1. Операционные системы
1.1. Эволюция операционных систем

Операционная система выполняет две функции:

- Предоставляет пользователю виртуальную машину, обеспечивающую стандартизованный интерфейс к аппаратным ресурсам реального компьютера.
- Обеспечивает управление ресурсами компьютера.

Управление ресурсами предполагает распределение ресурсов между задачами и пользователями и отслеживание состояния ресурса.

Потребность в данных функции возникло не сразу, в первых поколениях ЭВМ функции операционных систем выполнял машинный язык, на котором осуществлялась решение задач. Фактически данный машинный язык и его библиотеки можно рассматривать в качестве прообраза операционных систем. Расширение парка ЭВМ потребовало выделения набора функций общих для различных пользователей, необходимости переключения различных задач без выключения ЭВМ. Выделение этих функций в отдельную задачу и привело к появлению операционных систем первого поколения.

Эволюцию операционных систем можно представить следующим образом:

Первый период (1945 -1955)

Программирование осуществлялось на машинном языке, который выполнял функции прообраза операционной системы. Все задачи организации вычислительного процесса решались вручную каждым программистом.

Второй период (1955 - 1965)

В эти годы появились первые алгоритмические языки. Появились первые системы пакетной обработки, которые просто автоматизировали запуск одной программ за другой. Был разработан язык управления пакетом заданий (колоды перфокарт).

Третий период (1965 - 1980)

Появление программно–совместимых машин. Программная совместимость требовала и совместимости (универсальности) операционных систем, работающих на различных ЭВМ. Первым универсальным операционным системам была характерна низкая эффективность. Появление таких понятий как: Мультипрограммирование, спулинг, разделения времени. ***Мультипрограммирование*** - попеременное выполнение несколько программ. ***Спулинг*** – загрузка программы с носителя (перфокарты) на диск и потом (при выполнении) в память. ***Операционные системы с разделением времени*** - эмуляция единоличного использования вычислительной машины.

Четвертый период (1980 - настоящее время)

Современное поколение операционных систем. С середины 80-х стали бурно развиваться сетевые или распределенные операционные системы, в качестве примера которых можно привести UNIX, Novel, WindowsNT.

1.2. Классификация операционных систем

Операционные системы могут различаться особенностями реализации алгоритмов управления ресурсами компьютера, особенностями методов проектирования, архитектурными особенностями аппаратных платформ, областью применения.

Поддержка многозадачности

Операционные системы делятся на: однозадачные (например, MS-DOS, MSX) и многозадачные (OS/2, UNIX, Windows 95 и выше).

Однозадачные операционные системы предоставляют ресурсы всей виртуальной машины только одному пользователю.

Многозадачные операционные системы осуществляют разделение ресурсов реального компьютера между несколькими задачами и управление разделяемыми между задачами ресурсами.

Поддержка многопользовательского режима

В зависимости от особенностей работы пользователей операционные системы делят на однопользовательские (MS-DOS, Windows 3.x, ранние версии OS/2) и многопользовательские (UNIX, Windows NT).

Однопользовательские операционные системы – все выполняемые задачи принадлежат одному логическому пользователю, даже если это на самом деле различные люди. В этом случае не возможно разделять процессы различных пользователей и обеспечить их защиту.

Многопользовательские операционные системы – разделение пользователей позволяет организовать разделение и защиту от несанкционированного доступа процессов и ресурсов различных пользователей.

Реализация многозадачности

В зависимости от реализации многозадачности разделяют не вытесняющую (NetWare, Windows 3.x) и вытесняющую многозадачность (Windows NT, OS/2, UNIX).

В операционных системах с вытесняющей многозадачностью переключение процессов осуществляется только после завершения обработки текущего процесса.

При вытесняющей многозадачности инициатором переключения процессов выступает операционная система.

Многозадачные операционные системы разделяют:

- системы пакетной обработки (операционная система ЕС ЭВМ),
- системы разделения времени (UNIX, VMS),
- системы реального времени (QNX, RT/11).

Поддержка многонитевости – распараллеливания вычислений в рамках одной задачи.

Многопроцессорная обработка (*мультипроцессирование)*

При многопроцессорной обработке все системные и пользовательские процессы или их составные части распределяются между группой процессоров. Многопроцессорная обработка позволяет существенно повысить скорость выполнения процессов. К многопроцессорным операционным системам относятся Solaris 2.x, Open Server 3.x , OS/2, Windows NT и Novell NetWare 4.1.

Поддержка различных аппаратных платформ

В настоящее время созданы операционные системы для различных аппаратных платформ, таких как: персональные компьютеры, мини-компьютеры, мейнфреймы, кластеры и сети ЭВМ. Учет специфики аппаратных средств привило к появлению различных операционных систем.

Мобильные операционные системы

В зависимости от наличия возможности переносить операционную систему с одного компьютера на другой выделяют класс мобильных операционных систем, например Novell NetWare, UNIX.

Рассмотренные выше особенности функционирования тесто связанны с особенностями реализации операционных систем. К основным концепциям построения операционных систем относятся:

Принципы построения ядра системы – монолитное ядро или микроядерный подход.

При монолитном ядре операционная система является одной программой, выполняющей большинство системных функций.

При микроядерной архитектуре микроядро выполняет только минимально–достаточный набор наиболее существенных функций операционной системы по управлению аппаратурой компьютера на самом низком уровне. Все остальные функции, условно называемые высокого уровня выполняют внешние компоненты (системные программы) операционной системы.

Микроядерная архитектура позволяет организовать более гибкую и надежную операционную систему за счет некоторого снижения быстродействия.

Объектно–ориентированная система

Технология объектно-ориентированного программирования несет огромные преимущества при разработке программного обеспечения, в том числе и системного, по сравнению с другими стилями программирования. Фундаментальные механизмы объектно–ориентированного программирования – наследование, инкапсуляция, полиморфизм позволяют организовать эффективные программные структуры с защитой данных отдельных классов от несанкционированного доступа, конструирования и уничтожение объектов.

Множественность прикладных сред

Концепция множественных прикладных сред реализуется на основе микроядра, над которым создаются виртуальные среды для работы пользовательских приложений различных операционных систем.

Распределенная организация операционных систем

При работе в компьютерной сети предоставляется возможность распределить некоторые, в первую очередь ресурсоемкие, функции операционной системы между компьютерами сети. Реализация такого разделения требует создание единой сетевой службы управления разделяемыми ресурсами.

1.3. Структура сетевой операционной системы

Сетевая операционная система – это операционная система компьютера, обеспечивающая ему возможность работать в сети.

Структура взаимодействия сетевой операционной системы с операционной системой на компьютере клиента показана на рис. 1.1.

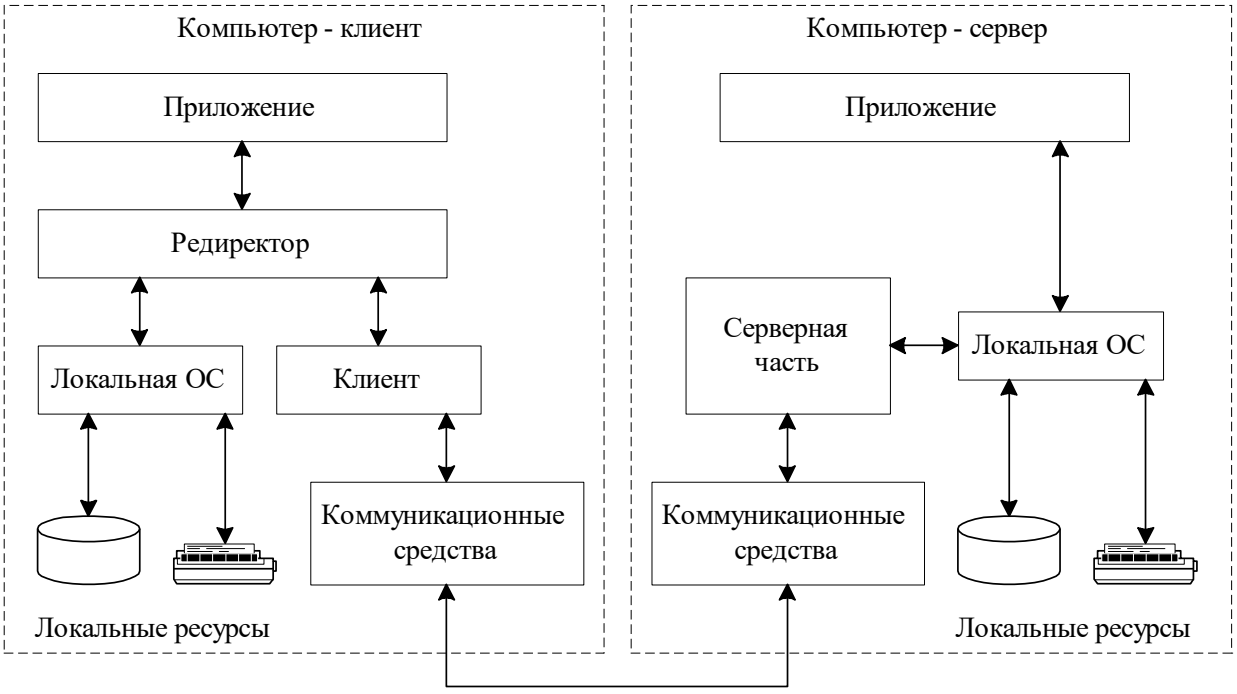


Рис. 1.1. Структура взаимодействия операционных систем клиента и сервера

Главная задача сетевой операционной системы – обеспечить взаимодействие операционных систем отдельных компьютеров, за счет объемна сообщениями в соответствии с установленным проколом.

В составе серверной и клиентской части (рис.1.1) можно выделить «Локальную ОС» задача которой – управление локальными ресурсами компьютера.

Серверная часть на компьютере–сервере обеспечивает возможность предоставление ресурсов компьютера в общий доступ остальным компьютерам сети.

На клиентском компьютере программа редиректор перехватывает запросы, поступающие от приложений. В случае если запрос адресован локальному ресурсу данного компьютера, то он передается локальной операционной системе. В том случае если адресуемое устройство является внешним, то он передается на клиентскую часть, которая преобразует запрос из локальной формы в сетевой формат и передает его транспортной подсистеме, отвечающий за доставку сообщения к компьютеру серверу через коммуникационные средства. Таким образом, программа редиректор поддерживает сетевые имена (адреса) устройств.

Получив запрос, серверная часть преобразует его для выполнения локальной операционной системы компьютера–сервера. Ответ сервера формируется аналогичным образом.

Рассмотренная архитектура взаимодействия может реализовываться в следующих вариантах, представленных на рис. 1.2.

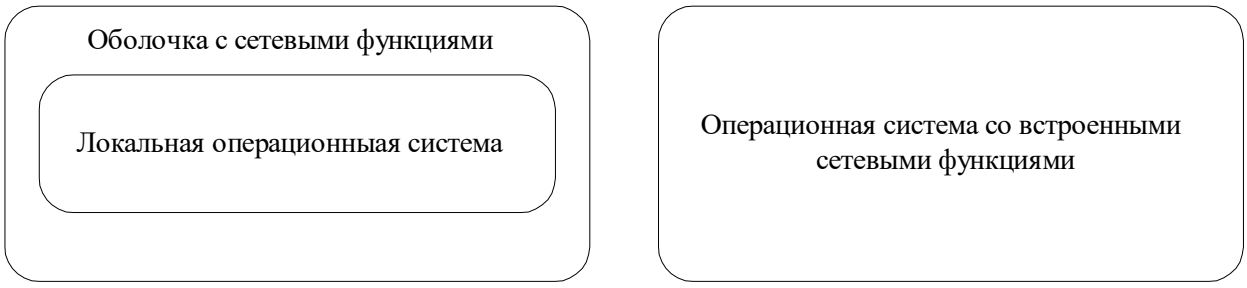


Рис. 1.2. Способы построения операционных систем

В первом случае над локальной операционной системой надстраиваются оболочка, выполняющая все сетевые функции. Примером такой организации операционных систем могут служить LANtastic работающая над OS/2, Personal Ware или Novell NetWare работающие над MS DOS. Данные операционные системы запускаются как задачи над локальной операционной системой компьютера сервера и обмениваются с ней набором установленных сигналов. Подобный обмен снижает производительность системы и допускает возможность перехвата информации, то есть становится не только узким, но и небезопасным местом в архитектуре сетевой операционной системы.

Другой способ построения сетевой операционной системы это изначальное встраивание сетевых функций в операционную системы. Например, так организованы операционные системы Windows NT сервер и выше. Такая организация позволяет разместить сетевые функции в необходимых местах операционной системы, обеспечивая быстрый и защищенный доступ к сетевым ресурсам. Такой подход имеет очевидные преимущества по скорости, но приводит к усложнению разработки и эксплуатации ядра операционных систем.

В зависимости от организации взаимодействия клиент-сервер выделяют следующие сетевые операционные системы:

Одноранговые сетевые операционные системы – все компьютеры равны в правах доступа к ресурсам друг друга (каждый компьютер имеет функции и клиента и сервера).

Сетевые операционные системы с выделенным сервером – существует выделенный компьютер (ы), предоставляющий свои ресурсы остальным компьютерам сети.

2. Управление процессами

Основная функция операционной системы – управление ресурсами компьютера. Заявка на потребление системных ресурсов называется процессом (задачей). Задача управления (планирования) процессов это распределение времени процессора между множеством процессов и организация взаимодействия процессов между собой, отслеживание ресурсов находящихся в совместном использовании.

Состояние процессов

В многозадачной системе процесс может находиться в одном из состояний:

выполнение – активное состояние процесса;

ожидание – пассивное состояние процесса;

готовность – пассивное состояние процесса (полностью готового, по ресурсам для выполнения) вызванное тем, что процессор занят обслуживанием другого процесса.

Граф состояний процесса показан на рис. 2.1.

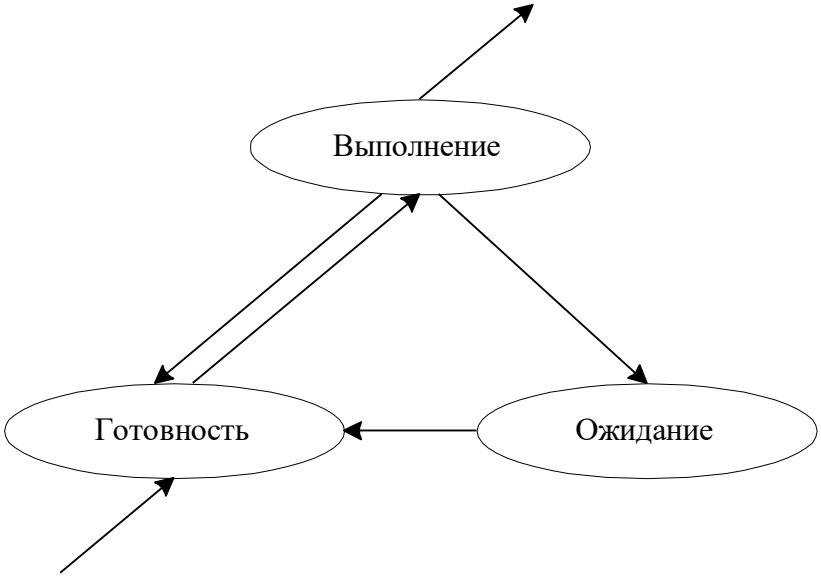


Рис. 2.1. Граф состояний процесса

Жизненный цикл процесса начинается с состояния «готовность», когда процесс готов к выполнению и ждет своей очереди. Из состояния «готовность» процесс переходит в состояние «выполнения». В однозадачной операционной системе в состоянии «выполнения» может находиться только один процесс. Он занимает процессор либо до завершения своего выполнения, либо процесс может быть вытеснен по инициативе процессора или в случае если ему не хватает ресурсов, в состояния «ожидание» или «готовность». В этих состояниях могут находиться одновременно сразу несколько процессов.

Контекст и дескриптор процесса

Контекстом процесса – состояние операционной среды отображается состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода–вывода, кодами ошибок выполняемых данным процессом системных вызовов и т.д.

Дескриптором процесса – идентификатор процесса, состояние процесса, данные о степени привилегированности процесса, место нахождения кодового сегмента и другая информация.

Дескриптор процесса по сравнению с контекстом содержит более оперативную информацию, которая должна быть легко доступна подсистеме планирования процессов. Контекст процесса содержит менее актуальную информацию и используется операционной системой только после того, как принято решение о возобновлении прерванного процесса.

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки.

Для создания процесса необходимо:

1. создать информационные структуры, описывающие данный процесс, то есть его дескриптор и контекст;
2. включить дескриптор нового процесса в очередь готовых процессов;
3. загрузить кодовый сегмент процесса в оперативную память или в область свопинга.

Алгоритмы планирования процессов

Планирование процессов включает в себя решение следующих задач:

1. определение момента времени для смены выполняемого процесса;
2. выбор процесса на выполнение из очереди готовых процессов;
3. переключение контекстов "старого" и "нового" процессов.

Первые две задачи решаются программными средствами, а последняя, в значительной степени аппаратно.

Существует множество различных алгоритмов планирования процессов, в первую очередь – алгоритмы, основанные на *квантовании,* и алгоритмы, основанные на *приоритетах*. Под *приоритетом* понимается число, которое ставится в соответствие важности данного процесса для системы.

В соответствии с алгоритмами, основанными на квантовании, смена активного процесса происходит, если:

- процесс завершился и покинул систему,
- произошла ошибка,
- процесс перешел в состояние ожидание,
- исчерпан квант процессорного времени, отведенный данному процессу.

В системах с абсолютными приоритетами, кроме того, смена процесса возможна, в случае если в очереди процессов появился процесс с большим значением приоритета, чем у выполняемого в данный момент времени.

Многие системы используют для алгоритмов планирования оба этих принципа: для работы процесса выделяется определенное количество квантов времени, а выборка следующего процесса из очереди происходит с учетом приоритетов.

Различают два основных типа алгоритмов планирования процессов – вытесняющие и не вытесняющие.

Невытесняющая многозадачность – предусматривает, что активный процесс выполняется до тех пор, пока он сам не отдаст управление планировщику операционной системы. Использование алгоритма невытесняющей многозадачности в операционной системе Novell NetWare, позволило добиться высокой скорости выполнения файловых операций, что очень важно для файл-сервера.

Вытесняющая многозадачность – предполагает, что решение о переключение выполнения процесса принимается планировщиком операционной системы. Планирование процессов на основе вытесняющей многозадачности используется в большинстве современных операционных систем, таких как UNIX, Windows NT, OS/2, VAX/VMS.

Средства синхронизации процессов

При взаимодействии процессов часто возникает проблема, когда выполнение одного из процессов должно быть приостановлено до тех пор, пока другой процесс не подготовит ему необходимые данные.

На рис. 2.2 показан граф вычислительного процесса. Выполнение процесса А, должно быть приостановлено, до тех пор, пока процесс Б не подготовит необходимые данные для выполнения блока «c».

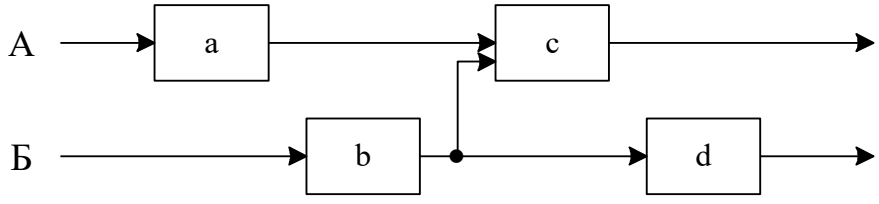


Рис. 2.2. Граф вычислительного процесса

Решение данной проблемы возможно при организации синхронизации процессов.

Критические секции

Критическая секция – часть программы, в которой осуществляется доступ к совместным ресурсам. Для нормальной работы программы необходимо обеспечить работу с указанным ресурсом во время критической сессии только одного процесса. Это может быть организованно следующим образом:

- Запрещение процессу во время нахождения в критической секции все прерывания. Данный способ может привести к сбою в работе всей системы по этому на практике, как правило, не применяется.
- Использование блокирующих переменных. Пример использования блокирующих переменных показан на блок–схеме (рис. 2.3). Данные операторы должны быть вставлены в каждый процесс, работающий с ресурсом D. Признаком что ресурс занят, является значение переменной flagD=1. Главный недостаток использования метода блокирующих переменных заключается в наличии холостого цикла ожидания освобождения ресурса. Во время данного холостого цикла процесс бесполезно тратит ресурсы процессора.
- Использование системных функций wait() и post(). Избежать бесполезных потерь времени и работе холостого цикла возможно при использовании функций работы с событиями. Функция wait(id) переводит процесс в режим ожидания до освобождения ресурса с идентификатором id. Функция post(id) сообщает операционной системе, что ресурс id освободился и можно выбрать из очереди процессов находящихся в ожидании данного ресурса следующий процесс. Блок–схема, показывающая работу с критической секцией на основе системных событий показана на рис. 2.4.

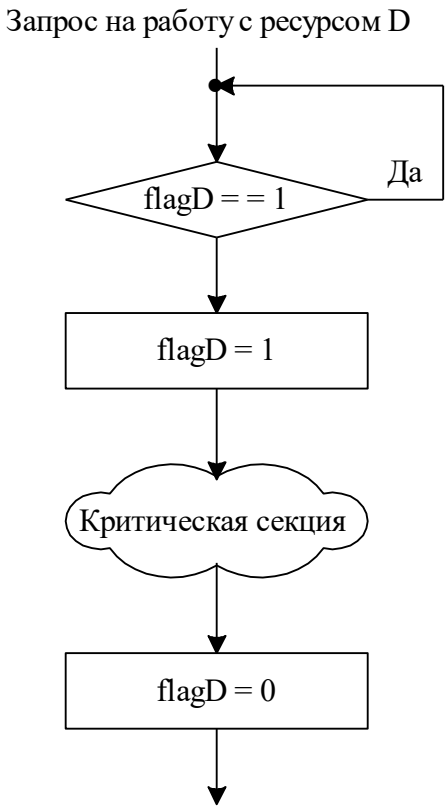


Рис. 2.3. Фрагмент блок–схемы, использующие блокирующие переменные

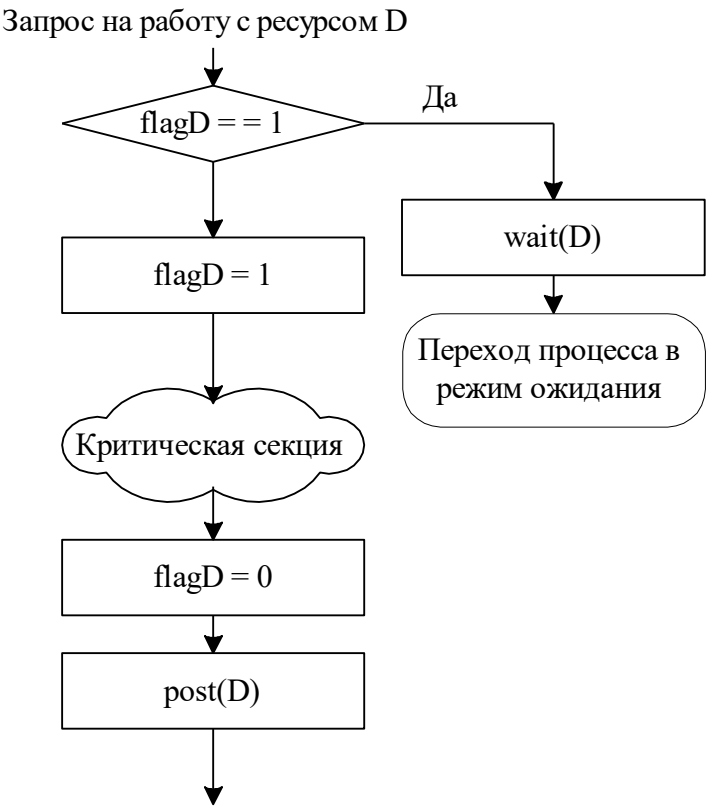


Рис. 2.4. Фрагмент блок–схемы, использующий системные события для работы с критической секцией

Тупики

Использование общих ресурсов несколькими процессами может привести к их взаимной блокировке. Например, обмен данными через общий файл, представленный на рис. 2.5.

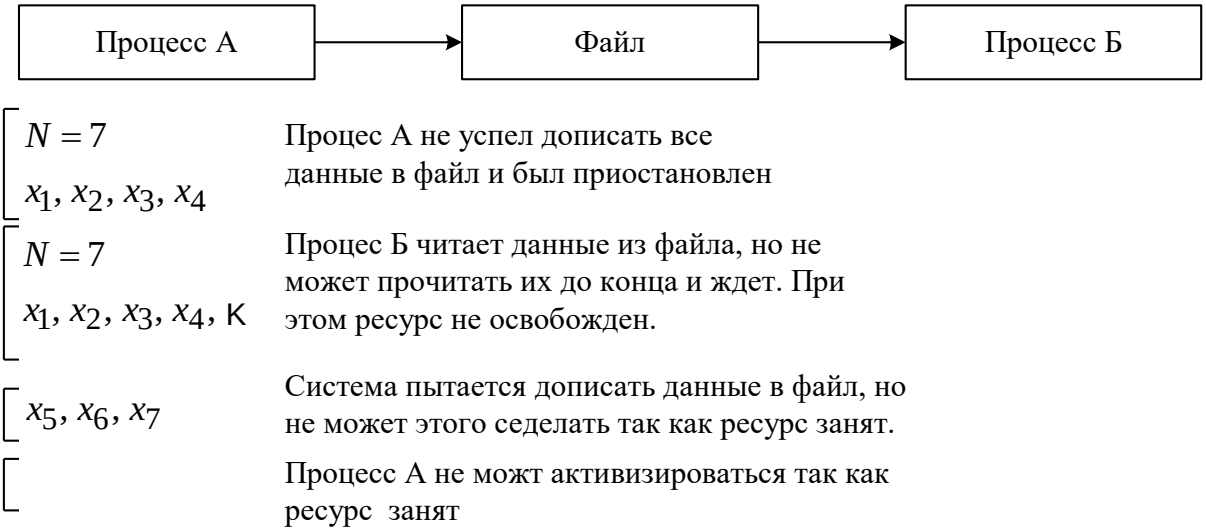


Рис. 2.5. Взаимная блокировка процессов

Процесс А записывает в файл 7 чисел с указанием их количества, но в файл записывается только первые 4 из них, например, остальные еще

находятся в буфере обмена с файлом. В это время получает управление процесс Б, который открывает общий файл и читает данные. Не получив ожидаемое количество чисел процесс Б ждет, продолжая оставлять ресурс занятым. Так как файл остается открытым система не может дописать последние числа. Данная ситуация приводит к тупику, так как процесс А не может записывать новые данные пока процесс Б не освободил файл. Процесс Б не может освободить файл, пока он не прочитает ожидаемого количества чисел.

Решение проблемы возникновения тупиков может быть следующим:

- Предотвращение тупиков. Данный подход предусматривает разрешение ситуаций с возможностью возникновения тупиков на уровне исходного текста.
- Распознавание тупиков. Определение взаимных блокировок может быть построено на основании таблиц запросов к ресурсам.
- Восстановление системы после тупиков. Возникший тупик может быть разрешен за счет снятия или отката к предыдущим этапам выполнения отдельных процессов.

Нити

Нить подобна процессу, каждая нить выполняется строго последовательно и имеет свой собственный программный счетчик и стек. Нити могут находиться в одном из следующих состояний: выполнение, ожидание и готовность.

Нити одного процесса имеют одно и то же адресное пространство и общие глобальные переменные. Наличие полного доступа ко всем виртуальным адресам остальных нитей не позволяет организовать защиту данных отдельных нитей процесса.

Нити имеют собственные:

- программный счетчик,
- стек,
- регистры,
- нити-потомки,
- состояние.

Нити разделяют:

- адресное пространство,
- глобальные переменные,
- открытые файлы,
- таймеры,
- семафоры,
- статистическую информацию.

3. Управление памятью

Память является важнейшим ресурсом, требующим тщательного управления со стороны операционной системы. Традиционно в самых младших адресах располагается сама операционная система.

Важнейшими функциями операционной системы по управлению памятью является

- отслеживание свободной и занятой памяти;
- защита занятой памяти от проникновения в нее других процессов;
- выделения памяти процессам;
- освобождение памяти при завершении процессов и очистка освободившейся памяти от кода и данных принадлежавших завершенному процессу;
- вытеснение процессов из оперативной памяти в свопинг файл на диске, возвращение процессов из свопинга;
- отслеживание перемещения сегментов памяти и соответствия логических и физических адресов переменных.

Работу с адресами в программах можно разделить на три уровня:

- символические имена – имена переменных и метки в программе;
- виртуальные адреса – адреса вырабатываемые компилятором для переменных и меток программы;
- физические адреса – реальные адреса размещения операторов и переменных в памяти компьютера в процессе выполнения.

Иллюстрация работы с адресами приведена на рис. 3.1.

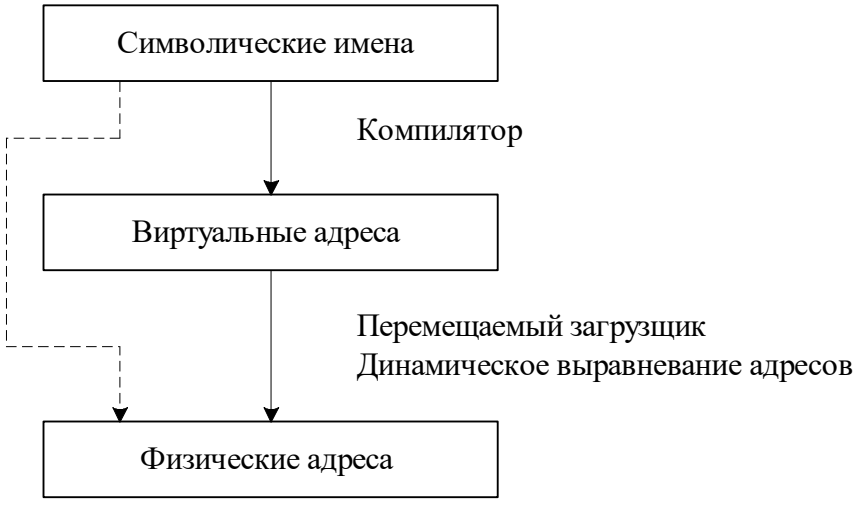


Рис. 3.1. Типы адресов

Преобразование виртуальных адресов в физические может выполняться перемещаемым загрузчиком операционной системы в процессе запуска или перемещения процесса в памяти (например, выгрузка не активного процесса в файл свопинга на диск). Кроме того, существует система

динамического выравнивания адреса реализованная аппаратно в процессоре при преобразовании из линейного адреса в физический адрес.

Пунктирная линия на рис. 3.1 показывает режим работы с конкретными физическими адресами в программе. Например, два процесса обмениваются данными, расположенными в заранее определенной области физической памяти.

Методы распределения памяти

Методы распределения памяти можно разделить на методы использующие обмен с использованием дискового пространства и методы, не использующие дисковое пространство. Классификация методов распределения памяти приведена на рис. 3.2.

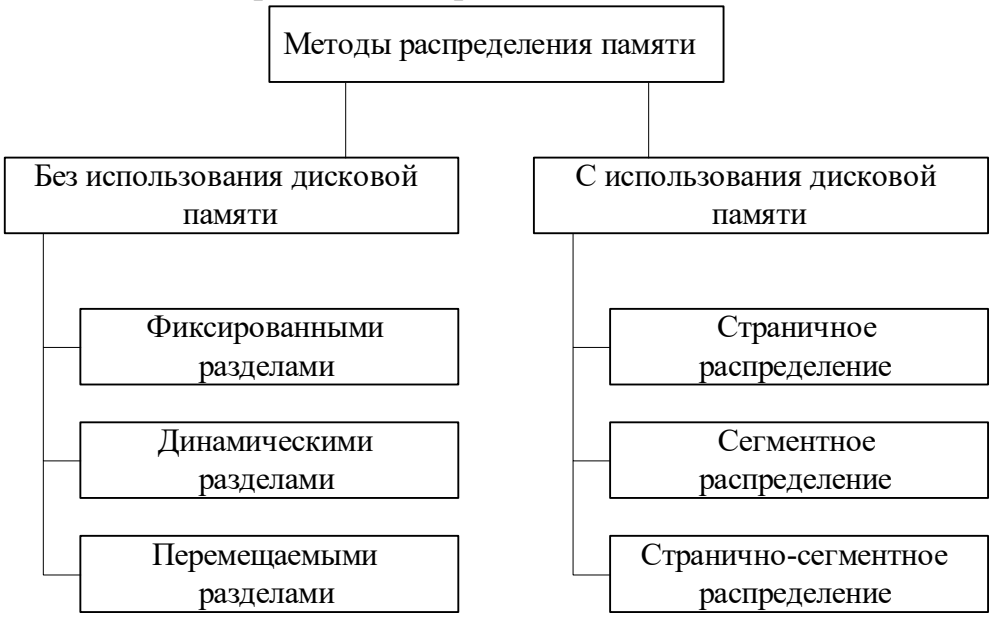


Рис. 3.2. Классификация методов распределения памяти

Методы распределения памяти без использования дисковой памяти

Фиксированные разделы

Распределение памяти с фиксированными разделами показано на рис. 3.3.

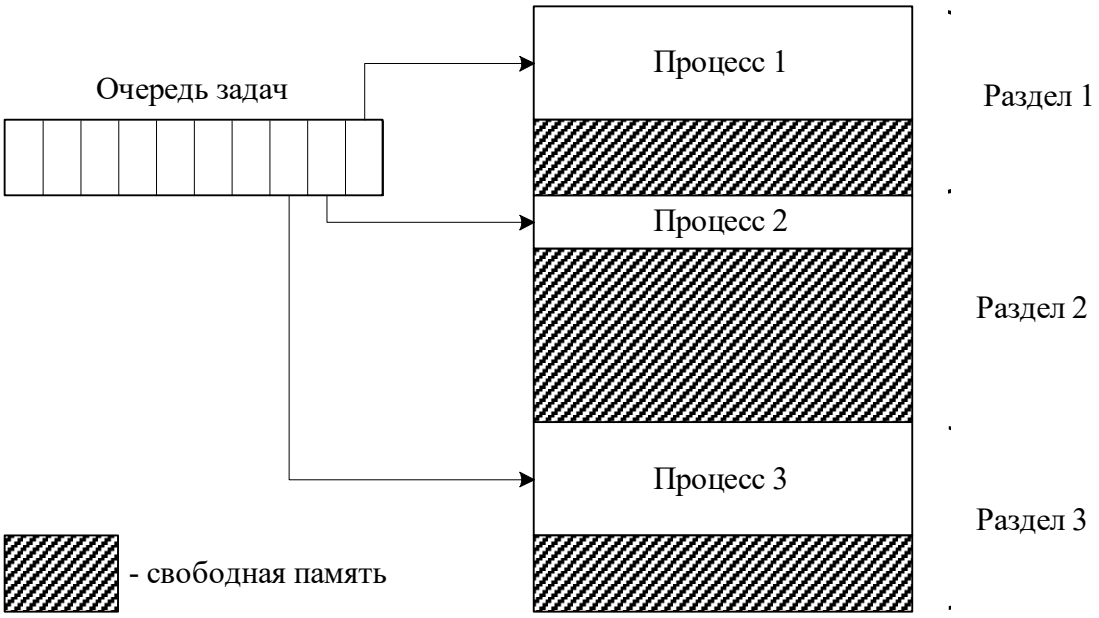


Рис. 3.3. Распределение памяти с фиксированными разделами

Распределение памяти с фиксированными разделами – самое простое в реализации. Оперативная память делится на разделы, например это, может быть выполнено программистом. Процессы из очереди задач загружаются в раздел соответствующего размера. Однако при таком способе распределения памяти, как это видно на рис. 3.3, большое количество оперативной памяти остается не использованной.

Динамические разделы

Распределение памяти с динамическими разделами представлено на рис. 3.4.

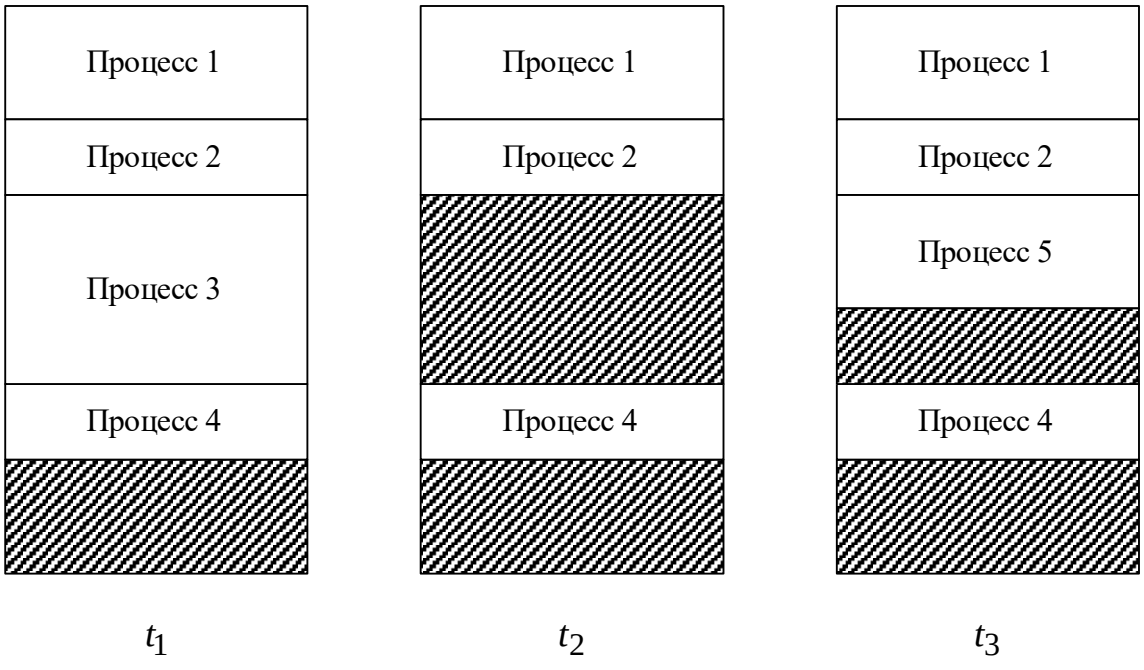


Рис. 3.4. Распределение памяти с динамическими разделами

Первоначально (момент времени t_1) процессы последовательно занимают память. В момент времени t_2 процесс 3 завершил свою работу, занятая им ранее память остается свободной. При запуске следующего приложения (процесс 5, в момент времени t_3) операционная система просматривает свободные разделы памяти и занимает под процесс первый найденный свободный раздел достаточного размера.

Длительная работа операционной системы с динамическими разделами приводит к фрагментации памяти – наличия большого количества несмежных свободных участков памяти, что может привести к невозможности запустить процесс, требующий размера большего, чем самый большой свободный участок.

Перемещаемые разделы

При распределении памяти на основе перемещаемых разделов операционная система периодически проводит сбор свободных разделов образовавшихся при фрагментации памяти. Иллюстрация данного способа приведена на рис. 3.5.

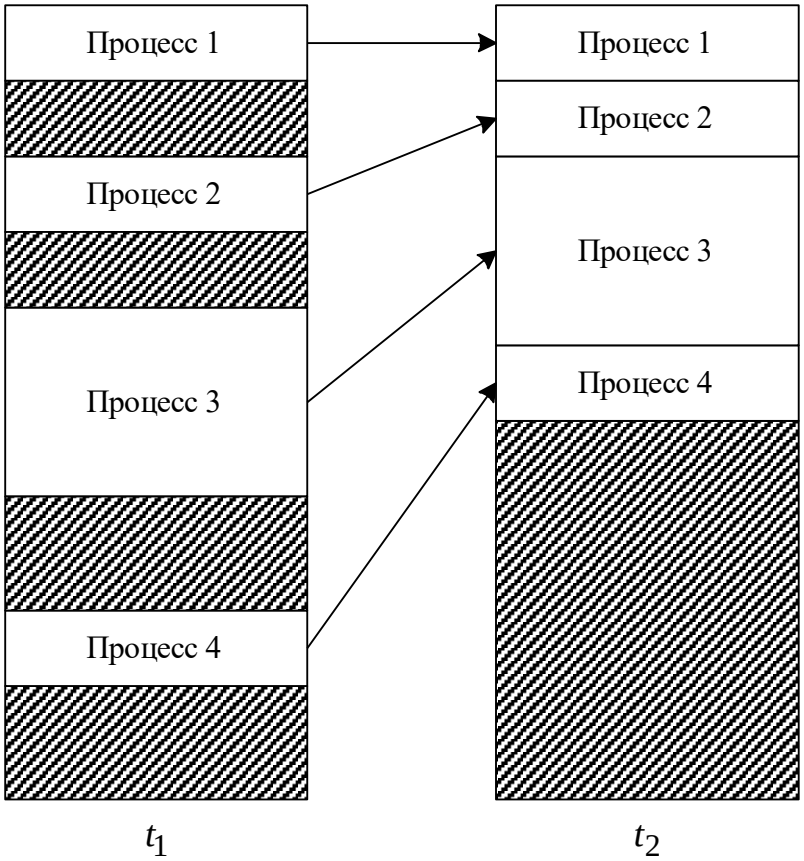


Рис. 3.5. Иллюстрация распределения памяти с перемещаемыми разделами

В момент времени t_1 система начинает сбор свободных разделов памяти для получения одного непрерывного раздела.

Данный способ работы с памятью позволяет предоставить максимальные ресурсы памяти для запуска нового процесса. Но при этом требует пересчета виртуальных адресов в физические адреса практически для всех процессов при каждой процедуре сбора свободных разделов.

Методы распределения памяти с использованием дисковой памяти

Довольно часто программисту приходится разрабатывать программы, размер которых превосходит размер оперативной памяти. Разработка их в этом случае производится с использованием виртуальной памяти.

Виртуальная память позволяет:

- организовать размещение части данных и кода процессов на дисковой (или другой) памяти;
- организовать автоматическую подгруздку необходимых данных и процедур в оперативную память;
- автоматически организовать преобразования виртуальных адресов в физические.

Организация виртуальной памяти с использованием дискового пространства организуется в виде: страничного, сегментного и странично–сегментного распределения памяти. Наиболее распространенных способов организации виртуальной памяти с выгрузкой на диск это работа с файлом свопинга.

Страничное распределение

Иллюстрация страничного распределения памяти приведена на рис. 3.6.

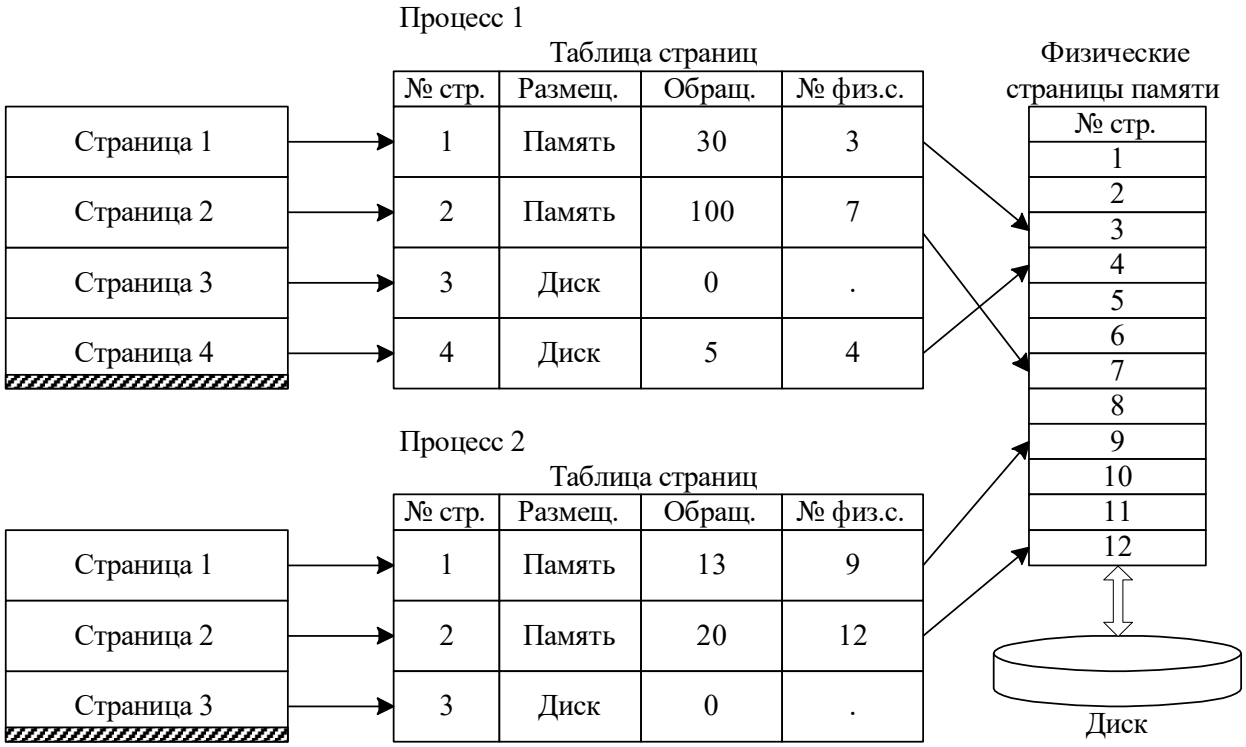


Рис. 3.6. Страничное распределение памяти

Каждый процесс, представленный на рис. 3.6, состоит из нескольких виртуальных страниц памяти. Для связи с физическими страницами для каждого из процессов строится таблица страниц, в которой размещается следующая информация: номер виртуальной страницы, способ размещения (память/диск), счетчик кол-ва обращений, номер физической страницы.

Физические страницы памяти могут располагаться в произвольном порядке. Операционная система выполняет функции загрузки требуемых процессам страниц памяти с диска и, при необходимости, а так же недостатке свободной памяти, выгрузке редко используемых страниц на диск.

Размер страницы во всех операционных системах выбирается кратным 2^n, чаще всего используются размеры 512 или 1024.

Механизм формирования физического адреса при страничном распределении памяти представлен на рис. 3.7.

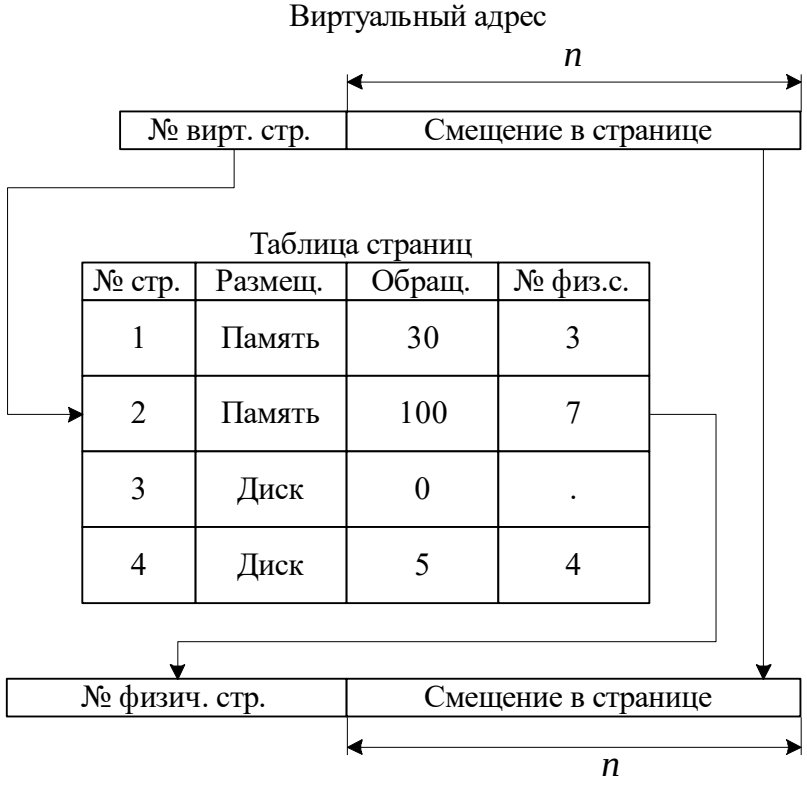


Рис. 3.7. Механизм формирования адреса

Виртуальный адрес, состоящий из номера виртуальной станицы и смещения в ней, преобразуется следующим образом: младшие n разрядов адреса, соответствующие смещению в странице передаются в младшие разряды физического адреса; номер физической страницы извлекается из таблицы страниц процесса по номеру виртуальной страницы, причем размеры адресов могут не совпадать.

Сегментное разделение

Главный недостаток страничного распределения памяти это одинаковый размер страниц и не возможность организации доступа нескольких процессов к одному модулю. Эти недостатки устранены при сегментном способе распределения памяти, представленном на рис. 3.8. Кроме того, сегментное разделение памяти предусматривает установление

различных атрибутов для сегментов – доступ только по чтению, по записи и другие.

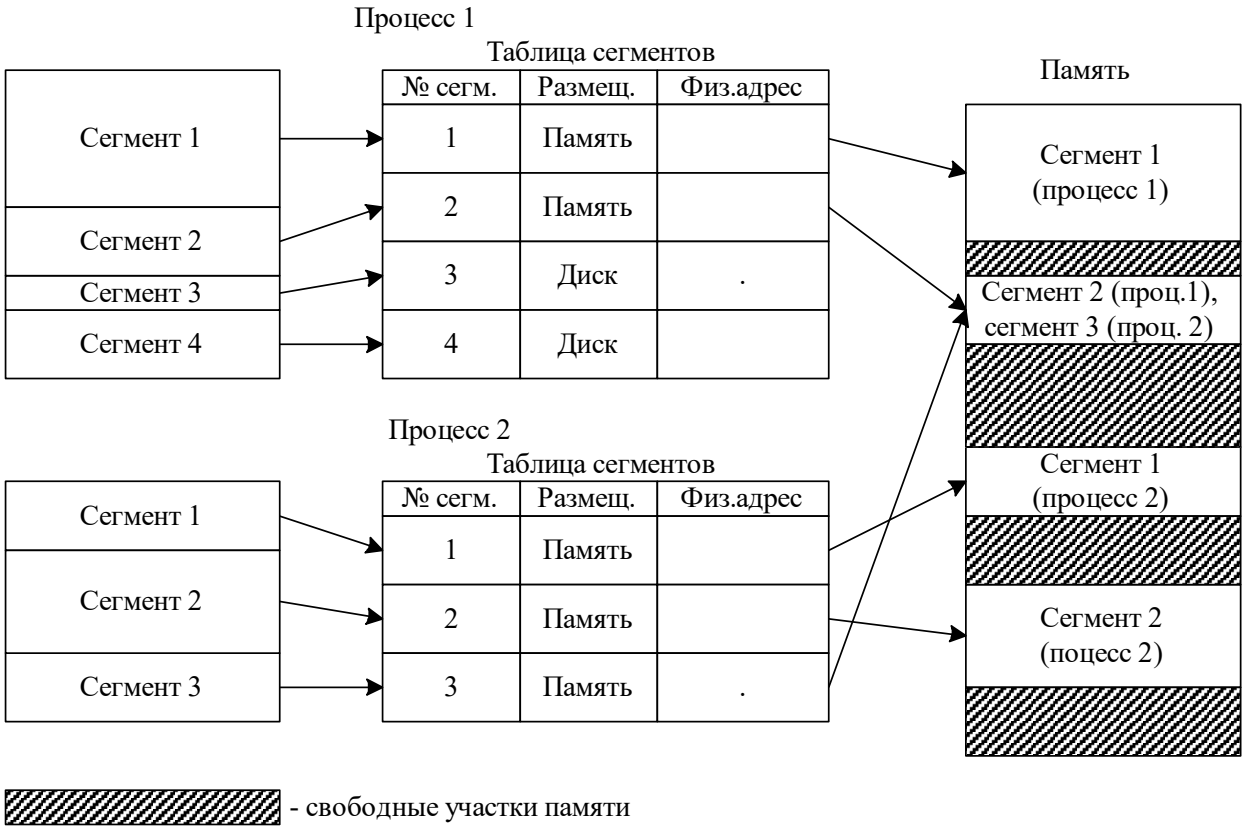


Рис. 3.8. Сегментное распределение памяти

При запуске процесса используемая часть сегментов помещается в оперативную память. Для каждого из сегментов, операционная система выбирает подходящий по размеру свободный участок памяти. Размер сегмента определяется программистом при написании программы (по умолчанию устанавливается компилятором). При этом возможен одновременный доступ нескольких процессов к одному сегменту, например к общей функции или сегменту данных.

Работа с сегментной организацией памяти – формирование адреса выгрузка/загрузка с диска аналогична используемой при страничной организации памяти, однако кроме этого проверяются атрибут доступа к сегменту.

Странично–сегментное распределение памяти

Странично–сегментное распределение памяти предусматривает сочетание изложенных выше подходов. Каждый определенный программистом сегмент, в этом случае состоит из группы страниц. Механизм формирования адреса в сранично–сегментном способе распределения памяти представлен на рис. 3.9.

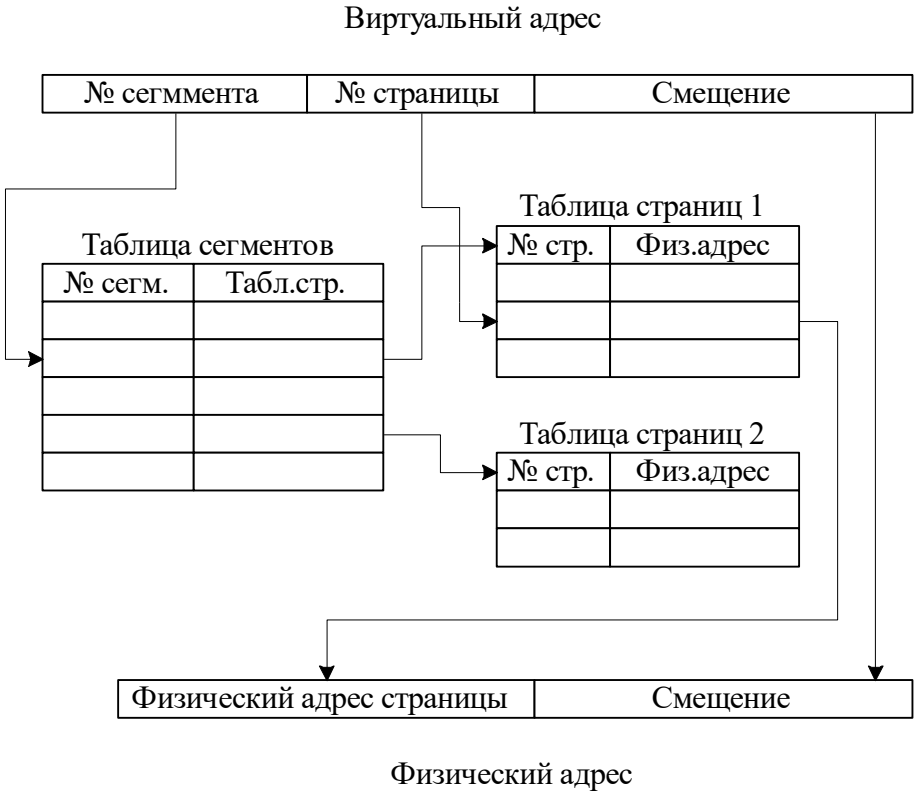


Рис. 3.9. Странично–сегментное распределение памяти

При странично–сегментное распределение памяти операционная система по номеру сегмента находит таблицу страниц этого сегмента. Номер страницы в виртуальном адресе позволяет сформировать физический адрес начала страницы в памяти компьютера. Смещение из виртуального адреса передается в младшие разряды физического адреса без изменений.

Иерархия запоминающих устройств

Иерархия запоминающих устройств приведена на рис. 3.10.

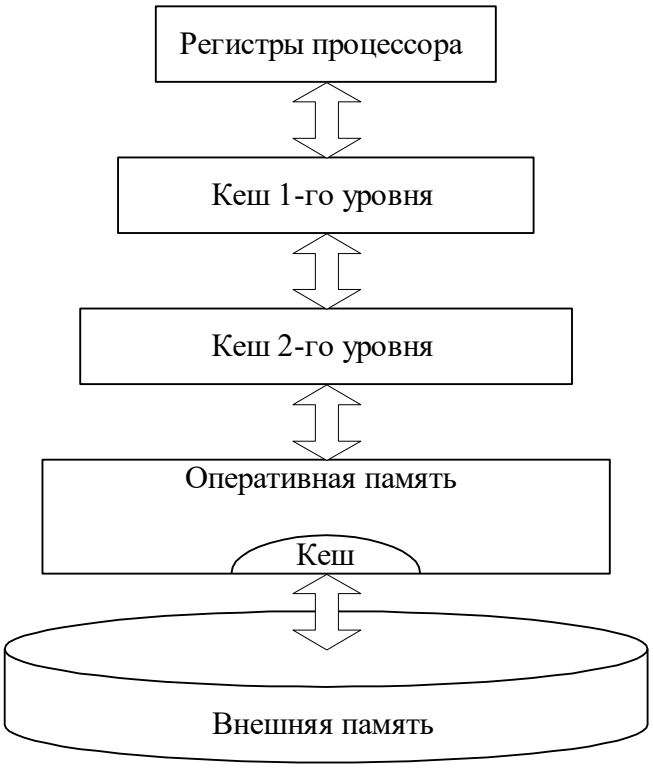


Рис. 3.10. Иерархия запоминающих устройств

Самым быстродействующим устройством памяти компьютера являются регистры процессора. В 16 пользовательских и 16 системных регистрах процессор выполняет все действия, которые реализуются на компьютере. Кеш память 1-го уровня располагается в процессоре и работает на полной частоте ядра процессора. Наличие кэш памяти – сверх быстрой оперативной памяти, позволяет процессору перегрузить данные в кэш и обработать их на высокой скорости. В современных процессорах разделяют внутренние КЭШи для данных, инструкций и команд. В виду высокой стоимости сверх быстрой оперативной памяти, кэш 1-го уровня стараются делать не большим, и на современных компьютерах используется кэш 2-го уровня, работающая на половинной частоте ядра. Кеш 2-го уровня медленнее, чем кэш 1-го уровня, но значительно превосходит по быстродействию основную память.

Все данные и выполняемые программы размещаются в оперативной памяти. Отдельный участок оперативной памяти выделяется для кэширования внешней памяти – например жестких дисков. Кэш память физических устройств позволяет организовать буфер объемна с устройством.

Организация кэш памяти представляется на рис. 3.11.

Физический адрес в памяти	Данные	Служебная информация	
		Бит модификации	Бит обращения

Рис. 3.11. Организация кэш памяти

Данные, часто используемые процессором, перегружаются в кэш, при этом запоминается физический адрес их размещения в основной памяти. Если процессор обращается к ячейке, значение которой перегружено в кэш, то на обращение процессора следует более быстрый ответ от кэша, аннулирующий запрос к основной памяти. При обращении к кэшированным данным устанавливаются биты модификации и обращения. Периодически процессор проверяет содержимое кэш памяти и выгружает те ячейки, к которым не было обращения длительное время или, в случае дефицита ресурсов, обращение производится реже.

Средства аппаратной поддержки управления памятью в процессорах Intel 80386 и выше

Процессоры i386 и выше имеет два режима работы – реальный и защищенный. В реальном режиме процессор работает так же как 8086. В защищенном режиме процессор может использовать 32-х разрядный режим доступа к памяти, в том числе механизмы поддержки виртуальной памяти и механизмы переключения задач. В защищенном режиме процессор для каждой задачи процессор может эмулировать виртуальные процессоры, имеющие общую память со страничной организацией.

Средства поддержки сегментации памяти

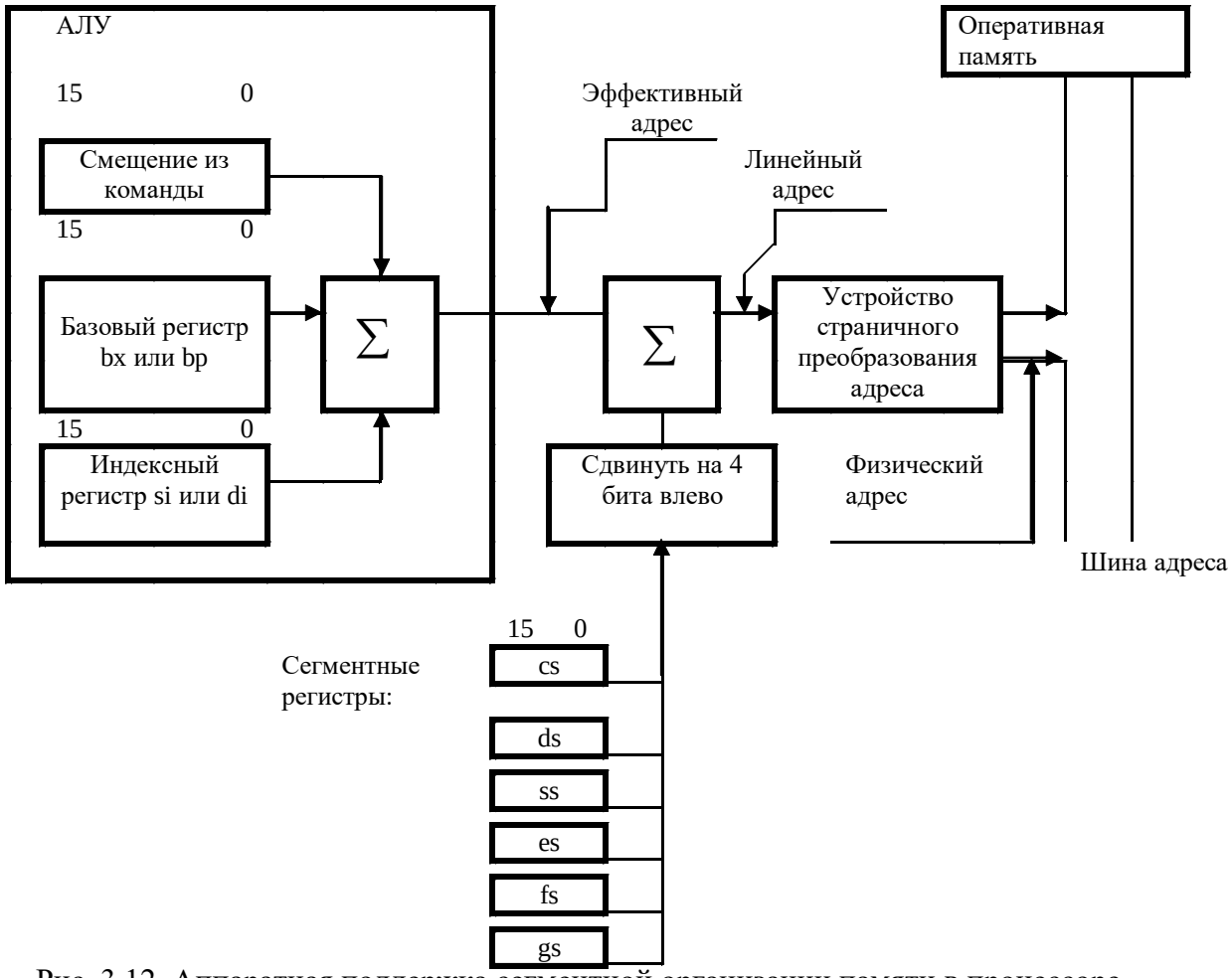


Рис. 3.12. Аппаратная поддержка сегментной организации памяти в процессоре

Адресное пространство процессора i386 с 32-х разрядной шиной адреса составляет 4 Гбайт. Архитектура процессора Intel приведена на рис.3.12. Виртуальный адрес, состоит из номера сегмента и смещение внутри сегмента. В зависимости от способа адресации используемого в команде смещение может получаться непосредственно из команды, или с помощью косвенной адресации. Номер сегмента выбирается из одного из сегментных регистров процессора: CS, SS, DS, ES, FS или GS.

Средства сегментной организации памяти образуют верхний уровень средств управления виртуальной памятью процессора, а средства страничной организации – нижний уровень. Структура формирования адреса при сегментной организации памяти приведена на рис. 3.13.

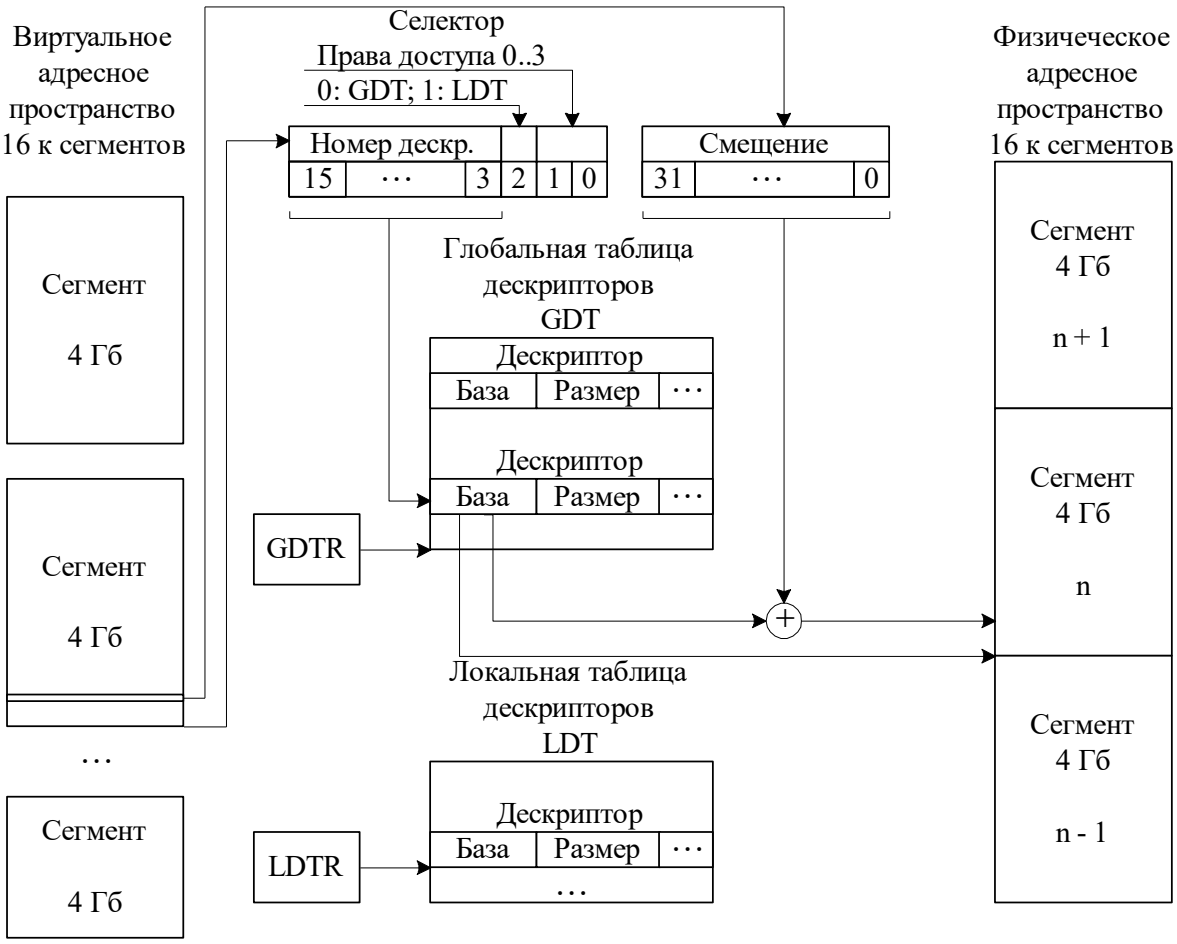


Рис. 3.13. Структурная схема поддержки сегментной организации памяти

Смещение (32 бита) позволяет адресовать $2^{32} = 4\,\text{Гб}$ памяти. Количество сегментов определяется размером поля селектора $2^{13} = 8192$ сегментов. Служебные поля селектора определяют режим доступа (2 бита) и признак таблицы дескрипторов: 0 – GDT, 1 – LDT. Глобальная таблица – GDT, используется операционной системой и для организации межзадачного взаимодействия. Локальная таблица дескрипторов – LDT создается для каждой из задач. База из таблицы дескрипторов указывает на

начало сегмента физической памяти. Сумма базы и смещения указывает на адрес операнда в физической памяти.

Адреса таблиц хранятся в регистрах GDTR и LDTR.

Странично–сегментный механизм

Работа странично–сегментного механизма в целом повторяет работу страничного режима. Физическое и виртуальное адресные пространства разбиты на страницы размером 4 К.

Линейный виртуальный адрес содержит в своих старших разрядах номер виртуальной страницы, а в младших 12 разрядах смещение внутри страницы. Схема формирования адреса приведена на рис. 3.14.

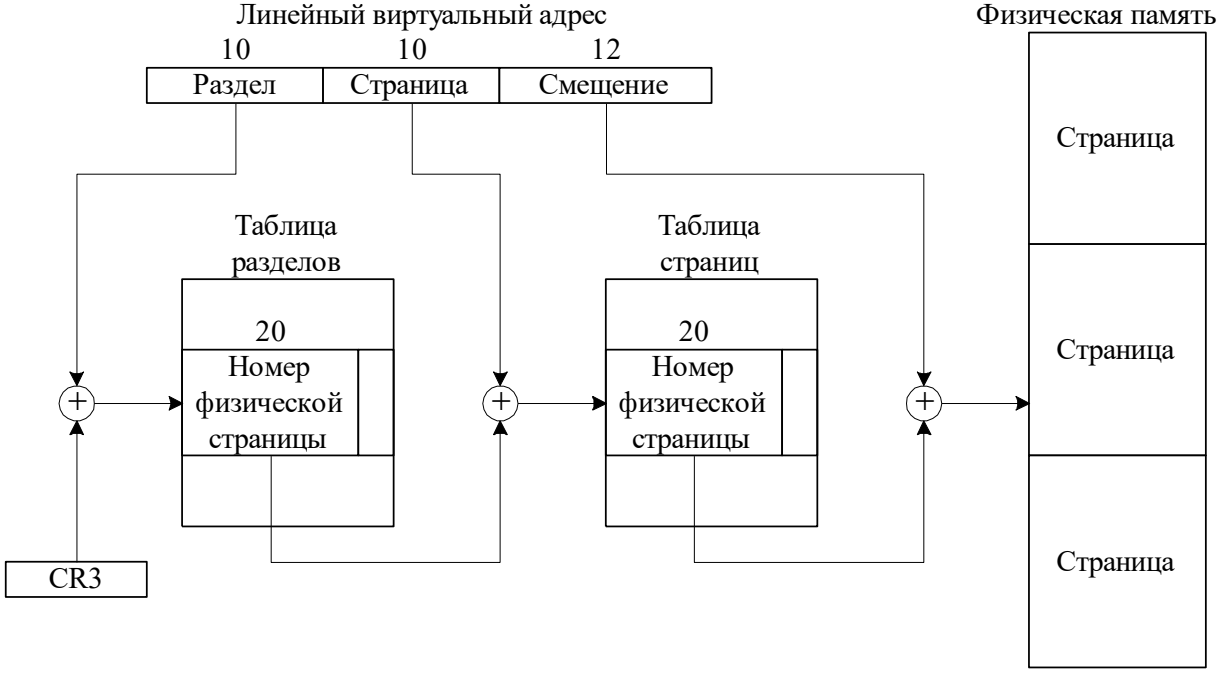


Рис. 3.14. Формирование адреса при странично–сегментном механизме управления памяти

Реализация странично–сегментной организации виртуальной памяти в процессорах Intel позволяет обеспечить защиту переменные и операторы различных процессов за счет:

1. изоляции адресных пространств процессов в физической памяти путем назначения им различных физических страниц или сегментов;

2. защиту сегментов от несанкционированного доступа с помощью многоуровневой системы привилегий.

Вызов программ

Вызов подпрограмм осуществляется с помощью команд JMP и CALL. Схема вызова подпрограммы приведена на рис. 3.15.

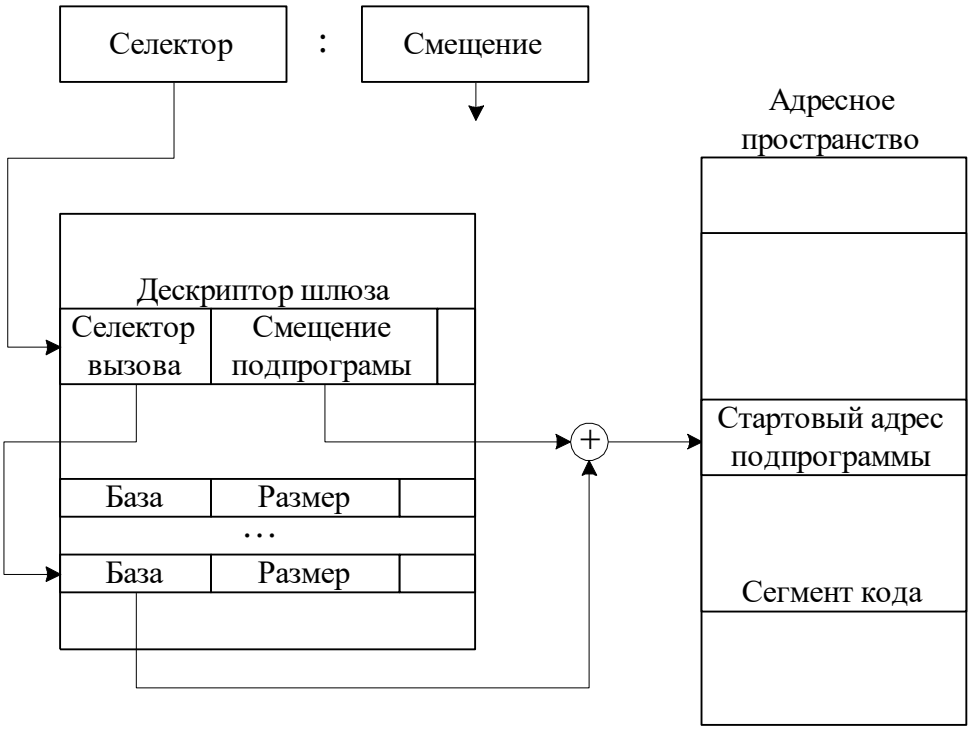


Рис. 3.15. Вызов подпрограммы

Простейший вызов может быть организован с помощью команд JMP или CALL указанием полного адреса (селектор : смещение) вызываемой процедуры. Однако данный способ позволяет вызывать из программы только пользовательские подпрограммы, имеющие 3-й уровень приоритета. Вызов системных функций, имеющих 0-й уровень приоритета, таким способом будет невозможен.

Для решения этой проблемы в процессоре есть другой способ вызова подпрограмм основанный на заранее определенном наборе точек входа в привилегированные кодовые сегменты. Эти точки входа описываются с помощью специальных дескрипторов – шлюзов вызова подпрограмм.

Данный способ вызова показан на рис. 3.15. Значение селектора определяет дескриптор шлюза в таблице. Из полей дескриптора определяется смещение и ссылка на значение базы, их сумма определяет стартовый адрес подпрограммы.

При определении адреса входа в вызываемом сегменте смещение из поля команды CALL не используется, а используется смещение из дескриптора шлюза, что не дает возможности задаче самой определять точку входа в защищенный кодовый сегмент.

При вызове процедур, с различными уровнями привилегий, возникает проблема передачи параметров между различными стеками, так как для надежной защиты задачи различного уровня привилегий имеют различные сегменты стеков. Селекторы этих сегментов хранятся в контексте задачи – сегменте TSS (Task State Segment). Структуру сегмента TSS можно найти в [].

4. Управление вводом/выводом и файловая система
4.1. Управление вводом/выводом

Одной из главных функций операционной системы является управление устройствами ввода–вывода компьютера. Операционная система должна обеспечивать интерфейс между устройствами и остальной частью системы. Интерфейс должен быть одинаковым для всех типов устройств – независимость от устройств.

Внешнее устройство состоит из самого устройства и программно–аппаратного контроллера. Операционная система работает с контроллером, обрабатывающим стандартизованный набор функций. Каждый контроллер имеет набор регистров, являющихся частью физического адресного пространства памяти и доступных для специальных операций ввода/вывода IN и OUT.

Программное обеспечение ввода/вывода разбито на четыре слоя, представленных на рис. 4.1. Каждый слой имеет свои системные функции, обеспечивающие его работу.

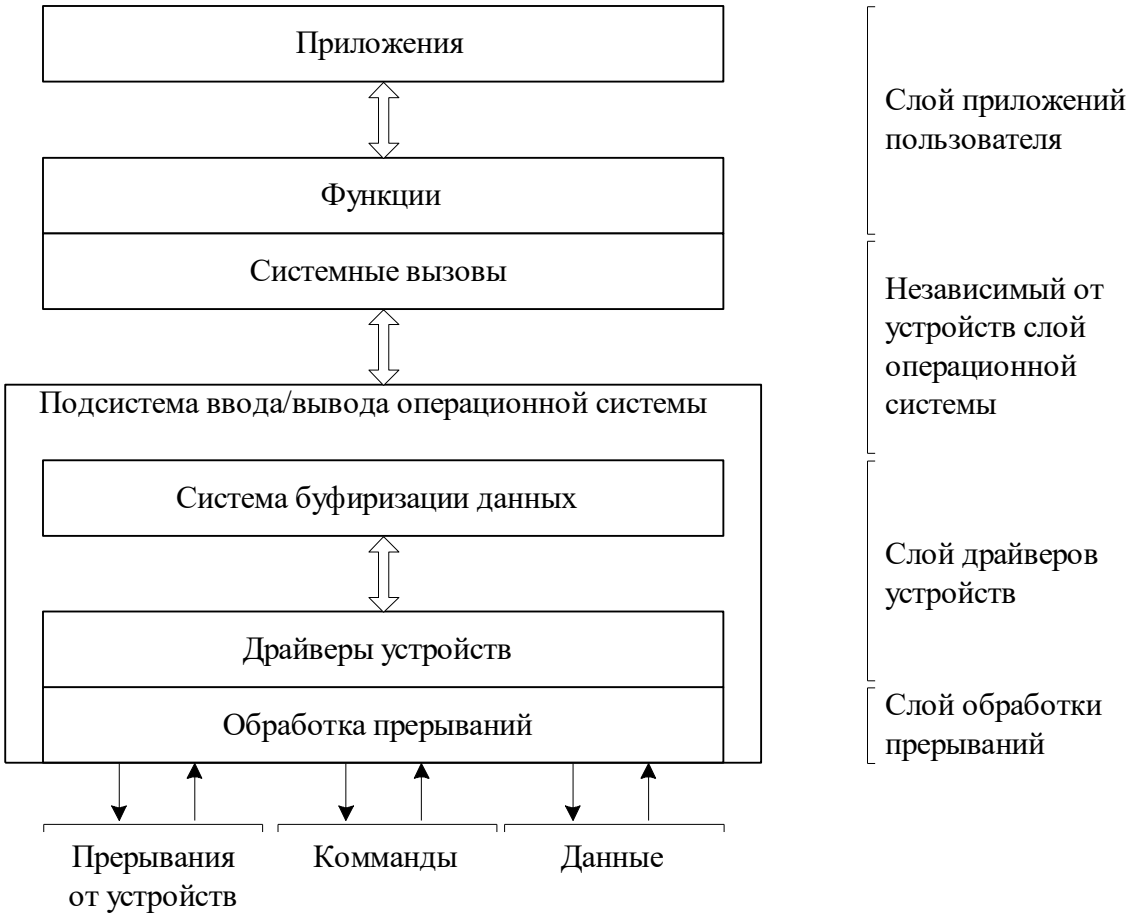


Рис. 4.1. Иерархия слоев ввода/вывода

Слой обработки прерываний

Прерывания – это готовые процедуры, вызываемые процессором для решения определенных задач. Прерывания делятся на аппаратные и

программные. Аппаратные прерывания инициализируются событиями, произошедшими в аппаратуре компьютера. Программные прерывания – содержат системные заготовки для решения многих задач, возникающих у программиста в процессе написания программы.

Когда вызывается прерывание, процессор прекращает обслуживание текущий задачи, запоминая адрес точки останова – CS:IP, а так же, содержимое системных регистров в стеке. Затем процессор вызывает системную функцию. После завершения прерывания процессор возвращается к выполнению прерванной задачи. Прерывания обладают различными уровнями приоритетов, например, в случае возникновения аппаратного прерывания, порывается исполнение не только программ пользователя, но и выполнение программных прерываний.

Адреса программ обслуживающих прерывания называются векторами, каждый вектор имеет длину 4 байта. Они хранятся в таблице векторов прерываний, начиная с адреса 0000:0000. Всего современные компьютеры поддерживают 256 векторов прерываний. Таблица векторов прерываний заполняется в процессе процедуры загрузки компьютера.

Для управления аппаратными прерываниями в компьютерах IBM используется микросхема контроллера прерываний Intel 8259. Схема приоритетов Intel 8259 поддерживает 2 набора по 8 уровней приоритетов внешних событий. Микросхема позволяет маскировать отельные прерывания, эта операция целесообразна в момент перенастройки системы прерываний или в случае если прерывание данной функции прерывания является не допустимым. Кроме этого, программисту предоставляется возможность заменить системную функцию обработки прерывания на свою функцию, указав новый адрес в таблице прерываний. Таким способом пользуются большинство компьютерных вирусов. Изменение функции прерывания может привести к серьезным сбоям в работе программного обеспечения компьютера, по этому в большинстве случаев разрабатывается дополнение к существующей функции прерывания.

Современные операционные системы стараются срыть прерывания как можно глубже в недрах операционной системы, чтобы как можно меньшая часть операционной системы имела с ними дело.

Слой драйверов устройств

Драйвер устройства содержит весь аппаратно–зависимый код для обеспечения бесперебойной работы устройства. Типичными функциями драйвера является обеспечение обмена блоками данных между устройством и программным слоем.

Независимый от устройств слой операционной системы

Большая часть программного обеспечения ввода/вывода является независимой от устройств. Типичными функциями для независимого от устройств слоя являются:

- обеспечение интерфейса к драйверам устройств;

- поддержка символических имен устройств;
- защита устройств;
- обеспечение независимого размера блока;
- буферизация;
- распределение памяти на блок-ориентированных устройствах;
- распределение и освобождение выделенных устройств;
- уведомление об ошибках.

Слой приложений пользователя

К пользовательскому слою относятся функции библиотек языков программирования, или аналогичные функции программиста, реализующих обмен данными с устройствами.

4.2. Файловая система

Файловая система – это часть операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю удобный интерфейс при работе с данными, хранящимися на диске, и обеспечить совместное использование файлов несколькими пользователями и процессами.

В широком смысле понятие "файловая система" включает:

- совокупность всех файлов на диске;
- наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске;
- комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами.

Под файлом понимается любая поименованная информация на логическом диске. Данное определение предусматривает, что файла есть имя, размер и служебные атрибуты.

Имена файлов, типы файлов

Операционные системы используют различные правила формирования имен файлов. Так, например, в операционных системах MS DOS, Novell NetWare используются имена файлов длиной до 8 символов (без учета регистра) и 3 символа выделяются для расширения файла. Операционная система UNIX на имя файла выделяет 14 символов; операционная система Windows предоставляет возможность работать пользователю с длинными именами файлов, до 256 символов, в том числе с использованием символов национальных алфавитов.

Кроме обычных файлов в операционных системах выделяют специализированные типы файлов:

Специальные файлы – файлы, ассоциированные с устройствами ввода-вывода. Пользователь может работать с ним как с обычными файлами, но операционная система преобразует данные действия у команды управления устройством.

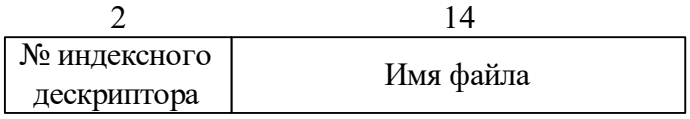
Каталог (папка) – файл, содержащий системную информацию о группе файлов. В каталоге содержится список файлов, входящих в него и их атрибуты, устанавливается соответствие между файлами и их физическим расположение на диске.

В разных файловых системах могут использоваться в качестве атрибутов:

- информация о разрешенном доступе;
- пароль для доступа к файлу;
- владелец файла;
- создатель файла;
- признак "только для чтения";
- признак "скрытый файл";
- признак "системный файл";
- признак "архивный файл";
- признак "двоичный/символьный";
- признак "временный" (удалить после завершения процесса);
- признак блокировки;
- длина записи;
- указатель на ключевое поле в записи;
- длина ключа;
- времена создания, последнего доступа и последнего изменения;
- текущий размер файла;
- максимальный размер файла.

Структура записи каталогов MS DOS и UNIX представлена на рис. 4.2.

а)



б)

Рис. 4.2. Структура записи каталога *а* – MS DOS, *б* – UNIX

Кроме этого, различные операционные систем имеют и различную логическую организацию. Большинство современных операционных систем образуют древовидную иерархическую структуру файлов (рис. 4.3.б). Первым операционным системам была характера

одноуровневая система (рис. 4.3.а). Операционная система UNIX использует сетевую модель (рис. 4.3.в).

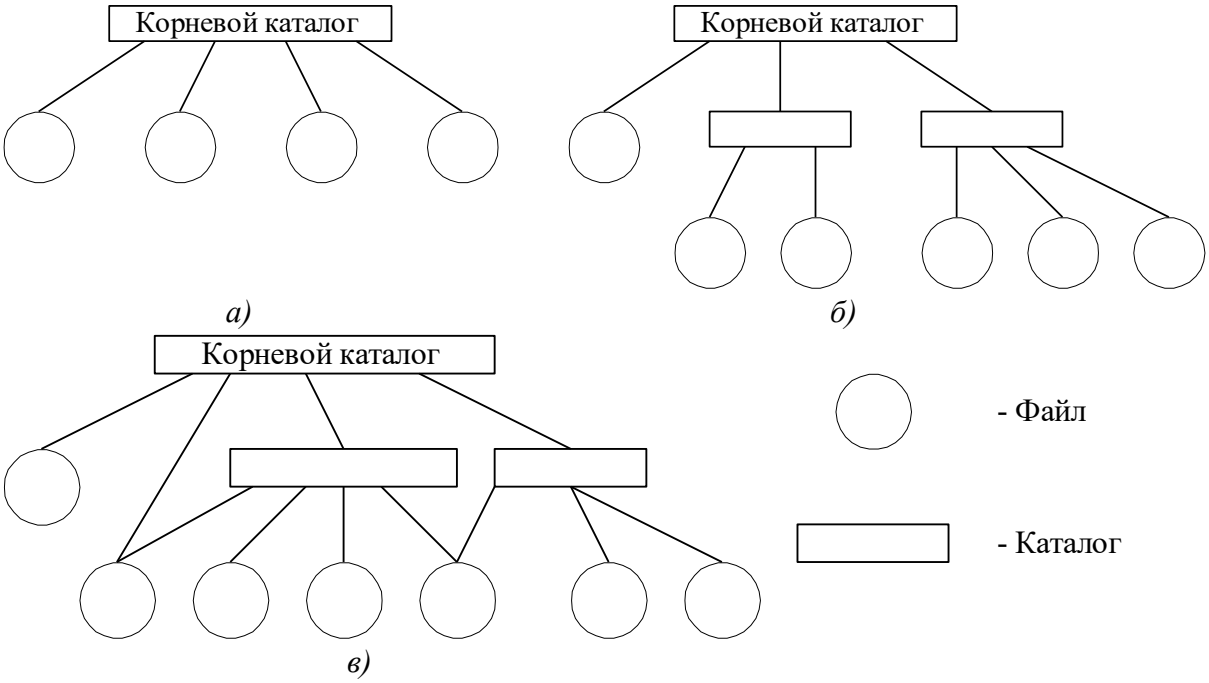


Рис. 4.3. Логическая организация файловой системы
а – одноуровневая, *б* – древовидная, *в* – сетевая

Физическая организация файла

Различные способы физической организации файла приведены на рис. 4.4.

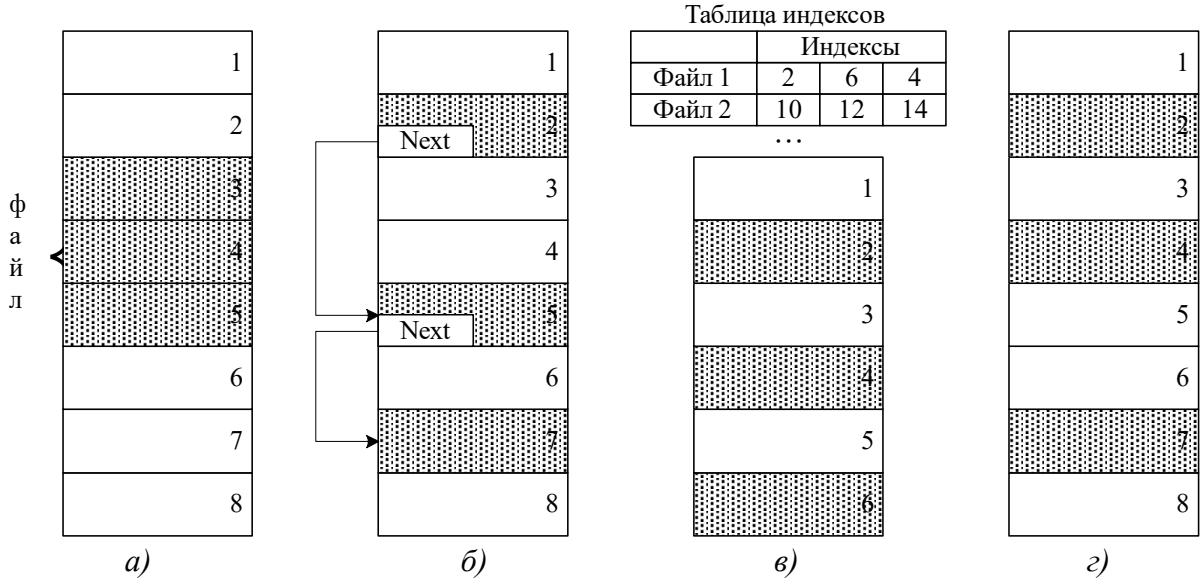


Рис. 4.4. Способы организации физических файлов
а – размещение файла в непрерывных областях; б – связный список блоков,
в - индексная таблица, г – перечень блоков

Простейший способ физической организации файла на устройстве (диске), это непрерывная цепочка блоков. Данный способ очень удобен для

дисковых накопителей, так как позволяет организовать быструю последовательную процедуру чтения блоков, без механического позиционирования по диску. Однако такой способ приводит к сильной фрагментации диска и создает большие проблемы при создании и редактировании файла, так как система не может сразу определить будущую длину файла.

Следующий способ – размещения файла в виде связанного списка блоков. Каждый блок файла, в этом случае содержит ссылку на следующий блок. Такой способ снимает проблемы с фрагментаций диска, допускает оптимизацию с последователем расположением блоков, требует для определения фала только ссылки на первый блок. Главный недостаток данного способа это значительные затраты времени требующиеся на позиционирование по файлу – для этого его приходится просматривать последовательно до обнаружения требуемого места.

Эффективный способ организации физического размещения файла это создание индексной таблицы. В MS DOS она называется FAT. Данный способ позволяет организовать позиционирование по файлу и не приводит к фрагментации диска.

Операционная система UNIX использует для организации физического расположения файла в виде простого перечисления блоков в структуре фиксированной длины. Для хранения адреса файла выделено 13 полей. Если для хранения файла достаточно 10 блоков, то номера этих блоков непосредственно перечислены в первых десяти полях адреса. Если размер файла больше 10 блоков, то следующее 11-е поле содержит адрес блока, в котором могут быть расположены еще 128 номеров следующих блоков файла. Если файл больше, чем 10+128 блоков, то используется 12-е поле, в котором находится номер блока, содержащего 128 номеров блоков, которые содержат по 128 номеров блоков данного файла. И если файл больше 10+128+128*128, то используется последнее 13-е поле для тройной косвенной адресации. Подробнее файловая структура UNIX рассмотрена в разделе 7.

Права доступа к файлу

Права доступа к файлу определяют для каждого пользователя набор возможных операций, применимых к данному файлу.

Современные операционные системы предлагают следующие варианты прав:

- создание файла,
- уничтожение файла,
- открытие файла,
- закрытие файла,
- чтение файла,
- запись в файл,
- дополнение файла,
- поиск в файле,
- получение атрибутов файла,
- установление новых значений атрибутов,
- переименование,
- выполнение файла,
- чтение каталога,

и другие операции с файлами и каталогами.

Права доступа задаются в виде матрицы прав, в которой столбцы соответствуют файлам, а строки пользователям. Пример матрицы прав приведен на рис. 4.5.

Пользователи	Файлы				
	prim.exe	data.dbf	text.doc	tabl.xls	base.mdi
krasov	выполн.	писать	–	писать	–
user	выполн.	читать	–	–	читать
test	–	читать	писать	–	писать

Рис. 4.5. Матрица прав доступа к файлам

Различают два основных подхода к определению прав доступа:

- избирательный доступ, когда для каждого файла и каждого пользователя сам владелец может определить допустимые операции;
- мандатный подход, когда система наделяет пользователя определенными правами по отношению к каждому разделяемому ресурсу в зависимости от того, к какой группе пользователь отнесен.

Общая модель файловой системы

Модель файловой системы может быть представлена в виде изображенном на рис. 4.6.

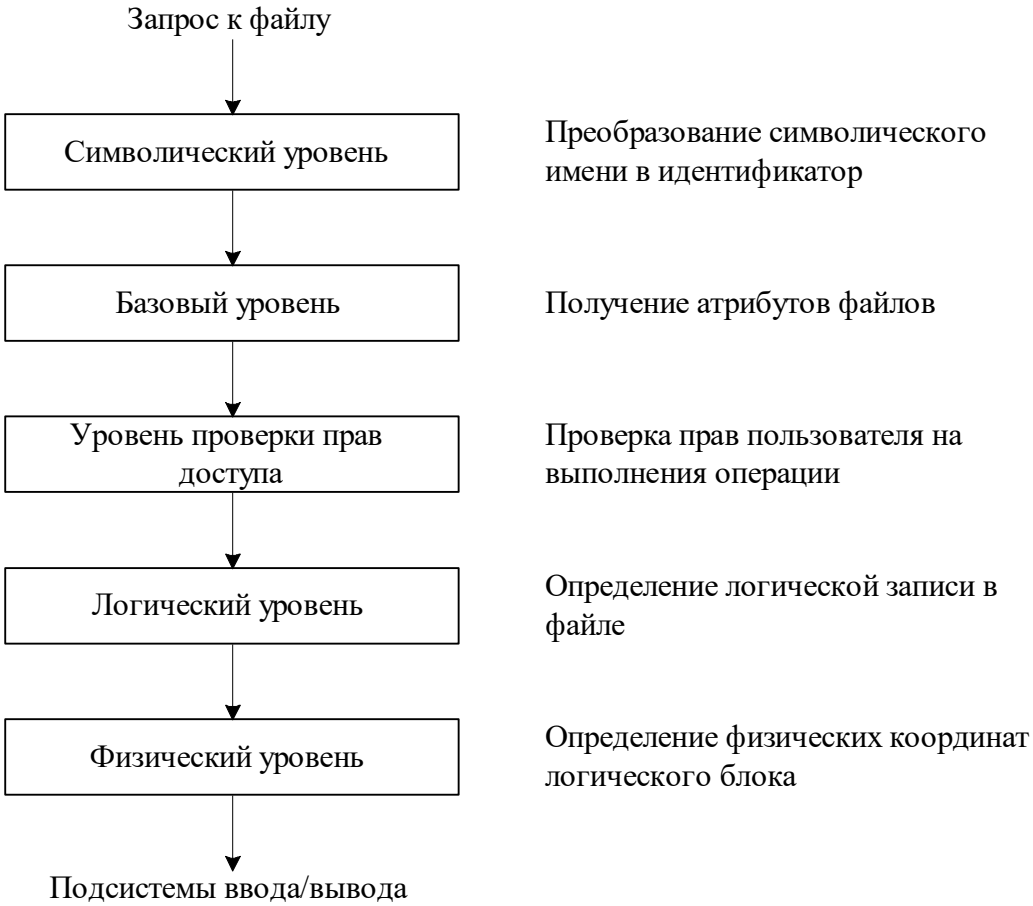


Рис. 4.6. Модель файловой системы

Символическое имя устройства, к которому обращается пользователь, преобразуется в его уникальный идентификатор. На следующем этапе получаются атрибуты файла и права доступа к файлу, они сравниваются с правами пользователя на возможность выполнения данного действия. После этого определяются логические координаты записи, по которым находятся физические координаты.

Для упрощения работы с файлами современные операционные системы предоставляют возможность отображения файла или его части в память,

Современный уровень развития средств вычислительной техники требует предоставление пользователю возможности работать сразу с несколькими файловыми системами. Многоуровневую файловую систему можно представить в виде изображенном на рис. 4.7.

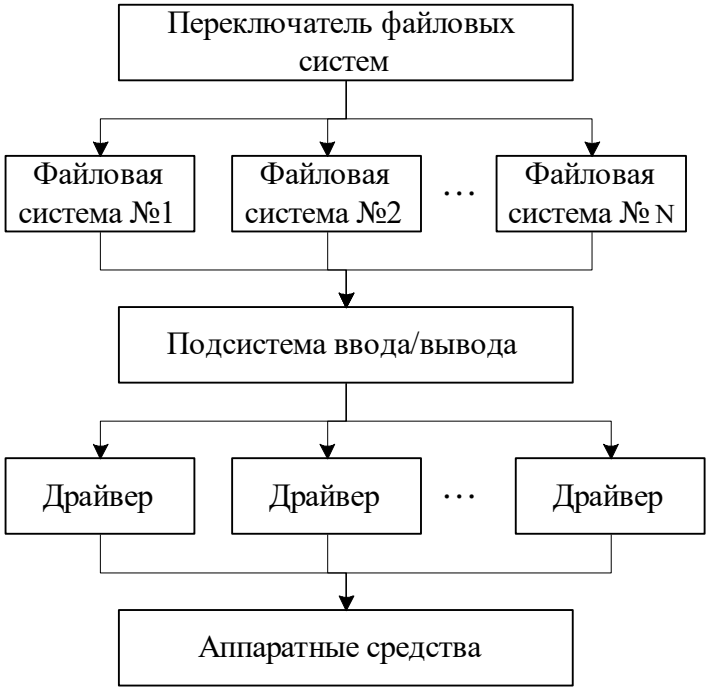


Рис. 4.7. Архитектура современной файловой системы

На верхнем уровне файловой системы располагается переключатель файловых систем. Он обеспечивает прозрачное для пользователя обращение к конкретной файловой системе. Файловая система обрабатывает этот запрос и передает его на подсистему ввода/вывода, откуда он поступает конкретный драйвер и дальше на аппаратное устройство.

5. Управление распределенными процессами

Отличительной чертой распределенных систем является необходимость организовать взаимосвязь между процессами, работающими на различных компьютерах по телекоммуникационным линиям.

Концепция удаленного вызова процедур *(Remote Procedure Call - RPC)*

Характерными чертами вызова локальных процедур являются:

- Асимметричность, то есть одна из взаимодействующих сторон является инициатором;
- Синхронность, то есть выполнение вызывающей процедуры приостанавливается с момента выдачи запроса и возобновляется только после возврата из вызываемой процедуры.

Реализация удаленного вызова процедур содержит дополнительные элементы, по сравнению с локальным вызовом процедур. Кроме того пи удаленном вызове процедур могут возникнуть проблемы, такие как обрыв связи, сбой в работе компьютера, обслуживающего один из взаимодействующих процессов, аппаратно–программные особенности данных компьютеров. Все это может привести к ситуации, когда один из взаимодействующих процессов будет безрезультатно ждать ответа.

В основе концепции удаленного вызова процедур заложена идея, сделать данный вызов максимально прозрачным для программиста. Взаимодействие программных компонентов при удаленном вызове показано на рис. 5.1.

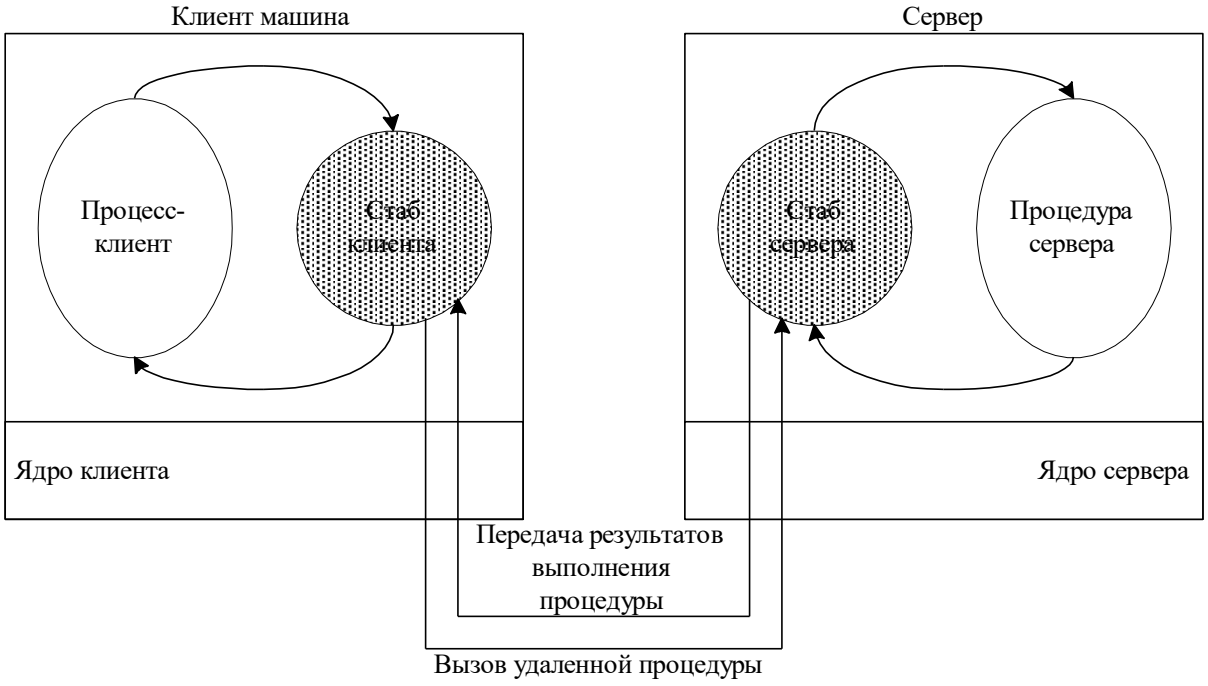


Рис. 5.1. Иллюстрация процесса удаленного вызова процедур

При вызове удаленной процедуры на машине клиента вызывается Стаб процедура – процедура заглушка, эмулирующая для процесса клиента вызов процедуры. Процедура Стаб клиента принимает параметры вызываемой процедуры и передает их по сети. Стаб сервера эмулирует на машине–сервере программу вызывающую указанную процедуру. Стаб сервера помещает в стек сервера параметры запуска и вызывает данную процедуру. После завершения ее выполнения стаб сервера получает

результаты выполнения процедуры и передает их по сети на стаб клиента. Стаб клиента помещает результаты в стек клиентской машины. Таким образом, с точки зрения процесса клиента, вызов процедуры осуществляется полностью аналогично вызову локальной процедуры.

Динамическое связывание

Важным вопросом организации удаленного вызова процедур является определение адреса сервера. Использование фиксированных (заданных) адресов приводит к снижению гибкости программы и проблемам при расширении масштабов сети.

Альтернативой в этом способе может быть динамическое определение адреса сервера и настройка стаб–процедур. Этот метод получил название – динамическое связывание, он заключающийся в импорте/экспорте интерфейсов, обладает высокой гибкостью. В начале работы сервер регистрируется в сети, посылая всем компьютерам сой адрес. Любой компьютер сети может обратиться на сервер с запросом импорта интерфейса. Сервер передает на клиентскую машину информацию о поддерживаемых функциях и интерфейс с ними. От компьютера клиента он получает аналогичный интерфейс для передачи данных.

Такой подход имеет большую гибкость, позволяет динамически переключаться между несколькими серверами, поддерживающими одинаковые функции для выбора оптимального режима работы. В рамках этого метода становится возможным периодический опрос серверов, анализ их работоспособности и, в случае отказа, автоматическое отключение, что повышает общую отказоустойчивость системы. Кроме того, пи таком подходе представляется возможным обеспечения различных прав клиентам по доступу к серверам за сет генерирования различных стаб–процедур.

Действия в случае отказов в обслуживании удаленного вызова процедур

Серьезную опасность при удаленном вызове процедур представляет отказ в обслуживании. Причины его могут быть различные, сбой сервера, сбой в машине клиента, переполнение очередей, времени ожидания разрыв связи. Такая ситуация приводит к тому что один из участников обмена данными ждет истечения предельного срока.

При анализе данной ситуации рассматриваются следующие классы отказов:

1. Клиент не может определить местонахождения сервера.
2. Потерян запрос от клиента к серверу. В этом случае обычно через установленное время повторяют запрос.
3. Потеряно ответное сообщение от сервера клиенту.
4. Сбой в работе сервера после получения запроса. В этом случае возможны следующие решения:

- Ждать до тех пор, пока сервер не перезагрузиться и пытаться выполнить операцию снова.
- Сразу сообщить приложению об ошибке сервере, с целью избежать повторных вызовов процедуры пока недоступной для выполнения.
- В случае сбоя сервера, не пересылать ни каких сообщений клиенту.

Если клиент потерпел аварию после отсылки запроса, то выполняются вычисления результатов, которых никто не ожидает. Такие вычисления называют «сиротами». Наличие сирот может вызвать различные проблемы: непроизводительные затраты процессорного времени, блокирование ресурсов, подмена ответа на текущий запрос ответом на запрос, который был выдан клиентской машиной еще до перезапуска системы.

В случае возникновения сирот возможны следующие решения:

- *Уничтожение.* До того, как клиентский стаб посылает запрос на запуск удаленной процедуры, он делает отметку в журнале. После перезагрузки сервера по записям в журнале, возможно, ликвидировать образовавшиеся процессы сироты.
- *Перевоплощение*. В этом случае все проблемы решаются без использования записи на диск. Метод состоит в делении времени на последовательно пронумерованные периоды. Когда клиент перезагружается, он передает сообщение всем машинам о начале нового периода. После приема этого сообщения все старые удаленные вычисления ликвидируются.
- *Мягкое перевоплощение* аналогично предыдущему случаю, за исключением того, что отыскиваются и уничтожаются не все удаленные вычисления, а только вычисления перезагружающегося клиента.
- *Истечение срока*. Каждому запросу отводится стандартный отрезок времени T, в течение которого он должен быть выполнен. После сбоя клиента сервер ждет в течение интервала T и уничтожает процессы сироты.

Синхронизация в распределенных системах

Обмен данными и совместное использование разделяемых ресурсов при удаленном вызове процедур остро ставят проблему синхронизации процессов.

Одним из способов решения данного вопроса – синхронизация часов всех компьютеров.

Алгоритм синхронизации часов (алгоритм Лампорта)

В распределенных системах часто нет необходимости в точном физическом соответствии значения часов реальному времени, важно обеспечить правильный порядок событий. Данная особенность привела к появлению логических часов.

Идея алгоритма синхронизации часов (алгоритма Лампорта) приведена на рис. 5.2.

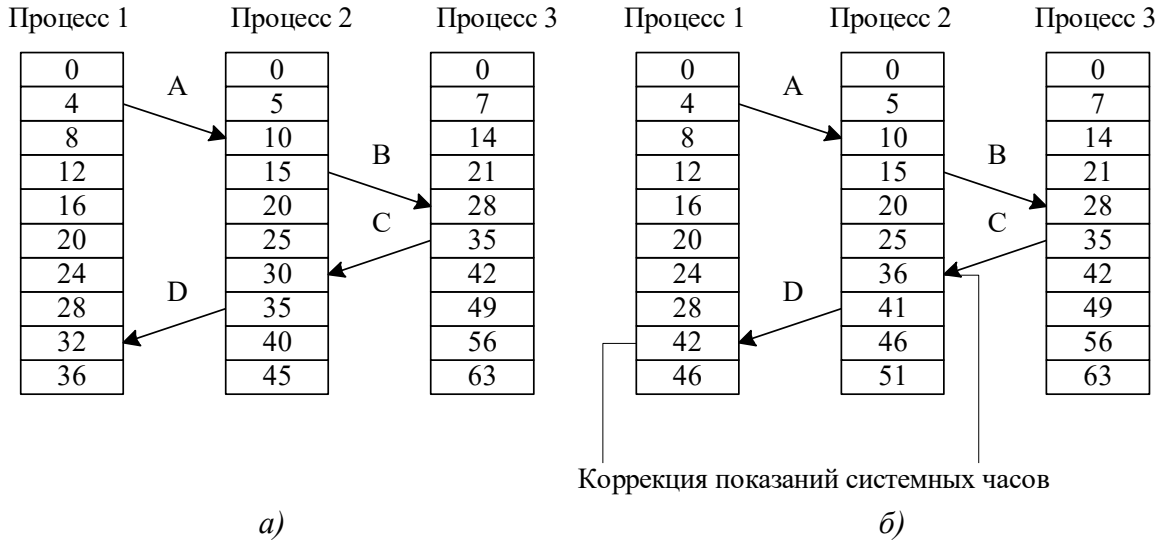


Рис. 5.2. Иллюстрация алгоритма синхронизации времени

Алгоритм синхронизации времени основан на фиксации времени отправки и получения сообщения. На рис. 5.2.а показан случай обмена сообщениями между тремя процессами на компьютерах имеющих расхождение в системных часах. На компьютере, выполняющим процесс 1 часы работаю немного быстрее, а на компьютере выполняющим процесс 3, чуть медленнее, чем системе часы на компьютере обслуживающим процесс 2. В итоге сообщение A, посланное с компьютера 1 в момент времени $t=4$, будет получено в момент времени $t=10$, по часам компьютера 2. Сообщение B, посланное в момент времени $t=28$, по часам компьютера 3. Данные задержки волне могут иметь место и с точки зрения скорости передачи информации по сети, по этому не нарушают работу распределенных компонент расчета. Однако при посылке сообщений C и D ситуация другая. Сообщение C, посланное в момент времени $t=35$ по часам компьютера 3, будет принято в момент $t=30$ по часам компьютера 2. Аналогично, сообщение A, посланное в момент времени $t=35$ по часам компьютера 2, будет принято в момент $t=32$ по часам компьютера 1. Подобная ситуация может привести к непоправимым сбоям в работе удаленных процедур.

Работа алгоритма синхронизации времени показана на рис. 5.2.б, она не затрагивает посылку сообщений A и B, так как значение момента времени приема сообщения по часам получателя больше значения времени отправки сообщения по часам отправителя $t^{j}_{\text{отпр.}} < t^{k}_{\text{получ.}}$. Однако, если данное соотношение не выполняется ($t^{j}_{\text{отпр.}} \geq t^{k}_{\text{получ.}}$), то производится коррекция системных часов получателя: $t^{k}_{\text{получ.}} = t^{j}_{\text{отпр.}} + 1$. Работу этого алгоритма не трудно проследить на рис. 5.2б. Сообщение C, отправленное

в момент времени $t = 35$ по часам компьютера 3, и принятое в момент времени $t = 30$ вызывает коррекцию системных часов второго компьютера, на них будет установлено значение $t = 36$.

Кроме алгоритма Лампорта, существуют еще ряд алгоритмов синхронизации времени которые можно найти в [].

6. Операционная система Novell NetWare

Сетевые операционные системы фирмы Novell являлись широко распространенными и занимали значительную часть рынка сетевых операционных систем до конца 90-х годов. Первые версии сетевых операционных систем Novell проявились в 1983 г.

Операционная система Novell является системой с выделенным высокоэффективным сервером, ориентированным на обеспечение скорости удаленного доступа к файлам и обеспечение повышенного уровня безопасности данных.

Операционную систему Novell NetWare отличает наличие единого для всех файл-серверов сетевого каталога (справочника сетевых ресурсов) – NetWare Directory Services (NDS), имеющего иерархическую древовидную структуру и основанного на международном стандарте X.500.

В Novell NetWare все сетевые ресурсы: файлы, принтеры, прикладные программы и так далее, составляют единую логическую структуру, не зависящую от их физического размещения, к которой пользователю достаточно подключится один раз.

Система имеет многоуровневую систему защиты каталогов и файлов, а также осуществляют контроль доступа пользователей к сети, не используется служебное имя Supervisor, что затрудняет поиск логического имени администратора сети. Кроме того, передача пароля по сети, основана на разделении ключей. Кроме того, в NetWare 4.0x используется новая технология, при входе пользователя в сеть сервер направляет рабочей станции запрос на идентификацию, зашифрованный с помощью пароля пользователя, случайного ключа и личного ключа пользователя. Рабочая станция расшифровывает этот запрос, используя случайный ключ и пароль, и получает значение личного ключа пользователя, который в дальнейшем используется при доступе ко всем сетевым ресурсам. Таким образом, ни личный ключ пользователя, ни пароль не передаются в явном виде по сети, что исключает возможность их перехвата и подделки. Система позволяет контролировать изменения в NDS и файловой системе.

Концепция построения Novell NetWare

Плоская модель памяти

Операционная система Novell NetWare работает в защищенном режиме (protected mode) с 32-разрядной адресацией памяти.

В защищенном режиме память адресуется непрерывным диапазоном адресов, так называемая "плоская" (flat) модель памяти. В этом случае нет

необходимости переключать сегменты памяти, так как вся память состоит из одного сегмента. Плоская модель не требует переключения сегментов в процессе работы, что значительно повышает скорость работы.

Нити и невытесняющая многозадачность

Защищенный режим позволяет выполнять несколько процессов одновременно. В Novell NetWare реализован механизм «нитей» (thread), который позволяет использовать расщепление одного процесса на несколько, параллельно выполняемых, нитей. При этом используется метод невытесняющей многозадачности, не допускающий прерывания нити другими процессами или нитями.

При разработке приложений под Novell NetWare важно понимать последствия монопольного захвата ресурсов процессора. Все нити должны регулярно возвращать управление операционной системе, переключение нитей возможно только в эти моменты.

Кэширование диска

Операционная система Novell NetWare работает только в качестве файл сервера, не предназначенного для работы на нем пользователя, вся оперативная память, оставшаяся после загрузки операционной системы, драйверов и приложений направляется на кэширование диска. Таким способом достигается высокая скорость доступа клиентам к файловой структуре сервера.

Дополнительным преимуществом операционной системы Novell NetWare, направленным на повышение производительности, является наличие элеваторного и параллельного поиска.

Структура операционной системы Novell NetWare

Структура операционной системы Novell приведена на рис. 6.1.

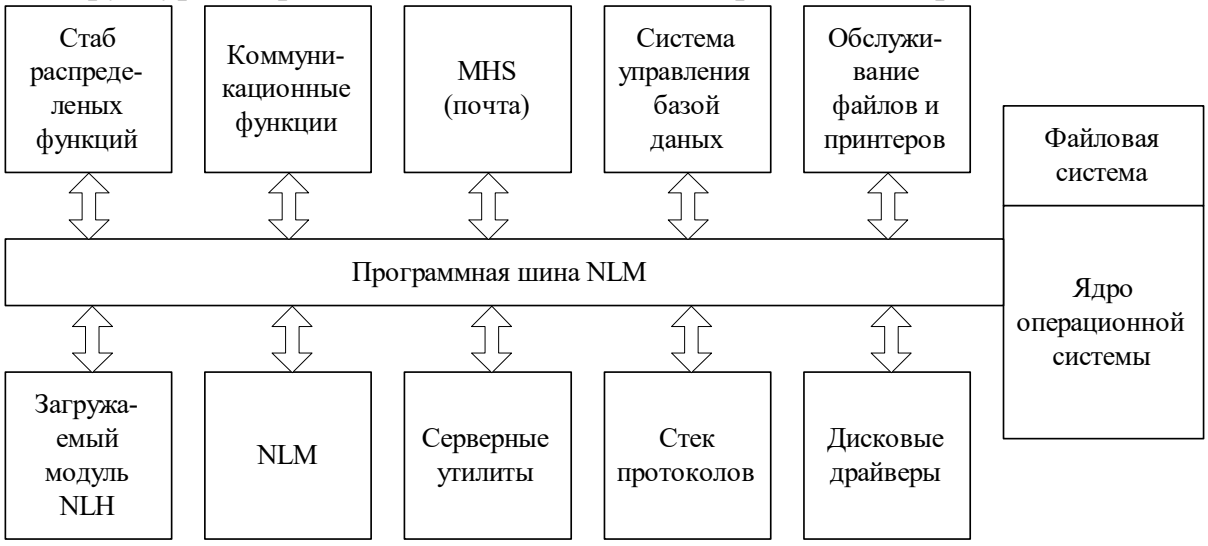


Рис. 6.1. Структура операционной системы Novell NetWare

Все сетевые сервисы, утилиты сервера, серверные приложения выполнены в Novell NetWare в виде загружаемых модулей – NLM, которые могут динамически загружаться и выгружаться в любое время без остановки сервера.

Ядро системы выполняет функции по управлению памяти, планирование и диспетчеризацию нитей, обслуживание файловой системы. Обмен данными осуществляется через программно–реализованную шину интерфейса NLM–ов. Каждый из NLM модулей выполняет определенные функции. Настройка сервера осуществляется путем загрузки необходимых файлов NLM.

Программный интерфейс модулей NLM является открытым и документированным, что позволяет программисту встраивать необходимые новые функции прямо в ядро операционной системы.

Novell NetWare поддерживает все наиболее популярные протоколы, такие как: TCP/IP, Apple Talk и другие. Для своих внутренних целей операционная система Novell NetWare использует протокол IPX/SPX. Кроме того, операционная система поддерживает большое количество программных шлюзов к другим широко распространенным сетям, таким как сети Internet и SNA.

Защита информации

Обеспечение целостности информации

Операционная система Novell NetWare обеспечивает следующие функции обеспечения целостности информации:

- Проверка записи. После записи на диск каждый блок данных читается в память. Записываемая на диск информация сохраняется до завершения проверки записи.
- Дублирование каталогов. Novell NetWare хранит копию структуры корневого каталога.
- Дублирование таблицы размещения файлов FAT.
- Оперативное исправление 1 (Hot Fix 1) – запись данных из дефектных блоков в специальную резервную область диска.
- Контроль UPS.
- Зеркальное отображение дисков, подключенных к одному дисковому контроллеру (Disk Mirroring).
- Дуплексирование дисков, подключенных к различным дисковым контроллерам (Disk Duplexing).
- Оперативное исправление 2 (Hot Fix 2). Восстановление данных из сбойных секторов путем переписывания с зеркалированного или дуплексированного дисков в резервную область.
- Система отслеживания транзакций (TTS).
- Динамическое зеркалирование двух серверов, находящихся на значительном удалении друг от друга.

Обеспечение защиты информации

Средства обеспечения информационной безопасности встроены в ядро операционной системы и обеспечивают следующие уровни защиты:

- защита информации о пользователе;
- защита паролем;
- защита каталогов;
- защита файлов;
- межсетевая защита.

Пароли и другая служебная информация о пользователях хранится и передается по сети в зашифрованном виде.

Планирование процессов

Схема планирования процессов в операционной системе Novell NetWare приведена на рис. 6.2.

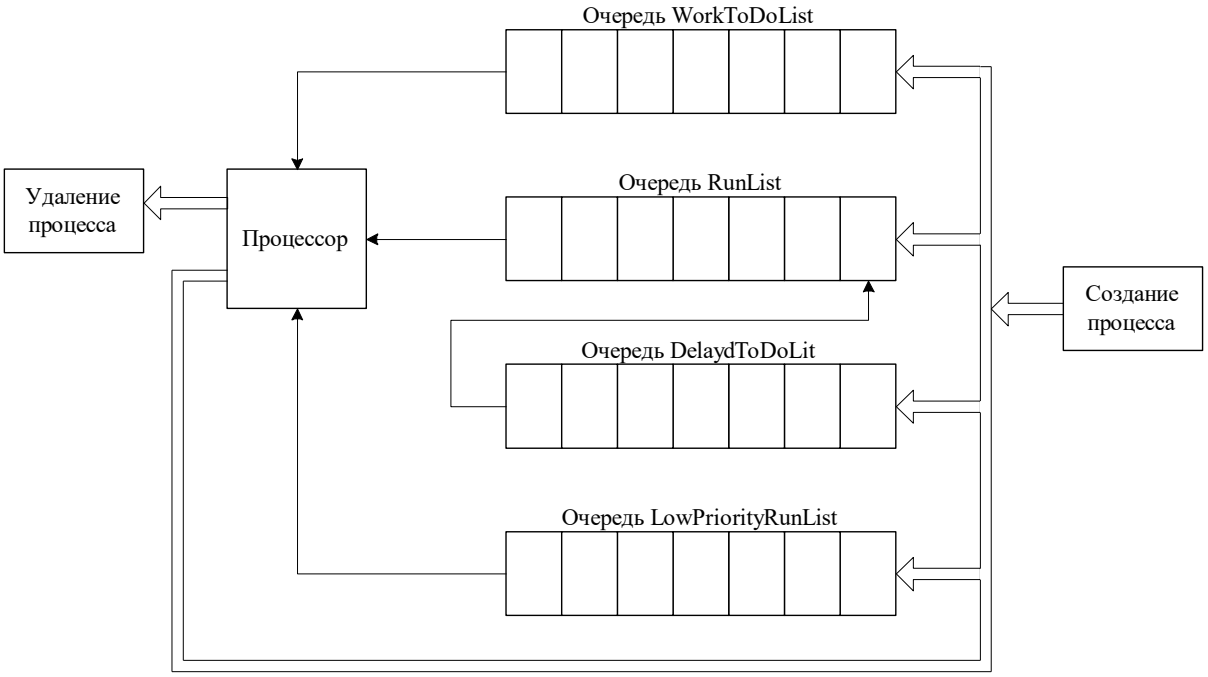


Рис. 6.2 Структурная схема планирования процессов в операционной системе Novell NetWare

Ядро операционной системы Novell NetWare поддерживает четыре очереди процессов, в которые помещаются, как новые (создаваемые) так и приостановленные процессы. При этом используется не вытесняющая многозадачность.

Очередь WorkToDoList предназначена для самых приоритетных процессов. Выполнение процессов из остальных очередей возможно, только если очередь WorkToDoList пуста.

Очередь RunList содержит готовые к выполнению процессы с обычным уровнем приоритета.

В очередь DelaydToDoList помещаются процессы, находящиеся в режиме ожидания. После завершения условий ожидания процесс помещается в конец очереди RunList.

Очередь LowPriorityRunList предназначена для процессов с самым низким приоритетом. Процессы из этой очереди могут попасть на выполнение только в случае если в очередях WorkToDoList и RunList не ни одного процесса готового к выполнению. Обычно в эту очередь попадают процессы обслуживающие фоновые задачи.

Для создания процессов операционная система Novell NetWare использует несколько функций. Вызов соответствующей функции определяет, в какую из очередей попадет создаваемый процесс.

Файловая система

Файловая система операционной системы Novell NetWare отличается от файловых систем обычных операционных систем, рассчитанных на обслуживания пользователя.

Том – первичная структура данных файловой системы. Том содержит логическую информацию о файлах, пространство имен для поддержки не имен не формата MS DOS, собственные системы исправления ошибок и отслеживания транзакций, а так же физическое место для размещения файлов.

Сервер Novell NetWare может поддерживать до 64 томов, каждый из которых может иметь объем до 32 ТБ.

На физическом уровне том состоит из блоков по 4 КБ (8 непрерывных секторов). Том может содержать до 32 сегментов, каждый из которых является отдельным диском.

Логическая структура диска включает в себя таблицы FAT и DET. Таблица FAT аналогична таблицы FAT MS DOS. Таблица DET аналогична структуре корневого каталога операционной системы MS DOS, однако с учетом пространства имен не формата MS DOS, допускает множественное вхождение в файл.

Команды и утилиты для работы с сервером

Команда смены контекста CX. Команда CX позволяет перемещаться по дереву. Запуск команды CX без параметров выводит Ваше текущее положение. Для перехода на вышестоящий уровень необходимо вести команду «CX .». Для отображения дерева Novell необходимо подать команду CX /T.

Команда MAP. Команда MAP позволяет назначить новый привод, например: MAP X: =SERV1/SYS: USERS\BOX. В приведенном примере: SERV1 – имя сервера, SYS – имя тома, USERS\BOX – путь до каталога, представляющего пользователю в виде диска X: .

Команда NCOPY. Команда NCOPY аналогична команде COPY в MS DOS, но предназначена специально для работы с серверными дисками.

Команда NLIST. Данная команда позволяет получать списки объектов определенного класса. Например: NLIST USER выведет список пользователей.

Монтирование тома. Команда MOUNT позволяет смонтировать том сервера. Пример: MOUNT TOOLS.

Размонтирование тома. Размонтирование тома производится командой DISMOUNT. Команды MOUNT и DISMOUNT доступны для выполнения только с консоли сервера.

Команда SET. Команда SET это основная команда конфигурирования сервера. Возможности данной команды очень обширны, операционная система Novell NetWare содержит целый NLM модуль по обработки только команды SET. Информацию о работе с этой командой можно найти в [].

Разрешения работы с файлами не в формате MS DOS. Для разрешения работы с файлами не в формате MS DOS необходимо воспользоваться командой ADD NAME SPACE. Например, разрешение работать на томе USERS с файлами в формате Windows: ADD NAME SPACE WINDOWS to volume USERS. Предварительно необходимо загрузить NML модуль для поддержки файловой системы указанной операционной системы, в данном случае WINDOWS.NML.

Внимание, включение файлового пространства выполняется только один раз.

Запуск NLM модулей. Запуск и завершение работы NML модулей осуществляется с помощью команд LOAD и UNLOAD. Например: LOAD NE2000 PORT=300.

Завершение работы сервера

Для завершения работы сервера необходимо подать команды DOWN (остановка сервера) и после ее завершения команду EXIT (выход из сетевой операционной системы Novell NetWare).

Сценарии входа

Сценарий входа это аналог файла autoexec.bat, но написанный для конкретного пользователя. Файл сценария запускается после загрузки локального компьютера и позволяет заданным образом настроить сетевое окружение пользователя.

Утилита NwAdmin

Утилита NwAdmin представляет собой удобную графическую утилиту для создания пользователей и задания им необходимых прав. С помощью утилиты NwAdmin можно задать следующие ограничения на подключения пользователя: на сетевой адрес, объем тома, время входа. Подробнее с работой утилиты NwAdmin можно ознакомится в методических указаниях к лабораторным работам по данной дисциплине.

Удаленное администрирование сервера

В операционной системе Novell NetWare имеется утилита для удаленного администрирования сервера RCONSOLE. Для того, что бы сервер был доступен для удаленного администрирования, на нем должен быть загружен модуль REMOTE.NML.

Основные направления развития операционной системы Novell NetWare

Стратегия распределенной параллельной обработки

Структура параллельной обработки информации в операционной системе Novell NetWare приведено на рис. 6.3.

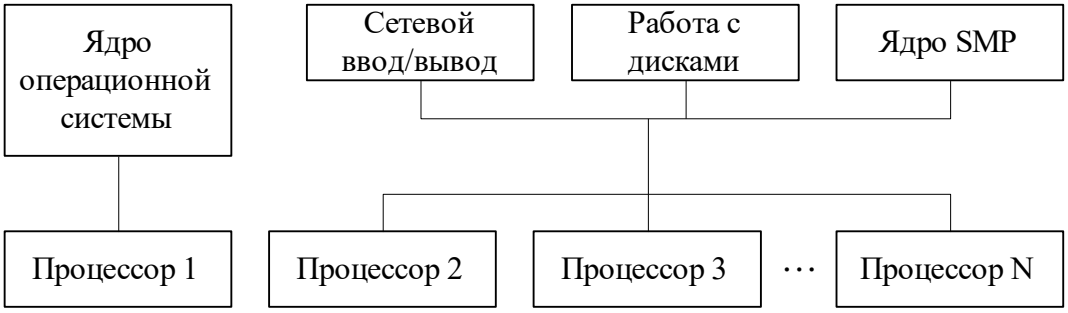


Рис. 6.3. Схема распараллеливания процессов в последних версиях Novell NetWare

В приведенной схеме один процессор выделяется на обслуживание ядра операционной системы и существующего набора модулей NLM. Остальные процессоры выделяются на обслуживания работы с сетевыми средствами ввода/вывода, физическими дисками и SMP – ядро параллельной обработки специальных загружаемых модулей, предназначенных для параллельного выполнения (NLM SPM).

Обеспечение процессорной независимости

Операционная система Novell NetWare запускается поверх базовой операционной системы, что делает возможным перенос ее с машины на машину простым копированием.

Кроме того, в настоящее время ядро операционной системы избавлено от команд x86, архитектуры памяти и шин PC, системы прерываний. Данная особенность позволяет переносить систему без изменений настроек, модулей NLM и файловой структуры с одной аппаратной платформы на другую, заменяя только модуль, связанный аппаратным особенностями.

Шлюзы IP сетей

Расширение масштабов компьютерных сетей делает неизбежным обеспечение стыка сетей построенных на базе Novell NetWare с сетями, построенными на использовании протокола TCP/IP. Проблема обеспечения взаимодействия заключаются в использования различных протоколов верхнего уровня.

Для решения этой проблемы используется механизм, упаковывающий IPX пакеты в TCP/IP. Кроме того, в последние версии Novell NetWare вставлена непосредственная поддержка протокола IP для обеспечения отдельных локальных протоколов входящих в состав IPX (SLP, CSLIP и PPP).

7. Операционные системы Microsoft

Компания Microsoft выпустило целое семейство операционных систем. В этом пособии операционные системы Windows рассматриваются на примере операционной системы Windows NT, которая с самого начала проектировалась с учетом всех современных требований: расширяемости, переносимости, надежности, совместимости, производительности. Эти свойства были достигнуты на основе использования объектно–ориентированного проектирования, клиент-сервер архитектуры, микроядерного принципа построения.

В отличие от предыдущих версий Windows, в которой была реализована многозадачность без вытеснения, в Windows NT и далее используется механизм многозадачности с вытеснением.

Для управления нитями Windows NT использует механизм приоритетов. Многопроцессорная организация вычислительного процесса позволяет выполнять нити, с учетом приоритетов, на различных процессорах одновременно, разделяя память между ними. Операционная система поддерживает до 16 процессоров.

Структура операционной системы Windows, на примере Windows NT показана на рис. 7.1.

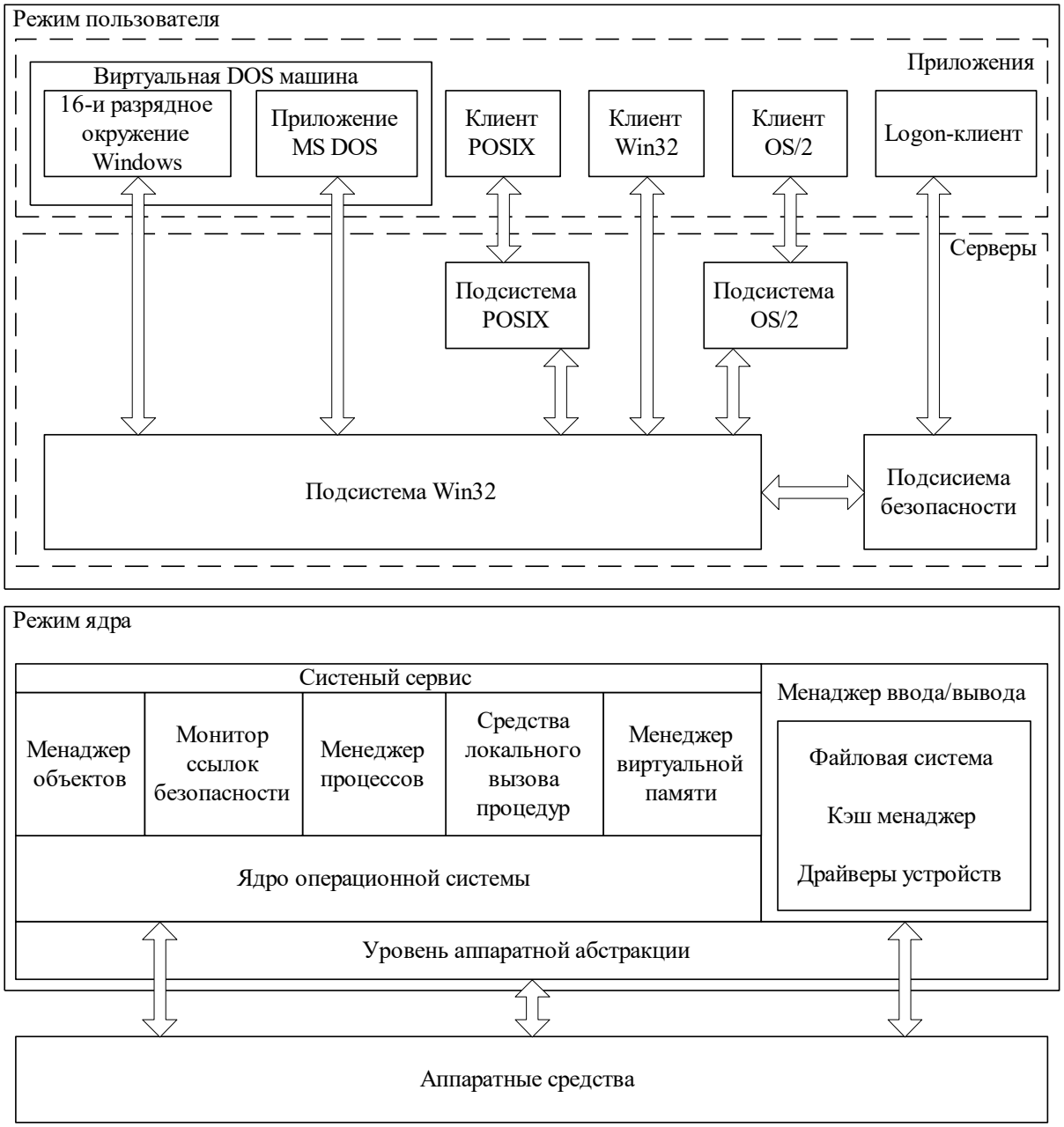


Рис. 7.1 Структура операционной системы Windows

Исполнительная часть Windows NT работает в режиме ядра. Она обеспечивает функционирование виртуальной памяти, подсистемами ввода/вывода, файловой системы и другим процессами и объектами, в том числе и подсистемы безопасности. Все компоненты взаимодействуют между собой с помощью межмодульной связи. Вызовы организованы в виде набора внутренних процедур. Такой подход позволяет легко наращивать возможности операционной системы за счет замены и добавление новых функций.

Пользовательскую часть составляют серверы – защищенными подсистемами, выполняющие в отдельном процессе, с собственной памятью отделенной от других процессов. Серверы общаются друг с другом и с клиентами посредством посылки сообщений. Все сообщения

проходят через ядро системы. Защищенные подсистемы работают в пользовательском режиме, но создаются в процессе загрузки операционной системы.

Наиболее важной подсистемой окружения является Win32, которая обеспечивает доступ для приложений к 32-bit Windows API. Кроме того, эта система обеспечивает графический интерфейс пользователя, управляет пользовательским вводом/выводом данных пользователя, через нее обеспечивается поддержка подсистем POSIX, OS/2, а так же 16–и разрядной Windows и MS-DOS, входящие в состав виртуальной DOS машины.

На рис. 7.2 показан граф переходов алгоритма планирования нитей реализующий вытесняющую многозадачность и систему приоритетов.

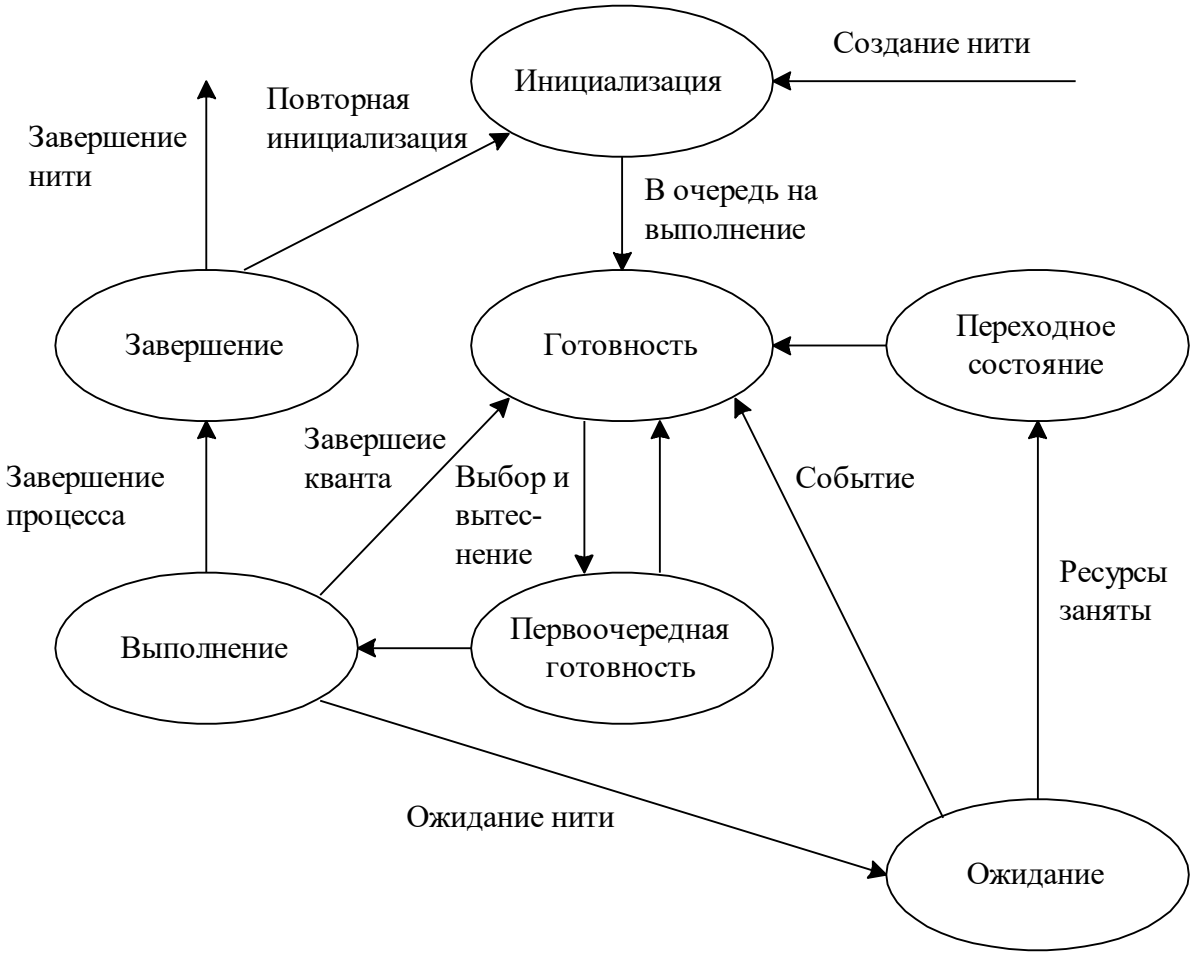


Рис. 7.2. Граф переходов состояний в операционной системе Windows NT

Особенностью приведенного на рис. 7.2 алгоритма является то, что если квант времени отведенного на выполнение нити завершен, или в списке готовых на выполнение нитей, появилась новая нить с более высоким приоритетом, то происходит переключение процессора на выполнение новой нити. Таким образом, ни одна нить не может монополизировать ресурсы процессора. Приоритеты нитей устанавливаются следующим образом – нити реального времени имеют

значение приоритета от 16 до 31, приоритеты обычных нитей имеют значение от 0 до 15.

В рассмотренной системе, нить может находиться в одном из семи состояний:

- Инициализация. Менеджер процессов обращается к ядру операционной системы за выделением соответствующих ресурсов.
- Готовность. В состоянии готовности находятся полностью готовые к выполнению и имеющие все необходимые для работы ресурсы.
- Первоочередная готовность. В состоянии первоочередной готовности находится нить, предназначенная для выполнения следующей. Если в списке готовых нитей появляется нить с большим значением приоритета, то она вытеснит нить из состояния первичная готовность.
- Выполнение. В состоянии выполнения нить находится до завершения, или до вытеснения ее нитью с более высоким приоритетом, или до завершения отведенного на ее работу кванта времени.
- Ожидание. В состоянии ожидания помещаются нити, ожидающие необходимые им ресурсы компьютера или отведенного им момента выполнения.
- Переходное состояние. В переходном состоянии находятся готовые к выполнению нити, ожидающие необходимого ресурса, например загрузки сегмента виртуальной памяти с диска. При этом надо отметить, что это ожидание более краткое, чем для нитей из очереди ожидание.
- Завершение. После завершения нить может быть удалена, а выделенные для ее работы ресурсы освобождены, или она может быть возвращена на повторную инициализацию.

Сетевые средства операционной системы Windows NT имеют следующую структуру, приведенную на рис. 7.3.

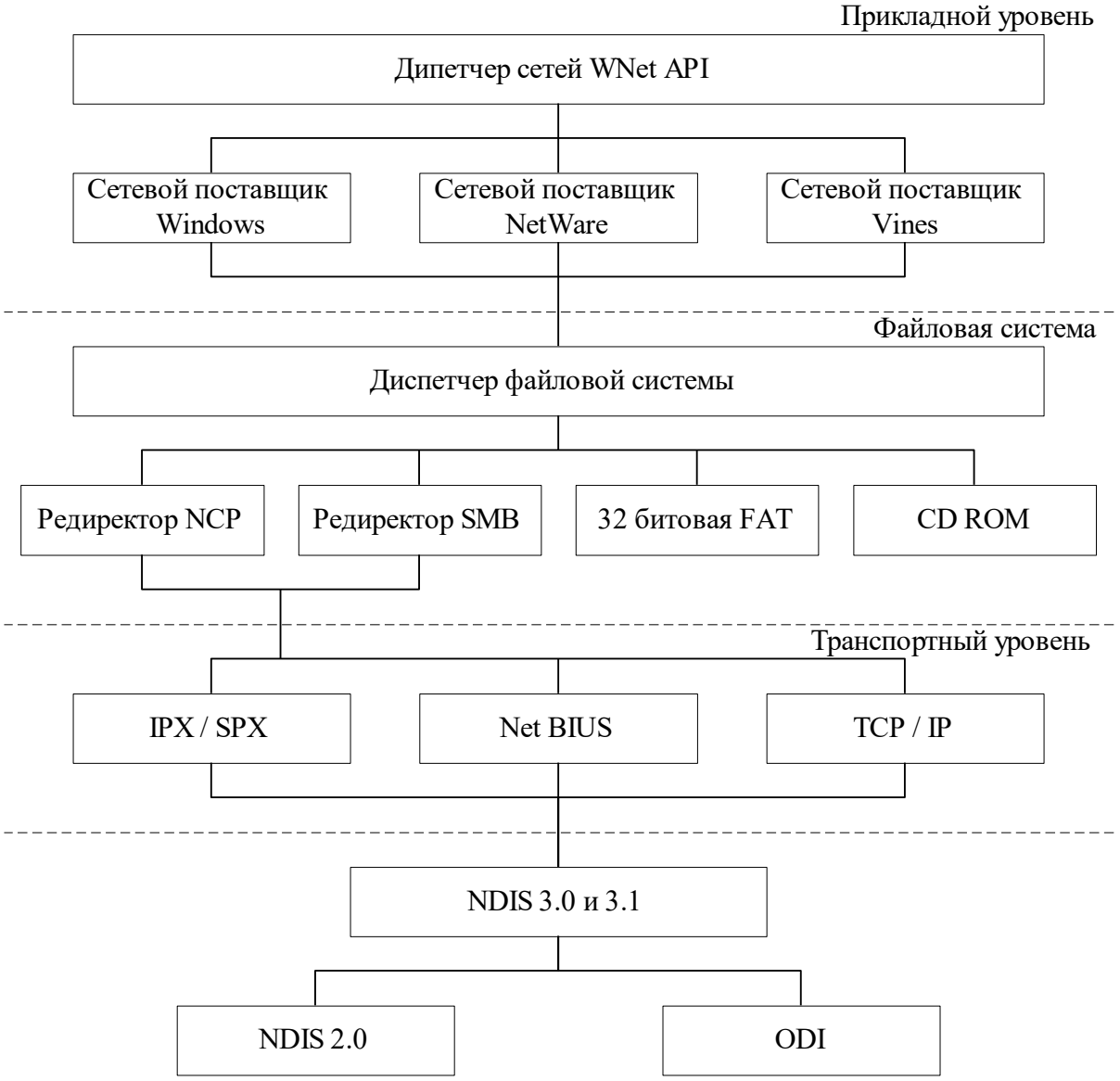


Рис. 7.3. Структура сетевых средств операционной системы Windows NT

Операционная система поддерживает программный интерфейс WNet, с помощью которого приложения и системные утилиты получают доступ к просмотру, отображению и потреблению сетевых ресурсов, работающий на прикладном уровне. На уровне файловой системы, диспетчер инсталлируемой файловой системы обслуживает редиректоры блоков SMB и NCP наравне с файловой системой FAT32. На транспортном уровне поддерживаются различные протоколы, которые взаимодействуют с канальным уровнем сетевых средств.

8. Операционная система UNIX

Концепция построения операционной системы UNIX

В настоящее время существует целое семейство операционных систем UNIX. При изложении особенностей построения в данной работе автор ориентировался на версию UNIX System V Release 4.

Характерными особенностями всех операционных систем UNIX является:

- многопользовательский режим;
- средства защиты данных от несанкционированного доступа,
- реализация мультипрограммной обработки в режиме разделения времени, основанная на использовании алгоритмов вытесняющей многозадачности;
- использование механизмов виртуальной памяти и свопинга;
- унификация операций ввода–вывода на основе расширенного использования понятия "файл";
- иерархическая файловая система, образующая единое дерево каталогов независимо от физической организации устройств;
- переносимость системы, за счет написания ее основной части на языке С;
- возможности организации взаимодействия процессов, в том числе и через сеть,
- кэширование диска для уменьшения времени доступа к файлам.

В основе UNIX лежит понятие процесс, каждый процесс работает в своем виртуальном адресном пространстве.

При управлении процессами операционная система использует два основных типа информационных структур: дескриптор процесса и контекст процесса. Дескрипторы отдельных процессов образуют таблицу процессов.

В дескрипторе содержится информация о состоянии процесса, расположении образа процесса в виртуальной памяти, о значении приоритетов, идентификатор пользователя, создавшего процесс, информация о родственных процессах, о событиях, осуществления которых ожидает данный процесс и некоторая другая информация.

Контекст процесса содержит менее оперативную, информацию о процессе, необходимую для возобновления выполнения процесса с прерванного места: содержимое регистров процессора, коды ошибок выполняемых процессором системных вызовов, информацию об открытых файлов и незавершенных операциях ввода/вывода. Контекст и дескриптор процесса доступен только программам ядра.

В UNIX для процессов предусмотрены два режима выполнения: привилегированный режим - "система" и обычный режим - "пользователь". В режиме "пользователь" запрещено выполнение действий, связанных с управлением ресурсами системы, в частности, корректировка системных

таблиц, управление внешними устройствами, маскирование прерываний, обработка прерываний. В режиме "система" выполняются программы ядра, а в режиме "пользователь" - оболочка и прикладные программы. При необходимости выполнить привилегированные действия пользовательский процесс обращается с запросом к ядру в форме так называемого системного вызова. В результате системного вызова управление передается соответствующей программе ядра. Процесс, работающий в режиме системы, не мог быть вытеснен другим процессом.

Операционная система реализует вытесняющую многозадачность, основанную на использовании приоритетов и квантования по времени.

Глобальное значение приоритета имеют следующие значения:

- Процессы реального времени (значения приоритетов 100..159). Процессы реального времени используют стратегию фиксированных приоритетов, конкретное значение которого устанавливается пользователем. Процессы работают, весь отведенный квант времени.
- Системные процессы (значения приоритетов 60..99). Системные процессы это внутренние процессы ядра системы, значение приоритета устанавливается операционной системой и не меняется в процессе жизни процесса.
- Обычные процессы (значения приоритетов 0..59). Процессы с разделением времени используют динамическое значение приоритетов. Значение приоритета составляется из пользовательской и системной части. Пользовательская часть может изменяться пользователем в сторону уменьшения. Системная часть приоритета меняется в зависимости от загрузки процессора. Приоритет процессов, долго занимающих ресурсы процессора, снижается, при этом им устанавливаются большие значение кванта времени.

Файловая система операционной системы UNIX

В операционной системе реализован механизм виртуальной файловой системы (VFS), который позволяет ядру системы одновременно поддерживать несколько файловых систем. Механизм VFS поддерживает для ядра некоторое абстрактное представление о файловой системе, скрывая от него конкретные особенности каждой файловой системы.

Различаются следующие типы файлов:

- Обычные файлы – содержат любую информацию пользователя.
- Каталог – файл, содержащий служебную информацию о группе файлов (обычные, специальные файлы, подкаталоги), в него входящих.
- Специальный файл – файл, ассоциируемый с каким-либо устройством ввода/вывода, используется для унификации механизма доступа к файлам и внешним устройствам.

Файловая система UNIX имеет иерархическую структуру, в основе которой находится корневой каталог. Пример дерева каталогов приведен на рис. 8.1.

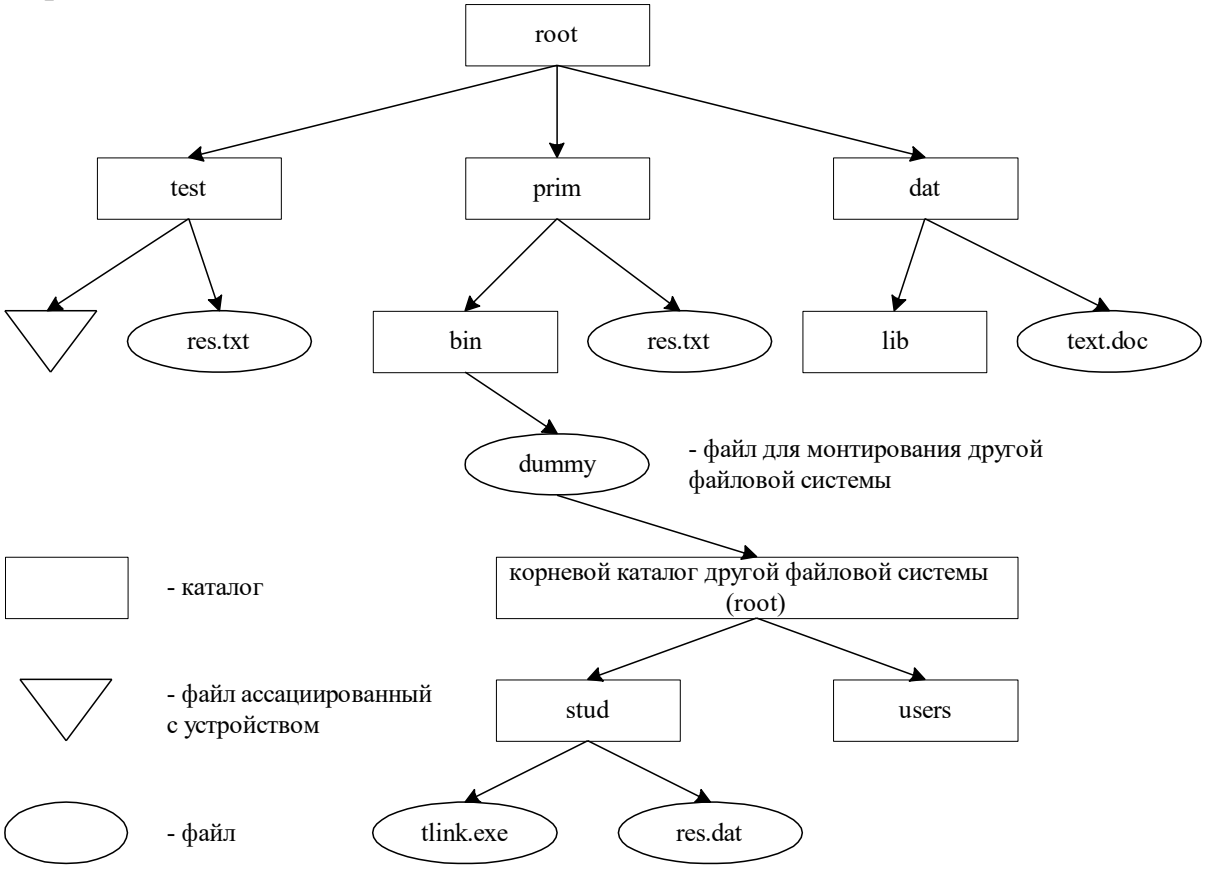


Рис. 8.1. Пример дерева каталогов файловой системы UNIX

На рис. 7.4, показана процедура монтирования другой файловой системы. Операция монтирования осуществляется с помощью системного вызова mount, после вызова каталог bin становится корневым каталогом подключаемой файловой системы.

Имена файлов. Имя файла, в месте с расширением файла, в UNIX может иметь длину до 14 символов. Расширение отделяется от имени файла символом «.» (точка). В качестве разделителя между именем файла и именем каталога, и между именами каталогов, используется символ «/».

Например, полное имя файла res.dat, из дерева, приведенного на рис. 8.1, будет выглядеть так: /prim/bin/stud/res.dat.

Физическая организация файла. Физическая организация файла приведена на рис. 8.2.

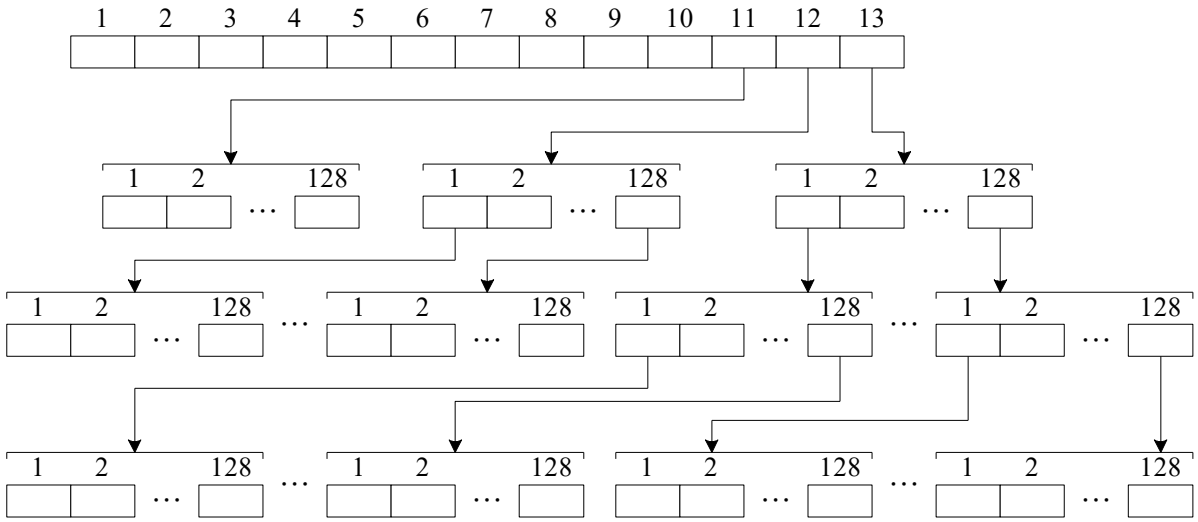


Рис. 8.2. Физическая организация файла в операционной системе UNIX

Файл располагается в блоках (возможно не смежных) дисковой памяти. Логическая последовательность блоков в файле задается набором из 13 элементов. Первые 10 элементов предназначаются для непосредственного указания номеров первых 10 блоков файла. Если размер файла превышает 10 блоков, то в 11 элементе указывается номер блока, в котором содержится список следующих 128 блоков файла. Если файл имеет размер более чем 10+128 блоков, то используется 12-й элемент для двухуровневой косвенной адресации, содержащий номер блока, в котором указываются номера 128 блоков, каждый из которых может содержать еще по 128 номеров блоков файла. Если файл больше, чем $10+128+128^2$ блоков, то используется 13 элемент для трехуровневой косвенной адресации. При таком способе адресации предельный размер файла составляет 2 113 674 блока. Традиционная файловая система операционной системы UNIX поддерживает размеры блоков 512, 1024 или 2048 байт

Индексные дескрипторы и каталоги. Информация о файле, кроме его символьного имени, хранится в специальной системной таблице, называемой индексным дескриптором файла, имеющего размер 64 байта. В индексный дескриптор файла входит информация о физическом расположении файла на диске, информация о размере файла, режимах доступа, датах создания, модификации, открытия. Номер индексного дескриптора является уникальным цифровым именем файла.

Каталог представляет собой совокупность записей обо всех файлах и каталогах, входящих в него. Каждая запись состоит из 16 байтов, 14 байтов отводится под короткое символьное имя файла или каталога, а 2 байта – под номер индексного дескриптора этого файла.

Расположение файловой системы на диске показано на рис. 7.6. Все дисковое пространство, отведенное под файловую систему, делится на четыре области:

- *загрузочный блок (boot),* в котором хранится загрузчик операционной системы;
- *суперблок (superblock)* – содержит самую общую информацию о файловой системе: размер файловой системы, размер области индексных дескрипторов, число индексных дескрипторов, список свободных блоков и список свободных индексных дескрипторов, а также другую административную информацию;
- *область индексных дескрипторов*, порядок расположения индексных дескрипторов в которой соответствует их номерам;
- *область данных*, в которой расположены как обычные файлы, так и файлы–каталоги. Специальные файлы представлены в файловой системе только записями в соответствующих каталогах и индексными дескрипторами специального формата, но места в области данных не занимают.

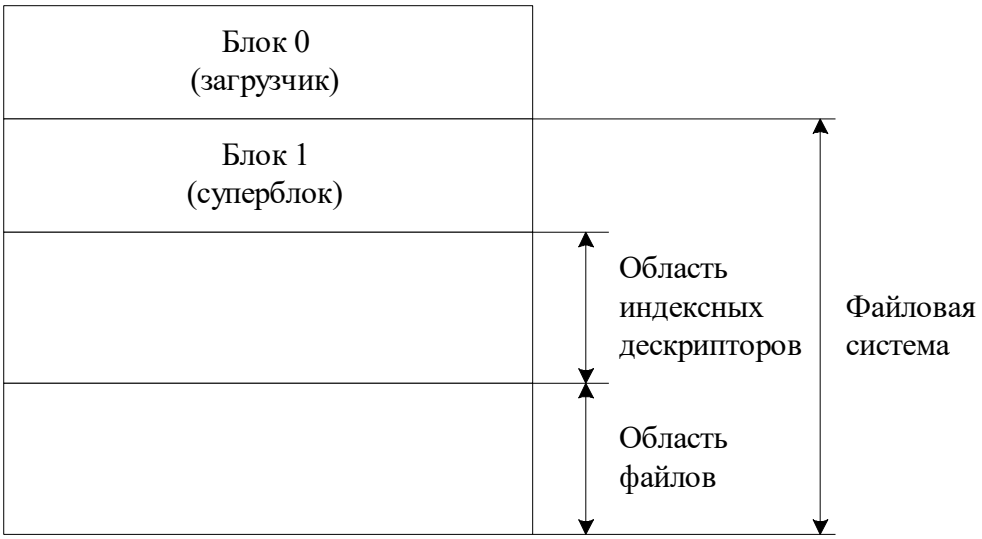


Рис. 8.3. Размещение физической файловой системы на диске

При открытии файла система выполняет следующие действия:

1. Проверяет, существует ли файл и разрешен ли к нему доступ. Если файл не существует, то можно ли его создать.
2. Копирует индексный дескриптор с диска в оперативную память.
3. Создает в области ядра структуру file, предназначенную для операций обмена данными с указанным файлом.
4. Делает отметку в контексте процесса, выдавшего системный вызов на операцию с данным файлом.

Виртуальная файловая система VFS

В UNIX System V Release 4 был реализован механизм переключения виртуальных файловых систем – Virtual File System (VFS), позволяющий операционной системе поддерживать различные файловые системы. Информация о файловой системе и файлах разбивается на две части – зависимую от типа файловой системы и не зависимую. Механизм VFS

транслирует запросы ядра в операции, зависящие от типа файловой системы, само ядро имеет данные только о независимой части файловой системы.

Виртуальная файловая система VFS поддерживает следующие типы файлов:

- обычные файлы,
- каталоги,
- специальные файлы,
- именованные конвейеры,
- символьные связи.

Именованные конвейеры – это средство обмена данными между процессами. Конвейер буферизует данные, поступающие на его вход, процесс, получающий данные на его выходе, получает их в порядке «первый пришел – первый вышел» (FIFO). Именованные конвейеры позволяют обмениваться данными произвольной паре процессов.

Символьная связь – это файл данных, содержащий имя файла (возможно даже еще не существующего), с которым предполагается установить связь. При создании символьной связи создается новый индексный дескриптор inode (каталог) и резервируется отдельный блок данных для хранения полного имени файла, на который он ссылается.

Механизм виртуальных файловых систем VFS реализуется в виде массива структур vfssw, каждая из которых описывает файловую систему конкретного типа, которая может быть установлена в системе. Структура vfssw состоит из четырех полей:

- символьного имени файловой системы;
- указателя на функцию инициализации файловой системы;
- указателя на структуру, описывающую функции, реализующие абстрактные операции VFS в данной конкретной файловой системе;
- флаги, которые не используются в описываемой версии UNIX.

Пример инициализированного массива структур vfssw:

```
struct vfssw vfssw[] = {
{0, 0 , 0 ,0 }, // нулевой элемент не используется
{"spec", specint, &spec_vfsops,  0},
{"dos", dosinit, &dos_vfsops,  0}.
```

Структура vfsops, описывающая операции, которые выполняются файловой системой, состоит из 7 полей:

VFS_MOUNT – монтирование файловой системы;
VFS_UNMOUNT – размонтирование файловой системы;
VFS_ROOT – получение vnode для корня файловой системы;
VFS_STATVFS – получение статистики файловой системы;
VFS_SYNC – выталкивание буферов файловой системы на диск;
VFS_VGET – получение vnode по номеру дескриптора файла;
VFS_MOUNTROOT – монтирование корневой файловой системы.

Структура монтирования нескольких файловых систем в единое дерево каталогов показана на рис. 8.4.

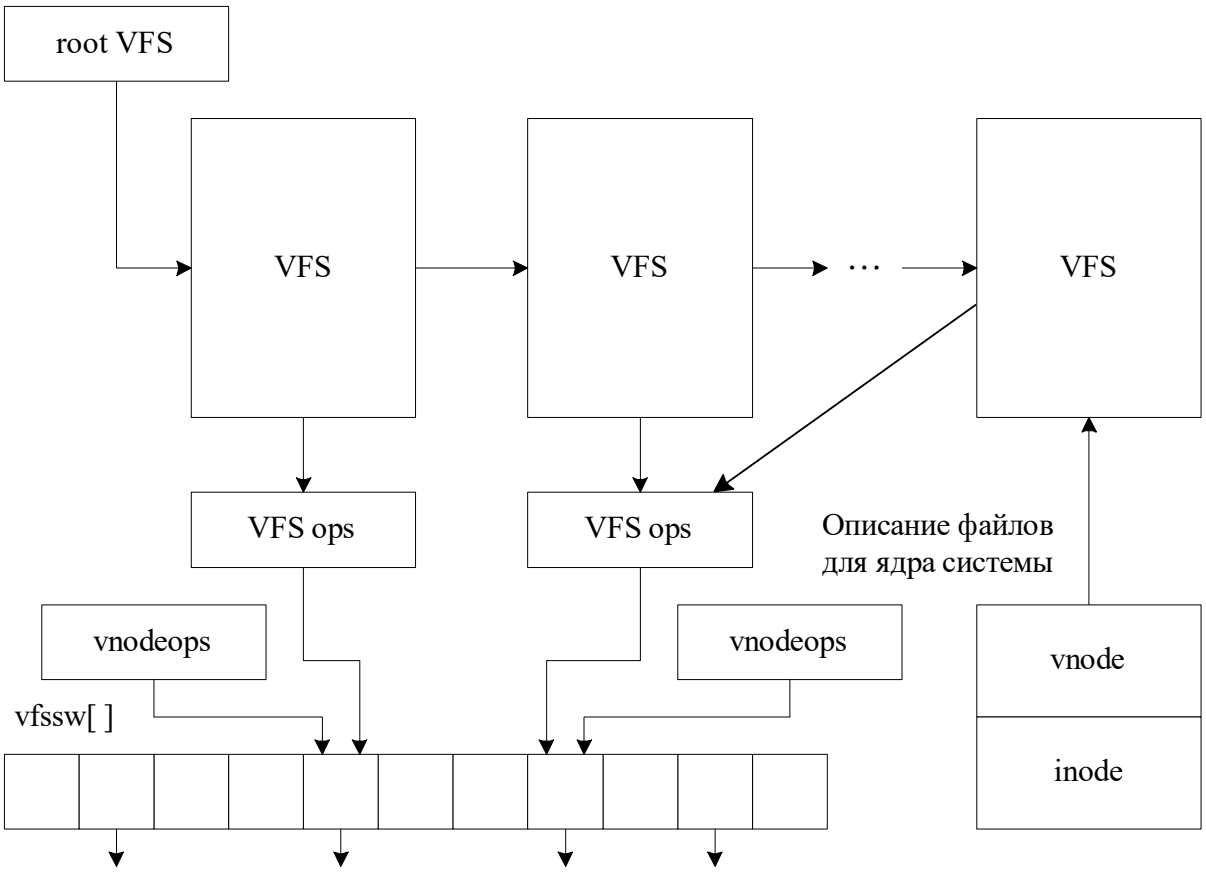


Рис. 8.4. Структура монтирования различных файловых систем при использовании виртуальной файловой системы

Работа ядра с файлами во многом основана на использовании структуры vnode (рис. 8.5).

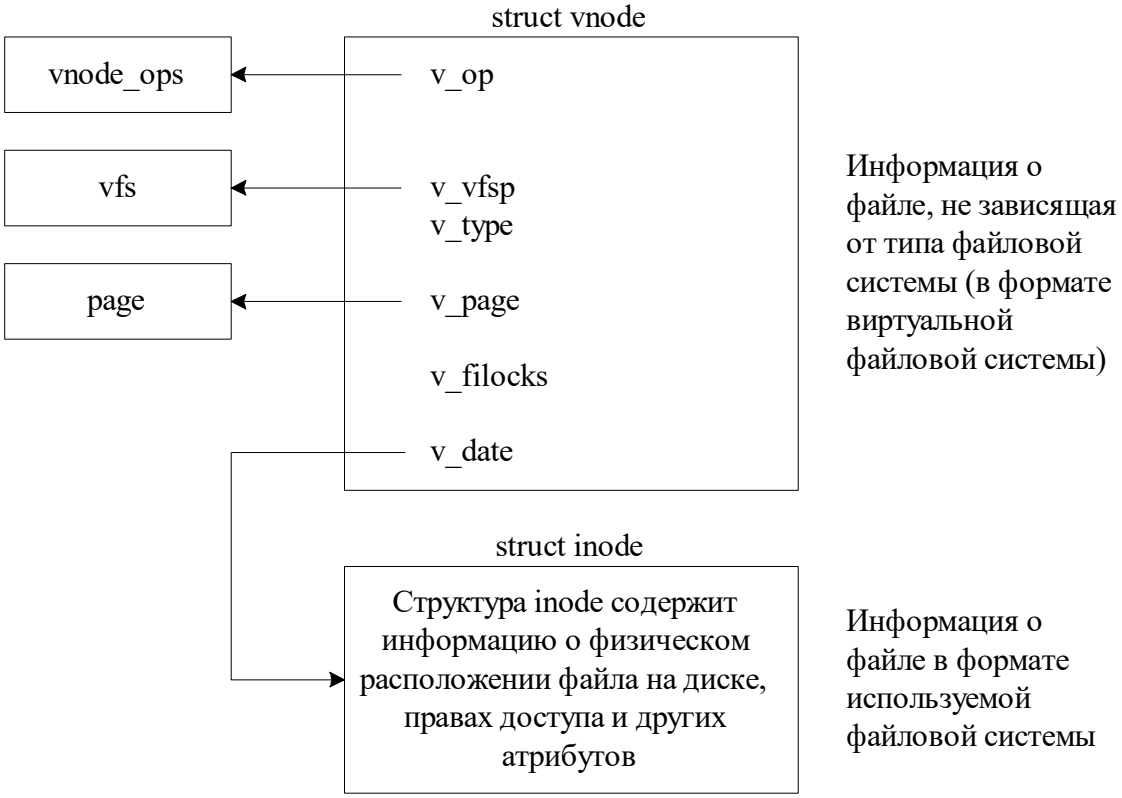


Рис. 8.5. Работа операционной системы с описателями файлов

В структуре vnode, приведенной на рис. 8.5, поле v_vsfp отвечает за используемый тип файловой системы; поле v_op – ссылка на функции, реализующие файловые операции для конкретной файловой системы; поле v_page – ссылка на таблицу страниц физической памяти; поле v_data, является ссылкой на структуру inode, описывающей физическое размещение файла на диске и атрибуты файла, зависимые от физической файловой системы, установленной на диске.

Для работы с файлами используется структура file, которая имеет следующие поля:

- flag - определение режима открытия (только для чтения, для чтения и записи и другие);
- struct vnode * f_vnode – указатель на структуру vnode;
- offset - смещение в файле при операциях чтения/записи;
- struct cred * f_cred – указатель на структуру, содержащую права процесса, открывшего файл (структура находится в дескрипторе процесса);
- struct file *next, *last – указатели на последующую и предыдущую структуру, для формирования динамического списка открытых файлов.

Сетевая файловая система NFS

Основная идея NFS (Network File System) – позволить различным пользователям разделять общую файловую систему.

NFS–сервер предоставляет свои каталоги для доступа удаленным клиентам. Список каталогов, которые сервер передает, содержится в файле /etc/exports, так что эти каталоги экспортируются сразу автоматически при загрузке сервера. Клиенты получают доступ к экспортируемым каталогам путем монтирования.

Взаимодействие клиент-сервер в системе NFS показано на рис. 8.6.

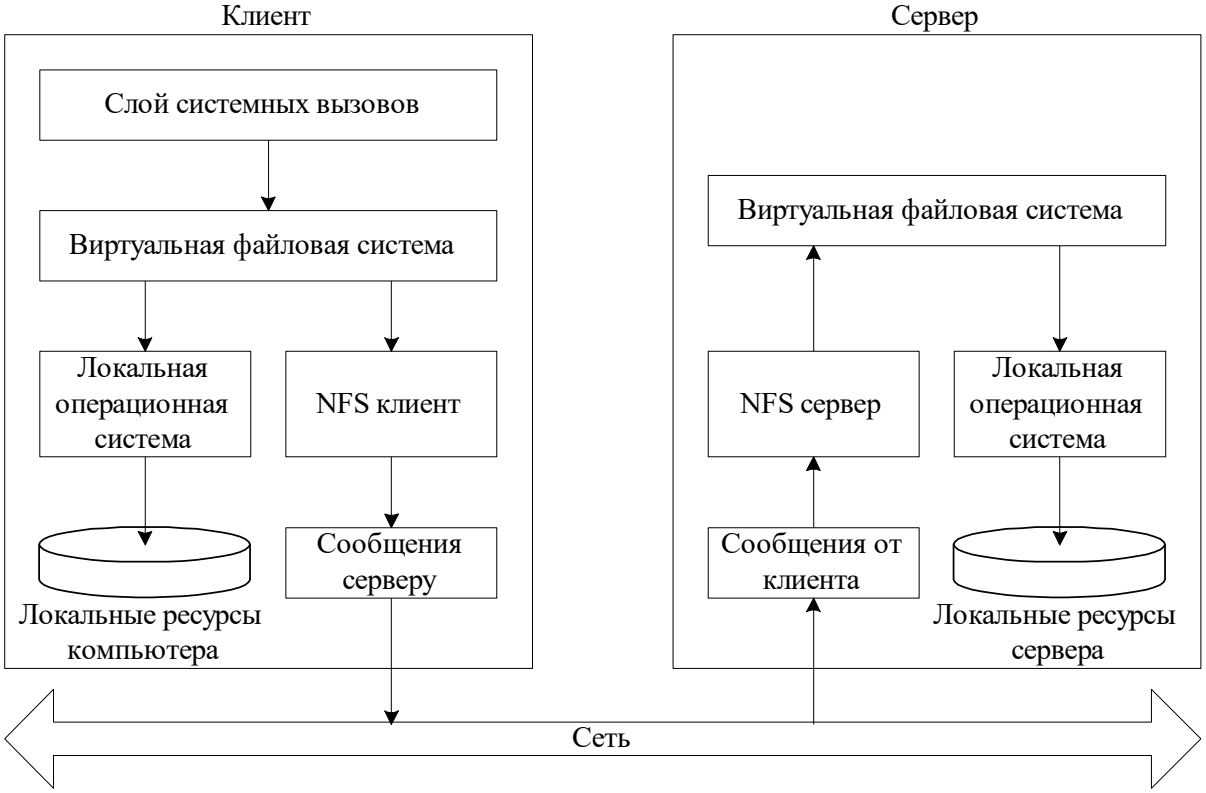


Рис. 8.6. Взаимодействие клиент-сервер в системе NFS

Верхний уровень клиента – уровень системных вызовов, таких как open, read, close. После синтаксического разбора, запрос проходит на виртуальную файловую систему (VFS). В создаваемой на этом уровне структуре vnode используется специальное поле, в котором кодируется вид открываемого файла: удаленный или локальный.

После этого через NFS клиента и сеть сообщение попадает на сервер. NFS сервера пересылает запрос в виртуальную файловую систему, откуда он пересылается локальной операционной системой сервера на физические устройства сервера.

Управление памятью

В UNIX реализована сегментно–страничная модель памяти и используется механизм свопинга, при котором на диск перемещаются страницы менее востребованного процесса.

Структурная схема работы с памятью операционной системы UNIX, приведена на рис. 8.7.

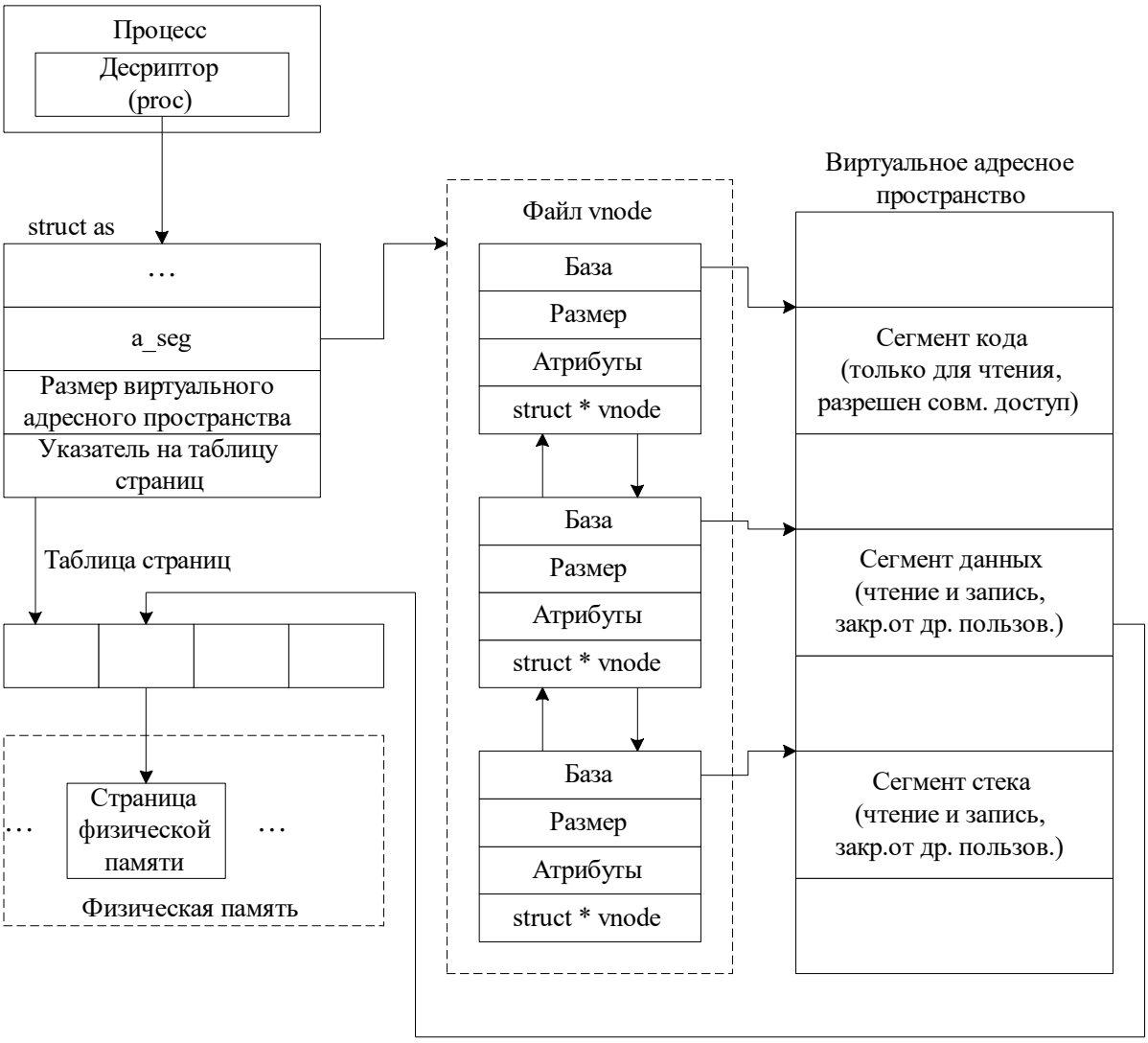


Рис. 8.7.Структура работы с памятью операционной системы UNIX

В дескрипторе процесса (proc) содержится указатель на структуру as, с помощью которой описываются виртуальные сегменты. Поле a_seg указывает на первый дескриптор сегмента процесса. Дескрипторы сегментов процесса связаны список. Дескриптор сегмента содержит базовый адрес начала сегмента в виртуальном адресном пространстве процесса, размер сегмента, а также указатели на операции, которые допускаются над этим сегментом (дублирование, освобождение, отображение и другие).

Для процесса различают следующие типы виртуальных сегментов:

- Текст (text) – содержит коды команд исполняемого модуля процесса. Данный сегмент имеет атрибут «только для чтения» и может разделяться нескольким процессами, использующими один набор функций.
- Данные (data) – содержит переменные процесса. Сегмент данных находится в монопольном использовании процесса и имеет доступ по чтению и записи.

- Стек (stack) – содержит стек процесса. Сегмент стека находится в монопольном использовании процесса и имеет доступ по чтению и записи.
- Разделяемая память – сегмент памяти, доступной для чтения и записи нескольким процессам.
- Отображенный файл – отображение файла в виртуальную память для ускорения доступа за счет использования стандартных функций операционной системы по выгрузке и загрузке страниц с диска.

Система ввода-вывода

Основу системы ввода–вывода операционной системы UNIX составляют драйверы внешних устройств и средства буферизации данных. Операционная система UNIX использует два различных интерфейса с внешними устройствами: байт–ориентированный и блок–ориентированный.

Подсистема буферизации

Запросы на ввод/вывод к блок–ориентированному устройству преобразуется в запрос к подсистеме буферизации, которая представляет собой буферный пул и комплекс функций управления этим пулом.

Буферный пул состоит из буферов, находящихся в области ядра. В заголовке буфера содержится следующая информация:

- Данные о состоянии буфера:
  - занят/свободен,
  - чтение/запись,
  - признак отложенной записи,
  - ошибка ввода-вывода.
- Данные об устройстве:
  - тип устройства,
  - номер устройства,
  - номер блока на устройстве.
- адрес буфера.
- Ссылка на следующий буфер в очереди буферов.

Алгоритм обработки запросов к системе буферизации приведен на рис. 8.8.

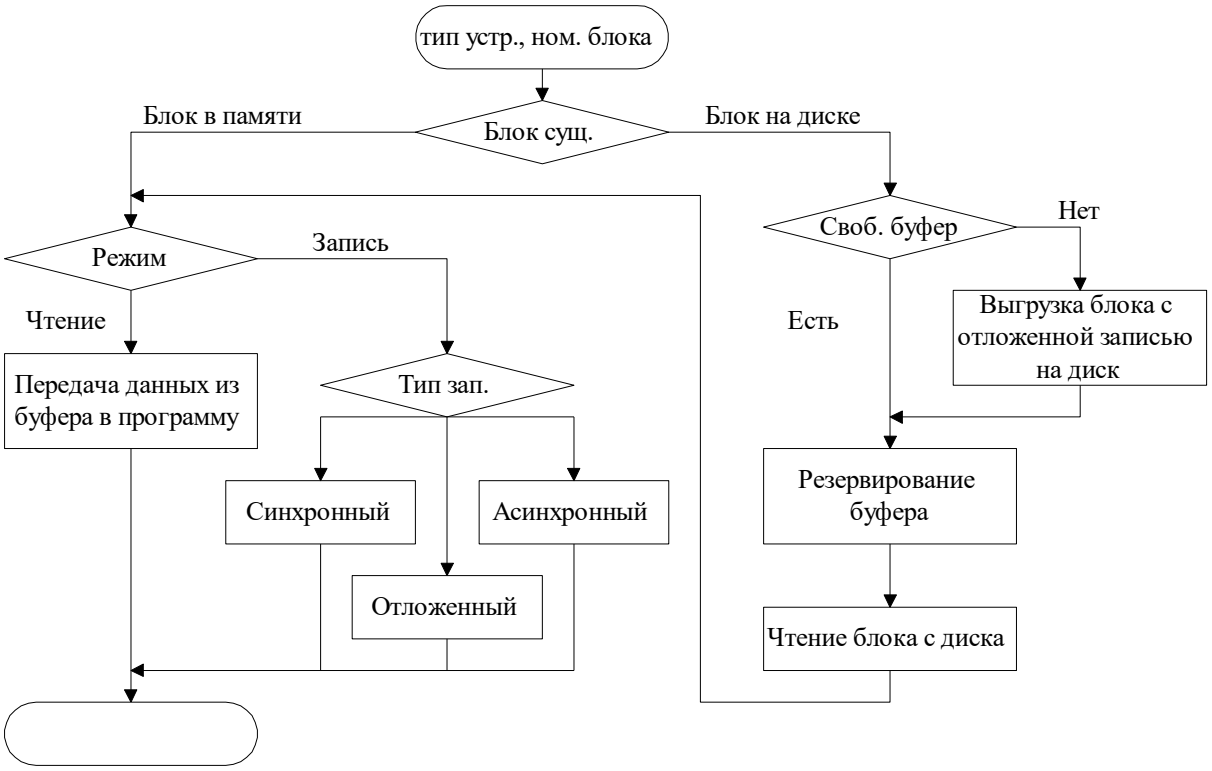


Рис. 8.8. Алгоритм обработки запросов к системе буферизации

Драйверы

Драйвер – это совокупность функций, предназначенных для управления передачей данных между внешним устройством и оперативной памятью. Адреса программных функций драйверов содержатся в двух таблицах:

- bdevsw – таблица блок–ориентированных устройств;
- cdevsw – таблица байт–ориентированных устройств.

Замена функции драйвера осуществляется изменением адреса в соответствующей строки одной из таблиц.

Драйвер блок–ориентированного устройства состоит из секций стратегии, секции устройства, модуля обработки прерывания, блок-схемы которых представлены на рис. 8.9.

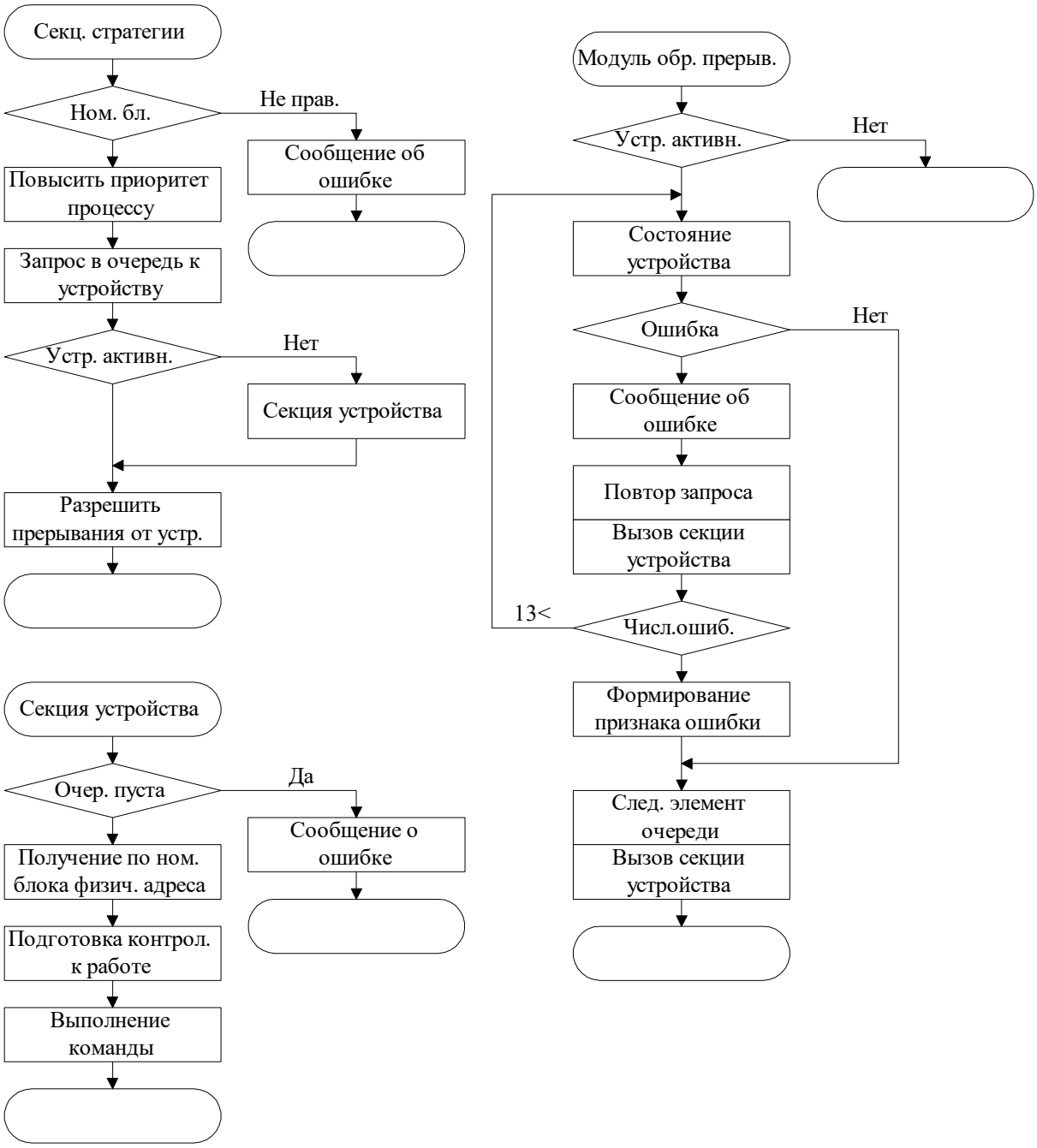


Рис. 8.9. Блок-схемы секций драйвера блок–ориентированного устройства

Секция стратегии проверяет правильность номера блока, формирует запрос к устройству, в случае необходимости запускает секцию устройства. Секция устройства отвечает за преобразование номера блока подготовку контроллера к работе, формирование и выполнение команды.

При выполнении операции, например ввода/вывода, устройство генерирует прерывание. Вектор прерывания содержит адрес функции драйвера отвечающего за обработку прерывания. Модуль обработки прерывания проверяет состояние устройства ввода/вывода, при необходимости обеспечивает повтор запроса или формирование признака ошибки и выбирает следующий элемент из очереди устройства.