрес (идентификатор или MAC-адрес), для того чтобы ему можно было адресовать *пакеты*. Существуют две основные системы присвоения адресов абонентам сети (точнее, сетевым адаптерам этих абонентов).

Первая система сводится к тому, что при установке сети каждому абоненту пользователь присваивает индивидуальный адрес по порядку, к примеру, от 0 до 30 или от 0 до 254. Присваивание адресов производится программно или с помощью переключателей на плате адаптера. При этом требуемое количество разрядов адреса определяется из неравенства:

$$2^n > N_{max}$$

где n – количество разрядов адреса, а N_{max} – максимально возможное количество абонентов в сети. Например, восемь разрядов адреса достаточно для сети из 255 абонентов. Один адрес (обычно `1111....11`) отводится для широковещательной передачи, то есть он используется для *пакетов*, адресованных всем абонентам одновременно.

Именно такой подход применен в известной сети Arcnet. *Достоинства* данного подхода – малый объем служебной информации в *пакете*, а также простота аппаратуры адаптера, распознающей адрес *пакета*. *Недостаток* – трудоемкость задания адресов и возможность ошибки (например, двум абонентам сети может быть присвоен один и тот же адрес). Контроль уникальности сетевых адресов всех абонентов возлагается на администратора сети.

Второй подход к адресации был разработан международной организацией IEEE, занимающейся стандартизацией сетей. Именно он используется в большинстве сетей и рекомендован для новых разработок. Идея этого подхода состоит в том, чтобы присваивать уникальный сетевой адрес каждому адаптеру сети еще на этапе его изготовления. Если количество возможных адресов будет достаточно большим, то можно быть уверенным, что в любой сети по всему миру никогда не будет абонентов с одинаковыми адресами. Поэтому был выбран 48-битный формат адреса, что соответствует примерно 280 триллионам различных адресов. Понятно, что столько сетевых адаптеров никогда не будет выпущено.

С тем чтобы распределить возможные диапазоны адресов между многочисленными изготовителями сетевых адаптеров, была предложена следующая структура адреса:

- Младшие 24 разряда кода адреса называются OUA (Organizationally Unique Address) – организационно уникальный адрес. Именно их присваивает каждый из зарегистрированных производителей сетевых адаптеров. Всего возможно свыше 16 миллионов комбинаций, то есть каждый изготовитель может выпустить 16 миллионов сетевых адаптеров.
- Следующие 22 разряда кода называются OUI (Organizationally Unique Identifier) – организационно уникальный идентификатор. IEEE присваивает один или несколько OUI каждому производителю сетевых адаптеров. Это позволяет исключить совпадения адресов адаптеров от разных производителей. Всего возможно свыше 4 миллионов разных OUI, это означает, что теоретически может быть зарегистрировано 4 миллиона производителей. Вместе OUA и OUI называются UAA (Universally Administered Address) – универсально управляемый адрес или IEEE-адрес.
- Два старших разряда адреса управляющие, они определяют тип адреса, способ интерпретации остальных 46 разрядов. Старший бит I/G (Individual/Group) указывает на тип адреса. Если он установлен в 0, то индивидуальный, если в 1, то групповой (многопунктовый или функциональный). *Пакеты* с групповым адресом получат все имеющие этот групповой адрес сетевые адаптеры. Причем групповой адрес определяется 46 младшими разрядами. Второй управляющий бит U/L (Universal/Local) называется флажком универсального/местного управления и определяет, как был присвоен адрес данному сетевому адаптеру. Обычно он установлен в 0. Установка бита U/L в 1 означает, что адрес задан не производителем сетевого адаптера, а организацией, использующей данную сеть. Это случается довольно редко.

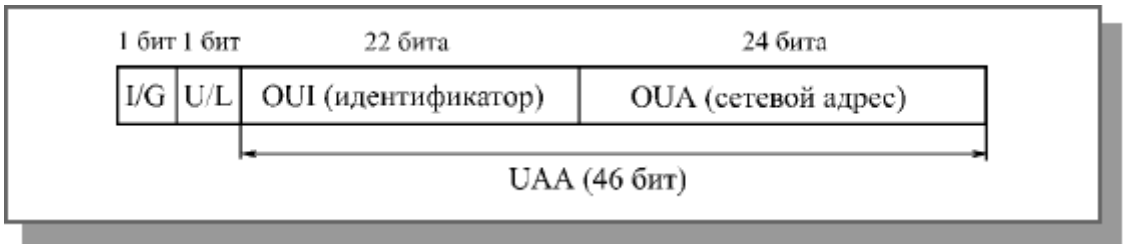


Рис. 4.7. Структура 48-битного стандартного MAC-адреса

Для широковещательной передачи (то есть передачи всем абонентам сети одновременно) применяется специально выделенный сетевой адрес, все 48 битов которого установлены в единицу. Его принимают все абоненты сети независимо от их индивидуальных и групповых адресов.

Данной системы адресов придерживаются такие популярные сети, как Ethernet, Fast Ethernet, Token-Ring, FDDI, 100VG-AnyLAN. Ее недостатки – высокая сложность аппаратуры сетевых адаптеров, а также большая доля служебной информации в

передаваемом *пакете* (адреса источника и приемника вместе требуют уже 96 битов *пакета* или 12 байт).

Во многих сетевых адаптерах предусмотрен так называемый циркулярный режим. В этом режиме адаптер принимает все *пакеты*, приходящие к нему, независимо от значения *поля* адреса приемника. Такой режим используется, например, для проведения диагностики сети, измерения ее производительности, контроля ошибок передачи. При этом один компьютер принимает и контролирует все *пакеты*, проходящие по сети, но сам ничего не передает. В данном режиме работают сетевые адаптеры мостов и коммутаторы, которые должны обрабатывать перед ретрансляцией все *пакеты*, приходящие к ним.

Топология локальных сетей

Топология сети указывает не только на физическое расположение компьютеров, как часто считают, но, что гораздо важнее, на:

- характер связей между ними,
- особенности распространения информации и сигналов по сети.

Именно характер связей определяет степень *отказоустойчивости* сети, требуемую сложность сетевой аппаратуры, наиболее подходящий метод управления *обменом*, возможные типы сред передачи (каналов связи), допустимый размер сети (длина *линий связи* и количество *абонентов*) необходимость электрического согласования и многое другое.

***Топология* шина** (или, как ее еще называют, общая шина сети Ethernet) самой своей структурой предполагает идентичность сетевого оборудования компьютеров, а также равноправие всех *абонентов* по доступу к сети. Компьютеры в шине могут передавать только по очереди, так как *линия связи* в данном случае единственная. Если несколько компьютеров будут передавать информацию одновременно, она исказится в результате наложения (**конфликта**, **коллизии**). В шине всегда реализуется режим так называемого **полудуплексного** (half duplex) *обмена* (в обоих направлениях, но по очереди, а не одновременно).

Поскольку центральный *абонент* отсутствует, разрешение возможных конфликтов в данном случае ложится на сетевое оборудование каждого отдельного *абонента*. В связи с этим сетевая аппаратура при *топологии* шина сложнее, чем при других *топологиях*. Тем не менее из-за широкого распространения сетей с *топологией* шина (прежде всего наиболее популярной сети Ethernet) стоимость сетевого оборудования не слишком высока.

Обрыв кабеля в сети с топологией шина

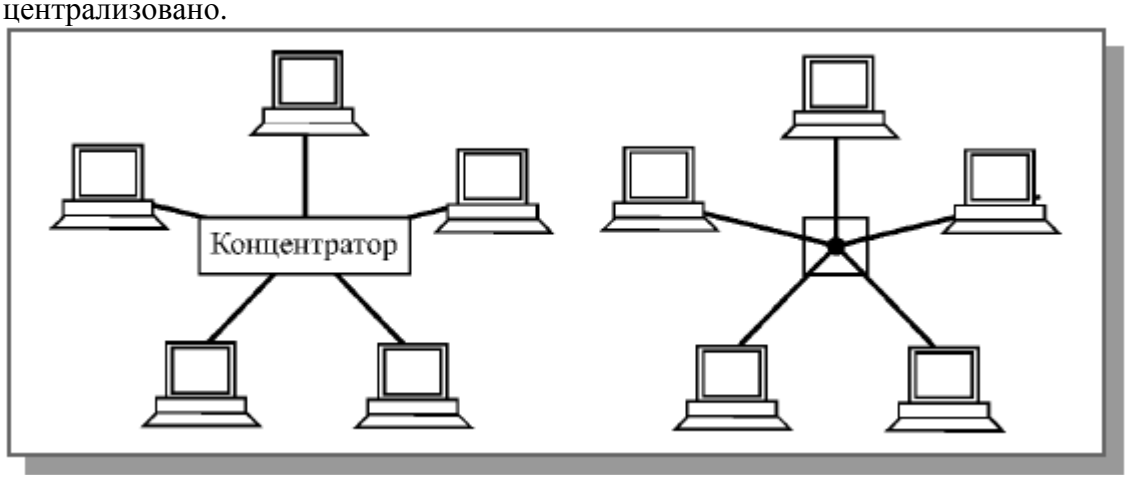
Преимущества:

1) В *топологии* шина отсутствует явно выраженный центральный *абонент*, через который передается вся информация, это увеличивает ее надежность (ведь при отказе центра перестает функционировать вся управляемая им система).
2) Добавление новых *абонентов* в шину довольно просто и обычно возможно даже во время работы сети.
3) В большинстве случаев при использовании шины требуется минимальное количество соединительного кабеля по сравнению с другими *топологиями*.
4) Стоимость сетевого оборудования не слишком высока.

Сложности:

1) Поскольку центральный *абонент* отсутствует, разрешение возможных конфликтов в данном случае ложится на сетевое оборудование каждого отдельного *абонента*. В связи с этим сетевая аппаратура при *топологии* шина сложнее, чем при других *топологиях*.
2) . В случае разрыва или повреждения кабеля нарушается согласование *линии связи*, и прекращается *обмен* даже между теми компьютерами, которые остались соединенными между собой.
3) При прохождении по *линии связи* сети с *топологией* шина информационные сигналы ослабляются и никак не восстанавливаются, что накладывает жесткие ограничения на суммарную длину *линий связи*. Причем каждый *абонент* может получать из сети сигналы разного уровня в зависимости от расстояния до передающего *абонента*. Это предъявляет дополнительные требования к приемным узлам сетевого оборудования.

Топология Звезда — это единственная *топология* сети с явно выделенным центром, к которому подключаются все остальные *абоненты*. *Обмен информацией* идет исключительно через центральный компьютер (Активная звезда), на который ложится большая нагрузка, поэтому ничем другим, кроме сети, он, как правило, заниматься не может. Понятно, что сетевое оборудование центрального *абонента* должно быть существенно более сложным, чем оборудование периферийных *абонентов*. О равноправии всех *абонентов* (как в шине) в данном случае говорить не приходится. Обычно центральный компьютер самый мощный, именно на него возлагаются все функции по управлению обменом. Никакие конфликты в сети с *топологией* звезда в принципе невозможны, так как управление полностью централизовано.



В центре сети с данной *топологией* помещается не компьютер, а специальное устройство — концентратор или, как его еще называют, хаб (hub), которое выполняет ту же функцию, что и репитер, то есть восстанавливает приходящие сигналы и пересылает их во все другие *линии связи* Пассивная звезда.

В отличие от шины, в звезде на каждой *линии связи* находятся только два *абонента*: центральный и один из периферийных. Чаще всего для их соединения используется две *линии связи*, каждая из которых передает информацию в одном направлении, то есть на каждой *линии связи* имеется только один приемник и один передатчик. Это так называемая передача **точка-точка**. Все это существенно упрощает сетевое оборудование по сравнению с шиной и избавляет от необходимости применения дополнительных, внешних терминаторов.

Преимущества:

1) Обрыв кабеля или короткое замыкание в нем при *топологии* звезда нарушает *обмен* только с одним компьютером, а все остальные компьютеры могут нормально продолжать работу.

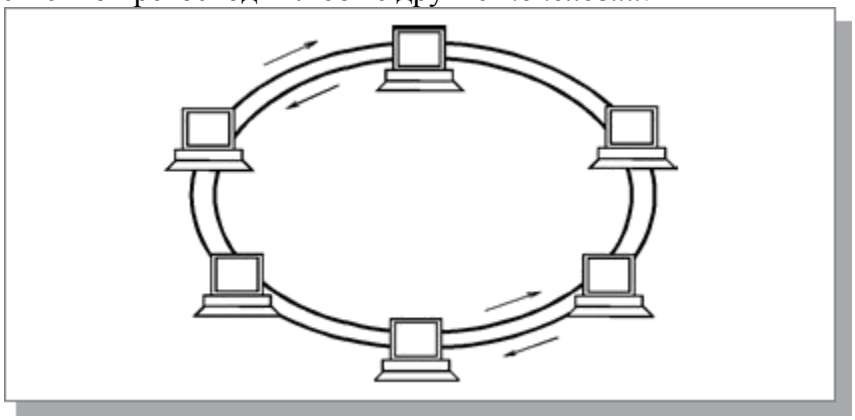
2) В отличие от шины, в звезде на каждой *линии связи* находятся только два *абонента*: центральный и один из периферийных. Чаще всего для их соединения используется две *линии связи*, каждая из которых передает информацию в одном направлении, то есть на каждой *линии связи* имеется только один приемник и один передатчик. Это так называемая передача **точка-точка**. Все это существенно упрощает сетевое оборудование по сравнению с шиной и избавляет от необходимости применения дополнительных, внешних терминаторов.
3) Проблема затухания сигналов в *линии связи* также решается в звезде проще, чем в случае шины, ведь каждый приемник всегда получает сигнал одного уровня. Предельная длина сети с *топологией* звезда может быть вдвое больше, чем в шине (то есть 2 $L_{пр}$), так как каждый из кабелей, соединяющий центр с периферийным *абонентом*, может иметь длину $L_{пр}$

Недостатки:

1) Серьезный недостаток *топологии* звезда состоит в жестком ограничении количества *абонентов*. Обычно центральный *абонент* может обслуживать не более 8—16 периферийных *абонентов*. В этих пределах подключение новых *абонентов* довольно просто, но за ними оно просто невозможно. В звезде допустимо подключение вместо периферийного еще одного центрального *абонента* (в результате получается *топология* из нескольких соединенных между собой звезд).
2) Любой отказ центрального компьютера делает сеть полностью неработоспособной.
3) Общим недостатком для всех *топологий* типа звезда (как активной, так и пассивной) является значительно больший, чем при других *топологиях*, расход кабеля

Топология Кольцо — это *топология*, в которой каждый компьютер соединен *линиями связи* с двумя другими: от одного он получает информацию, а другому передает. На каждой *линии связи*, как и в случае звезды, работает только один передатчик и один приемник (связь типа точка-точка). Это позволяет отказаться от применения внешних терминаторов.

Важная особенность кольца состоит в том, что каждый компьютер ретранслирует (восстанавливает, усиливает) приходящий к нему сигнал, то есть выступает в роли репитера. Затухание сигнала во всем кольце не имеет никакого значения, важно только затухание между соседними компьютерами кольца. Если предельная длина кабеля, ограниченная затуханием, составляет $L_{пр}$, то суммарная длина кольца может достигать $NL_{пр}$, где N — количество компьютеров в кольце. Полный размер сети в пределе будет $NL_{пр}/2$, так как кольцо придется сложить вдвое. На практике размеры кольцевых сетей достигают десятков километров (например, в сети FDDI). Кольцо в этом отношении существенно превосходит любые другие *топологии*.



Преимущества:

1) Кольцевая *топология* обычно обладает высокой устойчивостью к перегрузкам, обеспечивает уверенную работу с большими потоками передаваемой по сети информации, так как в ней, как правило, нет конфликтов (в отличие от шины), а также отсутствует центральный *абонент* (в отличие от звезды), который может быть перегружен большими потоками информации.
2) Подключение новых *абонентов* в кольцо выполняется достаточно просто, хотя и требует обязательной остановки работы всей сети на время подключения.

Недостатки:

1) Сигнал в кольце проходит последовательно через все компьютеры сети, поэтому выход из строя хотя бы одного из них (или же его сетевого оборудования) нарушает работу сети в целом. Это существенный недостаток кольца.
2) Точно так же обрыв или короткое замыкание в любом из кабелей кольца делает работу всей сети невозможной. Из трех рассмотренных *топологий* кольцо наиболее уязвимо к повреждениям кабеля, поэтому в случае *топологии* кольца обычно предусматривают прокладку двух (или более) параллельных *линий связи*, одна из которых находится в резерве.

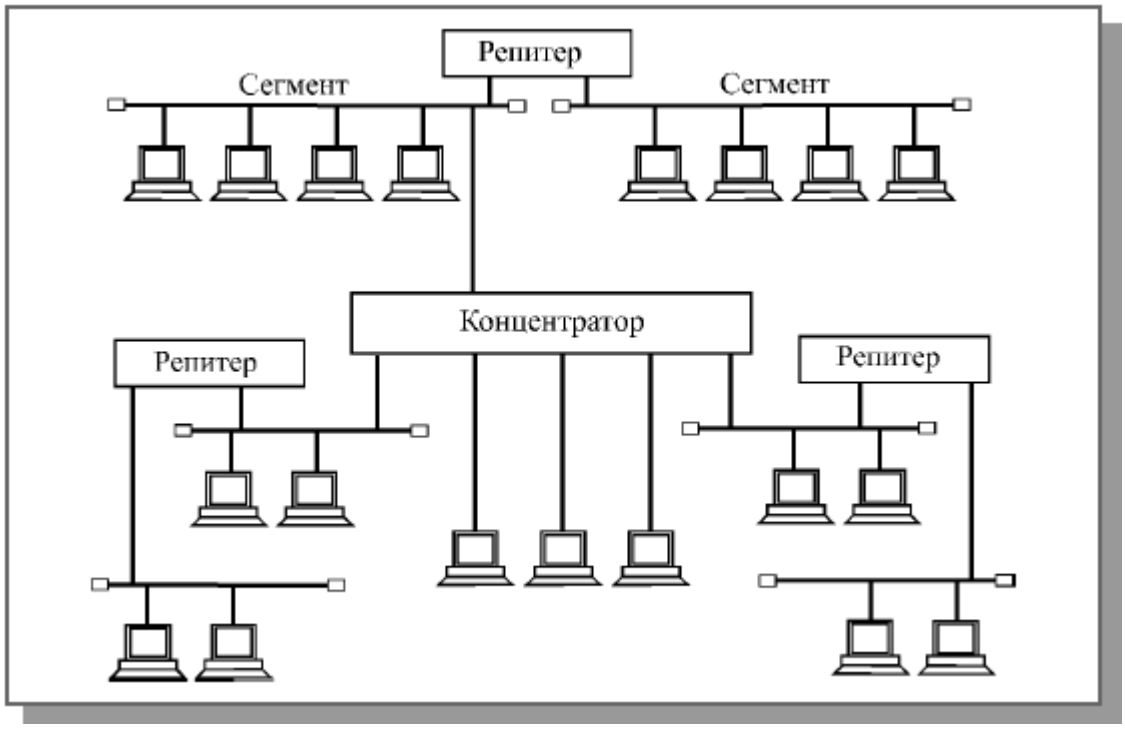
Методы управления обменом

Ethernet

Наиболее распространенный стандартный метод управления обменом CSMA/CD (Carrier Sense Multiple Access with Collision Detection – множественный доступ с контролем несущей и обнаружением коллизий). Стандарт IEEE 802.3-коаксиал. *Ethernet*, работающий на скорости 100 Мбит/с (так называемый *Fast Ethernet*, стандарт IEEE 802.3u), использующую в качестве *среды передачи* витую пару или оптоволоконный кабель. В 1997 году появилась и версия на скорость 1000 Мбит/с (Gigabit Ethernet, стандарт IEEE 802.3z).

Суть метода состоит в том, что абонент начинает передавать сразу, как только он выяснит, что сеть свободна. Если возникают коллизии, то они обнаруживаются всеми передающими абонентами. После чего все абоненты прекращают свою передачу и возобновляют попытку начать новую передачу *пакета* через временной интервал, длительность которого выбирается случайным образом. Поэтому повторные коллизии маловероятны.

Помимо стандартной топологии шина все шире применяются топологии типа пассивная звезда и пассивное дерево. При этом предполагается использование репитеров и репитерных концентраторов, соединяющих между собой различные части (сегменты) сети. В результате может сформироваться древовидная структура на сегментах разных типов.



Классическая топология сети Ethernet

Максимальная длина кабеля сети в целом (максимальный путь сигнала) теоретически может достигать 6,5 километров, но практически не превышает 3,5 километров.

В сети *Fast Ethernet* не предусмотрена физическая топология шина, используется только пассивная звезда или пассивное дерево. К тому же в *Fast Ethernet* гораздо более жесткие требования к предельной длине сети. Ведь при увеличении в 10 раз скорости передачи и сохранении *формата пакета* его минимальная длина становится в десять раз короче. Таким образом, в 10 раз уменьшается допустимая величина двойного времени прохождения сигнала по сети (5,12 мкс против 51,2 мкс в *Ethernet*).

Маркерные методы управления.

Самые популярные методы управления в кольцевых сетях ***маркерные (эстафетные)***, те, которые используют маркер (эстафету) – небольшой управляющий *пакет* специального вида. Именно эстафетная передача маркера по кольцу позволяет передавать право на захват сети от одного абонента к другому. *Маркерные методы* относятся к децентрализованным и детерминированным методам управления обменом в сети. В них нет явно выраженного центра, но существует четкая система приоритетов, и потому не бывает конфликтов.

По кольцу непрерывно ходит специальный управляющий *пакет* минимальной длины, маркер, предоставляющий абонентам право передавать свой *пакет*. Алгоритм действий абонентов:

1. Абонент 1, желающий передать свой *пакет*, должен дождаться прихода к нему свободного маркера. Затем он присоединяет к маркеру свой *пакет*, помечает маркер как занятый и отправляет эту посылку следующему по кольцу абоненту.
2. Все остальные абоненты (2, 3, 4), получив маркер с присоединенным *пакетом*, проверяют, им ли адресован *пакет*. Если *пакет* адресован не им, то они передают полученную посылку (маркер + *пакет*) дальше по кольцу.
3. Если какой-то абонент (в данном случае это абонент 3) распознает *пакет* как адресованный ему, то он его принимает, устанавливает в маркере бит подтверждения приема и передает посылку (маркер + *пакет*) дальше по кольцу.
4. Передававший абонент 1 получает свою посылку, прошедшую по всему кольцу, обратно, помечает маркер как свободный, удаляет из сети свой *пакет* и посылает

свободный маркер дальше по кольцу. Абонент, желающий передавать, ждет этого маркера, и все повторяется снова.

Основное преимущество маркерного метода перед CSMA/CD состоит в гарантированной величине времени доступа. Его максимальная величина, как и при централизованном методе, составит (N-1)• tпк, где N – полное число абонентов в сети, tпк – время прохождения пакета по кольцу. Вообще, маркерный метод управления обменом при большой интенсивности обмена в сети (загруженность более 30—40%) гораздо эффективнее случайных методов. Он позволяет сети работать с большей нагрузкой, которая теоретически может даже приближаться к 100%.

Метод *маркерного* доступа используется не только в кольце (например, в сети IBM Token Ring или FDDI), но и в шине (в частности, сеть Arcnet-BUS), а также в пассивной звезде (к примеру, сеть Arcnet-STAR). В этих случаях реализуется не физическое, а логическое кольцо, то есть все абоненты последовательно передают друг другу маркер, и эта цепочка передачи маркеров замкнута в кольцо.

Локальные и глобальные сети

Сформулировать отличительные признаки локальной сети можно следующим образом:

- Высокая скорость передачи информации, большая пропускная способность сети. Приемлемая скорость сейчас — не менее 10 Мбит/с.
- Низкий уровень ошибок передачи (или, что то же самое, высококачественные каналы связи). Допустимая вероятность ошибок передачи данных должна быть порядка 10-8 — 10-12.
- Эффективный, быстродействующий механизм управления обменом по сети.
- Заранее четко ограниченное количество компьютеров, подключаемых к сети.

При таком определении понятно, что глобальные сети отличаются от локальных прежде всего тем, что они рассчитаны на неограниченное число абонентов. Кроме того, они используют (или могут использовать) не слишком качественные каналы связи и сравнительно низкую скорость передачи. А механизм управления обменом в них не может быть гарантированно быстрым. В глобальных сетях гораздо важнее не качество связи, а сам факт ее существования.

Нередко выделяют еще один класс компьютерных сетей — городские, региональные сети (MAN, Metropolitan Area Network), которые обычно по своим характеристикам ближе к глобальным сетям, хотя иногда все-таки имеют некоторые черты локальных сетей, например, высококачественные каналы связи и сравнительно высокие скорости передачи. В принципе городская сеть может быть *локальной* со всеми ее преимуществами.

Сетевой уровень. Организация глобальной сети

Единица обмена протоколов сетевого уровня – дейтаграмма.

Функции сетевого уровня:

1) Определение пути, который должны пройти дейтаграммы от отправителя к получателю. Этот путь определяют маршрутизаторы, используя *алгоритмы маршрутизации.*
2) Продвижение данных. Маршрутизатор получает на входной интерфейс пакеты и передает их на соответствующий выходной интерфейс.
3) Установка соединения. Некоторые протоколы требуют обмена управляющими пакетами прежде чем передавать данные.

Маршрутизация – глобальный охватывающий всю сеть процесс определения пути. *Продвижение* – локальное действие конкретного маршрутизатора.

Модель сетевого обслуживания определяет характеристики сквозной передачи данных от отправителя к получателю.

Модель Виртуального канала определяет три фазы:

1) Установка виртуального канала. Службы сетевого уровня определяют путь от отправителя до получателя то есть последовательность линий связи и пакетных коммутаторов, через которые будут проходить все пакеты. Это предполагает установления определенных отметок в таблицах коммутации.
2) Передача данных. Перемещение пакетов по виртуальному каналу.
3) Разрыв виртуального канала. Отметки в таблицах коммутации уничтожаются.

Пакетные коммутаторы на всем пути знают о прохождении через них виртуальных каналов.

Дейтаграммная модель сетевого уровня работает следующим образом. Маршрутизатор ищет выходной интерфейс по таблице продвижения данных, используя адрес получателя. В этом случае пакеты могут прибыть на конечный пункт по разным маршрутам и в различном порядке.

Алгоритмы маршрутизации: глобальные и децентрализованные.

Глобальные – находит путь с минимальной стоимостью от отправителя до получателя, основываясь на полной информации о сети.

Децентрализованные – изначально каждому узлу известна только стоимость присоединенных линий связи. Затем, путем итерационных вычислений собирается информация о путях до получателей или групп получателей.

Алгоритмы маршрутизации: статические и динамические. В статических информация о маршрутах изменяется редко, часто с участием человека. В динамических запуск сбора информации осуществляется периодически.

Иерархическая маршрутизация

Сеть как совокупность соединенных друг с другом маршрутизаторов существовать не может по следующим причинам:

1) Масштабирование. Когда число маршрутизаторов очень большое, то накладные расходы на хранение данных о маршрутах, вычисление маршрутов, обмен информацией о маршрутах непомерно возрастают.
2) Административная автономия. Организация должна управлять сетью, как ей заблагорассудиться.

Выделяются **автономные системы**. В пределах АС используется один и тот же алгоритм маршрутизации и маршрутизаторы внутри АС имеют полную информацию о всех маршрутах АС.

АС соединяются с внешним миром с помощью специально выделенных маршрутизаторов – шлюзов. Таким образом, чтобы попасть из одной АС в другую необходимо пройти ряд шлюзов. Это называется **внешней маршрутизацией**.

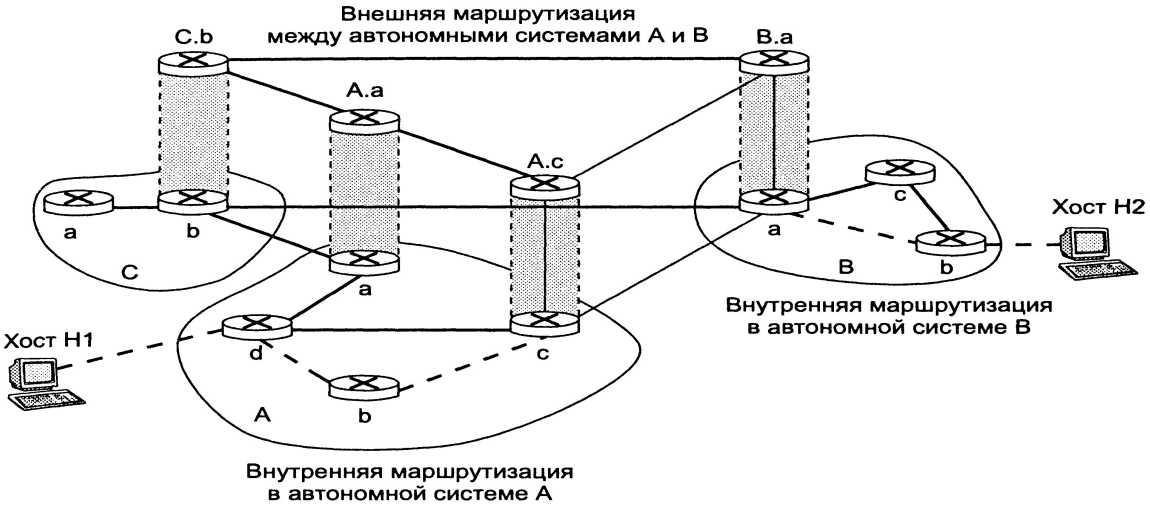


Рис. 4.12. Маршрут от A.d до B.b: внутренняя и внешняя маршрутизация

Передача от хоста H1 к хосту H2 в другой АС. Маршрутизатор A.d видит, что нет маршрута в А сети и передает на внешний маршрутизатор A.c, который по таблице внешней маршрутизации определяет, что нужно передать на внешний маршрутизатор B.a. Этот маршрутизатор видит, что пакет предназначен для его сети и по таблице внутренней маршрутизации передает на маршрутизатор B.b, к которому подключен хост H2.

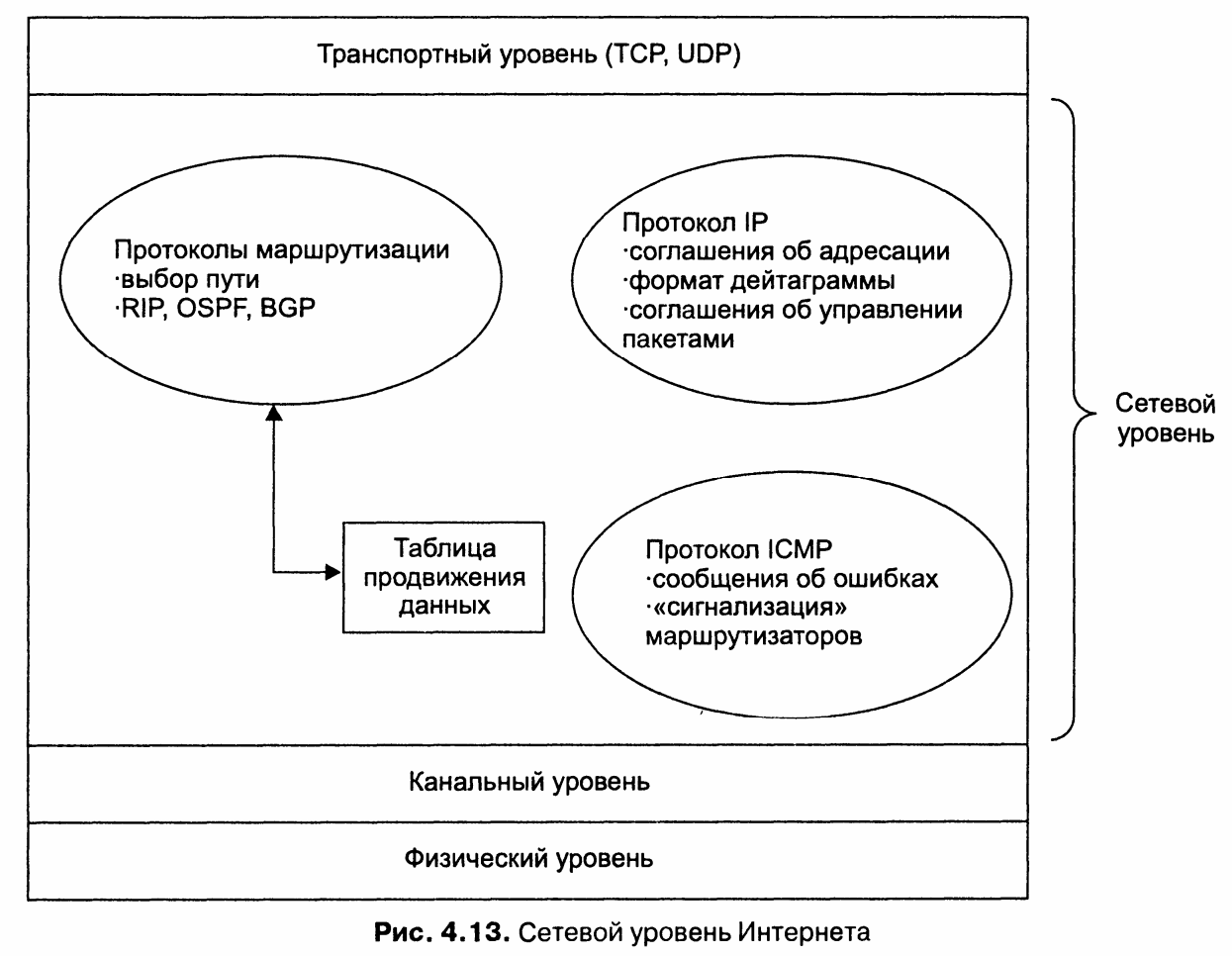


Рис. 4.13. Сетевой уровень Интернета

В качестве основного протокола сетевого уровня (в терминах модели OSI) в стеке используется протокол **IP**, который изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, объединенных как локальными, так и глобальными связями. Поэтому протокол IP хорошо работает в сетях со сложной топологией, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Протокол IP является дейтаграммным протоколом, то есть он не гарантирует доставку пакетов до узла назначения, но старается это сделать.

К уровню межсетевого взаимодействия относятся и все протоколы, связанные с составлением и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации **RIP** (Routing Internet Protocol) и **OSPF** (Open Shortest Path First), а также протокол межсетевых управляющих сообщений **ICMP** (Internet Control Message Protocol). Последний протокол предназначен для обмена информацией об ошибках между маршрутизаторами сети и узлом - источником пакета. С помощью специальных пакетов ICMP сообщается о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, об аномальных величинах параметров, об изменении маршрута пересылки и типа обслуживания, о состоянии системы и т.п.

IP адресация

IP-адрес связывается с сетевым интерфейсом, а не с компьютером. Представляется как 32-разрядное число. Принято записывать в виде четырех десятичных чисел, разделенных точкой.

11000001 00100000 11011000 00001001 193.32.216.9

Все интерфейсы должны иметь уникальные адреса. Адрес делится на две части: левая часть – адрес сети и адрес интерфейса в сети, который часто называют адресом хоста. В начальном варианте стандарта выделили 4 формата адресов.

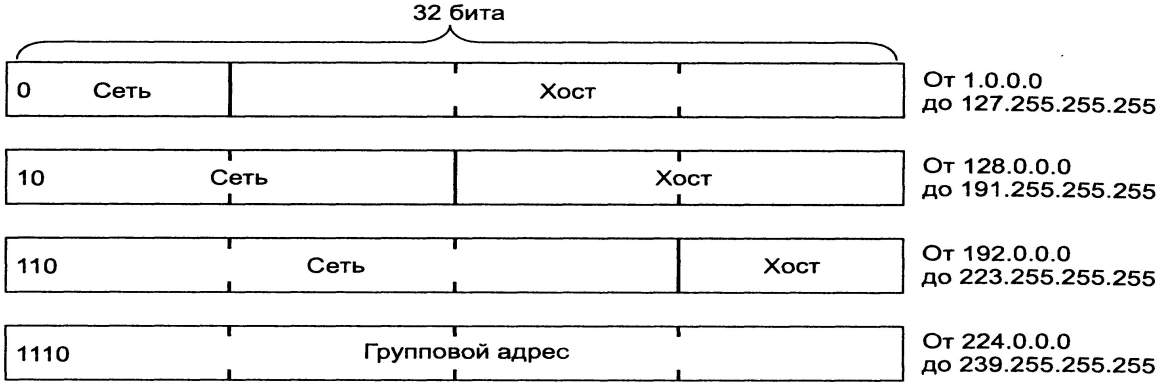


Рис. 4.17. Форматы адресов протокола IPv4

В 1994 года был принят стандарт, который определил подвижную границу между адресом сети и хоста. Вместе с адресом задается маска сети – левая часть содержит 1, правая 0. Разряды с 1 определяют адрес сети. Таким образом, можно формировать подсети.

Соглашения о специальных адресах: broadcast, multicast, loopback

В протоколе IP существует несколько соглашений об особой интерпретации IP-адресов:

- если IP-адрес состоит только из двоичных нулей,

0 0 0 0 ..... 0 0 0 0

то он обозначает адрес того узла, который сгенерировал этот пакет;

- если в поле номера сети стоят 0,

0 0 0 0 0	Номер узла

то по умолчанию считается, что этот узел принадлежит той же самой сети, что и узел, который отправил пакет;

- если все двоичные разряды IP-адреса равны 1,

то пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же сети, что и источник этого пакета. Такая рассылка называется ограниченным широковещательным сообщением (limited broadcast);

- если в поле адреса назначения стоят сплошные 1,

Номер сети	1111.....11

то пакет, имеющий такой адрес рассылается всем узлам сети с заданным номером. Такая рассылка называется широковещательным сообщением (broadcast);

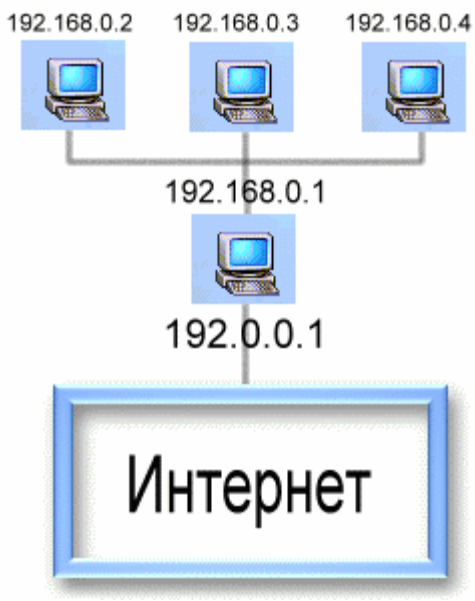
- адрес 127.0.0.1 зарезервирован для организации обратной связи при тестировании работы программного обеспечения узла без реальной отправки пакета по сети. Этот адрес имеет название loopback.

Уже упоминавшаяся форма группового IP-адреса - multicast - означает, что данный пакет должен быть доставлен сразу нескольким узлам, которые образуют группу с номером, указанным в поле адреса. Узлы сами идентифицируют себя, то есть определяют, к какой из групп они относятся. Один и тот же узел может входить в несколько групп. Такие сообщения в отличие от широковещательных называются мультивещательными. Групповой адрес не делится на поля номера сети и узла и обрабатывается маршрутизатором особым образом.

В протоколе IP нет понятия широковещательности в том смысле, в котором оно используется в протоколах канального уровня локальных сетей, когда данные должны быть доставлены абсолютно всем узлам. Как ограниченный широковещательный IP-адрес, так и широковещательный IP-адрес имеют пределы распространения в интерсети - они ограничены либо сетью, к которой принадлежит узел - источник пакета, либо сетью, номер которой указан в адресе назначения. Поэтому деление сети с помощью маршрутизаторов на части локализует широковещательный шторм пределами одной из составляющих общую сеть частей просто потому, что нет способа адресовать пакет одновременно всем узлам всех сетей составной сети.

Чтобы выделить локальную сеть в общей глобальной сети используются следующие адреса:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255



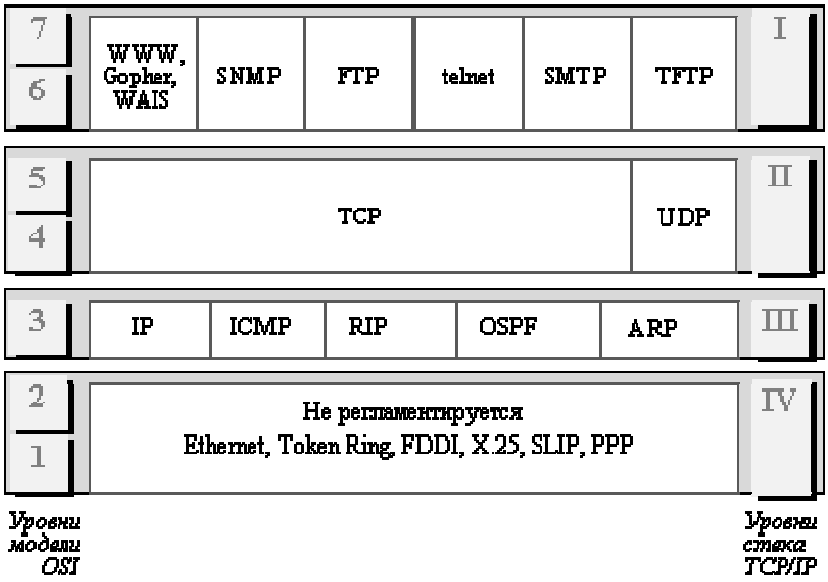
Хитрость состоит в том, что адреса, входящие в эти диапазоны, вычеркнуты из таблиц глобальной маршрутизации Интернета. Если кто-то, находящийся за пределами локальной сети, запросит или передаст информацию на адрес 192.168.0.4, то ему будет отказано: этот адрес не является глобально маршрутизируемым. Рисунок иллюстрирует связь локальной сети с глобальной.

Стек протоколов TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) - это промышленный стандарт стека протоколов, разработанный для глобальных сетей.

Следующий уровень (***уровень II***) называется основным. На этом уровне функционируют протокол управления передачей **TCP** (Transmission Control Protocol) и протокол

дейтаграмм пользователя **UDP** (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования виртуальных соединений. Протокол UDP обеспечивает передачу прикладных пакетов дейтаграммным способом, как и IP, и выполняет только функции связующего звена между сетевым протоколом и многочисленными прикладными процессами.



Верхний уровень (***уровень I***) называется прикладным. За долгие годы использования в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и сервисов прикладного уровня. К ним относятся такие широко используемые протоколы, как протокол копирования файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы доступа к удаленной информации, такие как WWW и многие другие. Остановимся несколько подробнее на некоторых из них.

Отображение физических адресов на IP-адреса: протоколы ARP и RARP

В протоколе IP-адрес узла, то есть адрес компьютера или порта маршрутизатора, назначается произвольно администратором сети и прямо не связан с его локальным адресом, как это сделано, например, в протоколе IPX. Подход, используемый в IP, удобно использовать в крупных сетях и по причине его независимости от формата локального адреса, и по причине стабильности, так как в противном случае, при смене на компьютере сетевого адаптера это изменение должны бы были учитывать все адресаты всемирной сети Internet (в том случае, конечно, если сеть подключена к Internet'у).

Локальный адрес используется в протоколе IP только в пределах локальной сети при обмене данными между маршрутизатором и узлом этой сети. Маршрутизатор, получив пакет для узла одной из сетей, непосредственно подключенных к его портам, должен для передачи пакета сформировать кадр в соответствии с требованиями принятой в этой сети технологии и указать в нем локальный адрес узла, например его MAC-адрес. В пришедшем пакете этот адрес не указан, поэтому перед маршрутизатором встает задача поиска его по известному IP-адресу, который указан в пакете в качестве адреса назначения. С аналогичной задачей сталкивается и конечный узел, когда он хочет отправить пакет в удаленную сеть через маршрутизатор, подключенный к той же локальной сети, что и данный узел.

Для определения локального адреса по IP-адресу используется протокол разрешения адреса *Address Resolution Protocol, ARP*. Протокол ARP работает различным образом в зависимости от того, какой протокол канального уровня работает в данной сети - протокол локальной сети (Ethernet, Token Ring, FDDI) с возможностью широковещательного доступа одновременно ко всем узлам сети, или же протокол глобальной сети (X.25, frame relay), как правило не поддерживающий широковещательный доступ. Существует также протокол, решающий обратную задачу - нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP - *RARP (Reverse Address Resolution Protocol)* и используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера.

В локальных сетях протокол ARP использует широковещательные кадры протокола канального уровня для поиска в сети узла с заданным IP-адресом.

Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель указывает свой локальный адрес. ARP-запросы и ответы используют один и тот же формат пакета. Так как локальные адреса могут в различных типах сетей иметь различную длину, то формат пакета протокола ARP зависит от типа сети. На рисунке 3.2 показан формат пакета протокола ARP для передачи по сети Ethernet.

0 8 16 31

Тип сети		**Тип протокола**
Длина локального адреса	Длина сетевого адреса	Операция
Локальный адрес отправителя (байты 0 - 3)		
Локальный адрес отправителя (байты 4 - 5)		IP-адрес отправителя (байты 0-1)
IP-адрес отправителя (байты 2-3)		Искомый локальный адрес (байты 0 - 1)
Искомый локальный адрес (байты 2-5)		
Искомый IP-адрес (байты 0 - 3)		

Рис. 3.2. Формат пакета протокола ARP

В поле типа сети для сетей Ethernet указывается значение 1. Поле типа протокола позволяет использовать пакеты ARP не только для протокола IP, но и для других сетевых протоколов. Для IP значение этого поля равно 0800_{16}.

Длина локального адреса для протокола Ethernet равна 6 байтам, а длина IP-адреса - 4 байтам. В поле операции для ARP запросов указывается значение 1 для протокола ARP и 2 для протокола RARP.

Узел, отправляющий ARP-запрос, заполняет в пакете все поля, кроме поля искомого локального адреса (для RARP-запроса не указывается искомый IP-адрес). Значение этого поля заполняется узлом, опознавшим свой IP-адрес.

В глобальных сетях администратору сети чаще всего приходится вручную формировать ARP-таблицы, в которых он задает, например, соответствие IP-адреса адресу узла сети X.25, который имеет смысл локального адреса. В последнее время наметилась тенденция автоматизации работы протокола ARP и в глобальных сетях. Для этой цели среди всех маршрутизаторов, подключенных к какой-либо глобальной сети, выделяется специальный маршрутизатор, который ведет ARP-таблицу для всех остальных узлов и маршрутизаторов этой сети. При таком централизованном подходе для всех узлов и маршрутизаторов вручную нужно задать только IP-адрес и локальный адрес выделенного маршрутизатора. Затем каждый узел и маршрутизатор регистрирует свои адреса в выделенном маршрутизаторе, а при необходимости установления соответствия между IP-адресом и локальным адресом узел обращается к выделенному маршрутизатору с запросом и автоматически получает ответ без участия администратора.

Отображение символьных адресов на IP-адреса: служба DNS

DNS (Domain Name System) - это распределенная база данных, поддерживающая иерархическую систему имен для идентификации узлов в сети Internet. Служба DNS предназначена для автоматического поиска IP-адреса по известному символьному имени узла. Спецификация DNS определяется стандартами RFC 1034 и 1035. DNS требует статической конфигурации своих таблиц, отображающих имена компьютеров в IP-адрес.

Протокол DNS является служебным протоколом прикладного уровня. Этот протокол несимметричен - в нем определены DNS-серверы и DNS-клиенты. DNS-серверы хранят часть распределенной базы данных о соответствии символьных имен и IP-адресов. Эта база данных распределена по административным доменам сети Internet. Клиенты сервера DNS знают IP-адрес сервера DNS своего административного домена и по протоколу IP передают запрос, в котором сообщают известное символьное имя и просят вернуть соответствующий ему IP-адрес.

Если данные о запрошенном соответствии хранятся в базе данного DNS-сервера, то он сразу посылает ответ клиенту, если же нет - то он посылает запрос DNS-серверу другого домена, который может сам обработать запрос, либо передать его другому DNS-серверу. Все DNS-серверы соединены иерархически, в соответствии с иерархией доменов сети Internet. Клиент опрашивает эти серверы имен, пока не найдет нужные отображения. Этот процесс ускоряется из-за того, что серверы имен постоянно кэшируют информацию, предоставляемую по запросам. Клиентские компьютеры могут использовать в своей работе IP-адреса нескольких DNS-серверов, для повышения надежности своей работы.

База данных DNS имеет структуру дерева, называемого доменным пространством имен, в котором каждый домен (узел дерева) имеет имя и может содержать поддомены. Имя домена идентифицирует его положение в этой базе данных по отношению к родительскому домену, причем точки в имени отделяют части, соответствующие узлам домена.

Корень базы данных DNS управляется центром Internet Network Information Center. Домены верхнего уровня назначаются для каждой страны, а также на организационной основе. Имена этих доменов должны следовать международному стандарту ISO 3166. Для обозначения стран используются трехбуквенные и двухбуквенные аббревиатуры, а для различных типов организаций используются следующие аббревиатуры:

- com - коммерческие организации (например, microsoft.com);
- edu - образовательные (например, mit.edu);

- gov - правительственные организации (например, nsf.gov);
- org - некоммерческие организации (например, fidonet.org);
- net - организации, поддерживающие сети (например, nsf.net).

Каждый домен DNS администрируется отдельной организацией, которая обычно разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям. Каждый домен имеет уникальное имя, а каждый из поддоменов имеет уникальное имя внутри своего домена. Имя домена может содержать до 63 символов. Каждый хост в сети Internet однозначно определяется своим *полным доменным именем (fully qualified domain name, FQDN)*, которое включает имена всех доменов по направлению от хоста к корню. Пример полного DNS-имени :

citint.dol.ru.