МИНИСТЕРСТВО ОБРАЗОВАНИЯ И МОЛОДЕЖНОЙ ПОЛИТИКИ СТАВРОПОЛЬСКОГО КРАЯ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ГЕОРГИЕВСКИЙ РЕГИОНАЛЬНЫЙ КОЛЛЕДЖ «ИНТЕГРАЛ»

ПРЕДМЕТНО-ЦИКЛОВАЯ КОМИССИЯ
ИНФОРМАТИКИ И КОМПЬЮТЕРНОГО ОБУЧЕНИЯ

Щербинина Е.А.

Сборник лекций

учебно-методическое пособие
по дисциплине
«Операционные системы»

Георгиевск - 2016

Настоящее учебно-методическое пособие предназначено для студентов второго курса специальности 09.02.03 «Программирование в компьютерных системах».

Утверждено методическим советом ГРК «Интеграл»
протокол №____ от «____» _____2016 г.

Секретарь методического совета Е.В. Шахова

Утверждено на заседании ПЦК ИКО
протокол №____ от «____» _____ 2016 г.

Председатель ПЦК ИКО А.А. Костина

Согласовано:

Зав. отделением энергетики,
металлообработки и электроники В.Н. Булгарин

Щербинина Е. А. **Сборник лекций** / учебно-методическое пособие для студентов второго курса колледжа отделения энергетики, металлообработки и электроники по дисциплине «Операционные системы», 2016 г., 224 стр.

Настоящий «**Сборник лекций**» содержит теоретический материал по дисциплине «Операционные системы», отвечающий требованиям программы дисциплины для специальности 09.02.03 «Программирование в компьютерных системах».

Цель сборника - формирование у студентов понимания состава, функций и принципов работы операционных систем

Содержание:

Лекция 1
Основные понятия ОС

Из чего состоит любая вычислительная система?

В первую очередь, это то, что в англоязычных странах принято называть словом ***hardware***, или **техническое обеспечение:** процессор, память, монитор, дисковые устройства и т.д., обычно объединенные магистральным соединением, которое называется шиной

Во вторую очередь, это ***программное обеспечение.***

Все программное обеспечение принято делить на две части: ***прикладное и системное*****. К прикладному программному обеспечению,** как правило, относятся разнообразные банковские и прочие business программы, игры, текстовые процессоры, и т.п.

Под системным программным обеспечением обычно понимают программы, способствующие функционированию и разработке прикладных программ. Надо сказать, что деление на прикладное и системное программное обеспечение является отчасти условным и зависит от того, кто осуществляет такое деление. Так, обычный пользователь, неискушённый в программировании, может считать Microsoft Word системной программой, а с точки зрения программиста это приложение. Компилятор языка Си для обычного программиста - это системная программа, а для системного прикладная.

К системному ПО относят ПО самого низкого уровня.

Таким ПО являются:

- ✓ ОС,
- ✓ система управления файлами,
- ✓ интерфейсные оболочки для взаимодействия пользователя с ОС,
- ✓ системы программирования,
- ✓ утилиты.

Операционная система является фундаментальным компонентом системного программного обеспечения.

Что такое ОС

Большинство пользователей имеет свой опыт эксплуатации операционных систем, но, тем не менее, затруднятся дать точное определение. Давайте кратко рассмотрим основные точки зрения.

Операционная система как виртуальная машина

Использование большинства компьютеров на уровне машинного языка затруднительно, особенно это касается ввода-вывода. Например, для организации чтения блока данных с гибкого диска программист может использовать 16 различных команд, каждая из которых требует 13 параметров, таких как номер блока на диске, номер сектора на дорожке и т. п. Когда выполнение операции с диском завершается, контроллер возвращает 23 значения, отражающих наличие и типы ошибок, которые, очевидно, надо анализировать. Даже если не входить в курс реальных проблем программирования ввода-вывода, ясно, что среди программистов нашлось бы не много желающих непосредственно заниматься программированием этих операций. **При работе с диском программисту-пользователю достаточно представлять его в виде некоторого набора файлов, каждый из которых имеет имя. Работа с файлом заключается в его открытии, выполнении чтения или записи, а затем в закрытии файла.** Вопросы подобные таким, как следует ли при записи использовать усовершенствованную частотную модуляцию или в каком состоянии сейчас находится двигатель механизма перемещения считывающих головок, не должны волновать пользователя. **Программа, которая скрывает от программиста все реалии аппаратуры и предоставляет возможность простого, удобного просмотра указанных файлов, чтения или записи - это, конечно, операционная система. Точно также, как ОС ограждает программистов от аппаратуры дискового накопителя и предоставляет ему простой файловый интерфейс, операционная система берет на себя все малоприятные дела, связанные с обработкой прерываний, управлением таймерами и оперативной памятью, а также другие низкоуровневые проблемы.** В каждом случае та абстрактная, воображаемая машина, с которой, благодаря операционной системе, теперь может иметь дело пользователь, гораздо проще и удобнее в обращении, чем реальная аппаратура, лежащая в основе этой абстрактной машины.

С этой точки зрения функцией ОС является предоставление пользователю некоторой расширенной или виртуальной машины, которую

легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальную машину.

ОС как система управления ресурсами

Идея о том, что ОС прежде всего система, обеспечивающая удобный интерфейс пользователям, соответствует рассмотрению сверху вниз. Другой взгляд, снизу вверх, дает представление об **ОС как о некотором механизме, управляющем всеми частями сложной системы.** Современные вычислительные системы состоят из процессоров, памяти, таймеров, дисков, накопителей на магнитных лентах, сетевых коммуникационной аппаратуры, принтеров и других устройств.

В соответствии со вторым подходом функцией ОС является распределение процессоров, памяти, устройств и данных между процессами, конкурирующими за эти ресурсы. ОС должна управлять всеми ресурсами вычислительной машины таким образом, чтобы обеспечить максимальную эффективность ее функционирования. Критерием эффективности может быть, например, пропускная способность или реактивность системы.

Управление ресурсами включает решение двух общих, не зависящих от типа ресурса задач:

- ✓ **планирование ресурса - то есть определение, кому, когда, а для: делимых ресурсов и в каком количестве, необходимо выделить данный ресурс;**

- ✓ **отслеживание состояния ресурса - то есть поддержание оперативной информации о том, занят или не занят ресурс, а для делимых ресурсов - какое количество ресурса уже распределено, а какое свободно.**

Операционная система как защитник пользователей и программ

Если вычислительная система допускает совместную работу нескольких пользователей, то возникает проблема организации их безопасной деятельности. Необходимо обеспечить сохранность информации на диске, чтобы никто не мог удалить или повредить чужие файлы. Нельзя разрешить программам одних пользователей произвольно вмешиваться в работу программ других пользователей. Нужно пресекать

попытки несанкционированного использования вычислительной системы. Всю эту деятельность осуществляет операционная система как организатор безопасной работы пользователей и их программ. С такой точки зрения операционная система выглядит системой безопасности в государстве, на которую возложены полицейские и контрразведывательные функции.

Операционная система как постоянно функционирующее ядро

Наконец, можно дать и такое определение: **операционная система - это программа, постоянно работающая на компьютере и взаимодействующая со всеми прикладными программами.** Казалось бы, это абсолютно правильное определение, но, как мы увидим дальше, **во многих современных операционных системах постоянно работает на компьютере лишь часть операционной системы, которую принято называть ее ядром.**

Операционная система (ОС) - это программа, которая обеспечивает возможность рационального использования оборудования компьютера удобным для пользователя образом.

Операционная система (ОС) - это комплекс программ, предназначенных для управления вычислительными ресурсами и выполнением пользовательских программ

Операционная система (ОС) – это упорядоченная последовательность системных управляющих программ, совместно с необходимыми информационными массивами, предназначенных для планирования и исполнения пользовательских программ, управления всеми ресурсами вычислительной машины (программами, данными, аппаратурой и другими распределяемыми и управляемыми объектами) с целью предоставления возможности пользователям эффективно, в некотором смысле, решать задачи, сформулированные в терминах вычислительной машины.

Краткая история эволюции вычислительных систем

Мы будем рассматривать историю развития именно вычислительных, а не операционных систем, потому что hardware и программное обеспечение эволюционировали совместно, оказывая взаимное влияние друг на друга. По-

явление новых технических возможностей приводило к прорыву в области создания удобных, эффективных и безопасных программ, а новые идеи в программной области стимулировали поиски новых технических решений. Именно эти критерии **удобство, эффективность и безопасность** играли роль факторов естественного отбора при эволюции вычислительных систем.

Первый период (1945-1955). Ламповые машины. Операционные систем отсутствовали.

Мы начнем исследование развития компьютерных комплексов с появления электронных вычислительных систем (опуская историю механических и электромеханических устройств).

Первые шаги по созданию электронных вычислительных машин были предприняты в конце второй мировой войны. **В середине 40-х были созданы первые ламповые вычислительные устройства, и появился принцип программы, хранимой в памяти машины (John Von Neumann, июнь 1945г).** В то время одна и та же группа людей участвовала и в проектировании, и в эксплуатации, и в программировании вычислительной машины. **Это была скорее научно-исследовательская работа в области вычислительной техники,** а не регулярное использование компьютеров в качестве инструмента решения каких-либо практических задач из других прикладных областей. **Программирование осуществлялось исключительно на машинном языке. Об операционных системах не было и речи, все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления. За пультом мог находиться только один пользователь. Программа загружалась в память машины в лучшем случае с колоды перфокарт, а обычно с помощью панели переключателей. Вычислительная система выполняла одновременно только одну операцию (ввод-вывод, собственно вычисления, размышления программиста).** Отладка программ велась с пульта управления с помощью изучения состояния памяти и регистров машины. В конце этого периода появляется первое системное программное обеспечение: в 1951-52 гг. возникают прообразы первых компиляторов с символических языков (Fortran и др.), а в 1954 г. Nat Rochester разрабатывает ассемблер для IBM-701. В целом первый период характеризуется крайне высокой стоимостью вычислительных систем, их малым количеством и низкой эффективностью использования.

Второй период (1955-Начало 60-х). Компьютеры на основе транзисторов. Пакетные операционные системы

С середины 50-х годов начался новый период в эволюции вычислительной техники, связанный с появлением новой технической базы - полупроводниковых элементов. Применение транзисторов вместо часто перегоравших электронных ламп привело к повышению надежности компьютеров. Теперь они смогли непрерывно работать настолько долго, чтобы на них можно было возложить выполнение действительно практически важных задач. Снизилось потребление вычислительными машинами электроэнергии. Проще стали системы охлаждения. Размеры компьютеров уменьшились. Эксплуатация и обслуживание вычислительной техники подешевели. Началось использование ЭВМ коммерческими фирмами. **Одновременно наблюдается бурное развитие алгоритмических языков (ALGOL-58, LISP, COBOL, ALGOL-60, PL-1 и т.д.).** Появляются первые настоящие компиляторы, редакторы связей, библиотеки математических и служебных подпрограмм. **Упрощается процесс программирования. Пропадает необходимости взваливать на одних и тех же людей весь процесс разработки и использования компьютеров. Именно в этот период происходит разделение персонала на программистов и операторов, специалистов по эксплуатации и разработчиков вычислительных машин.**

Изменяется сам процесс прогона программ. **Теперь пользователь приносит программу с входными данными в виде колоды перфокарт и указывает требуемые для нее ресурсы. Такая колода получает название *задания*. Оператор загружает задание в память машины и запускает его на исполнение. Полученные выходные данные печатаются на принтере, и пользователь получает их обратно через некоторое (довольно большое) время.**

Смена запрошенных ресурсов вызывает приостановку выполнения программ. В результате процессор часто простаивает. Для повышения эффективности использования компьютера задания с похожими требуемыми ресурсами начинают собирать вместе, создавая *пакет заданий*.

Появляются первые системы пакетной обработки, которые просто автоматизируют запуск одной программы из пакета за другой и, тем самым, увеличивают коэффициент загрузки процессора. При реализации систем пакетной обработки был разработан формализованный язык управления заданиями, с помощью которого программист сообщал системе и оператору, какую работу он хочет выполнить на вычислительной машине. **Систе-**

мы пакетной обработки явились прообразом современных операционных систем, они стали первыми системными программами, предназначенными для управления вычислительным процессом.

Третий период (Начало 60-х - 1980). Компьютеры на основе интегральных микросхем. Первые многозадачные ОС.

Следующий важный период развития вычислительных машин относится к началу 60-х - 1980 годам. В это время в технической базе произошел переход от отдельных полупроводниковых элементов типа транзисторов к интегральным микросхемам. Вычислительная техника становится более надежной и дешевой. Растет сложность и количество задач, решаемых компьютерами. Повышается производительность процессоров.

Повышению эффективности использования процессорного времени мешает низкая скорость механических устройств ввода-вывода (быстрый считыватель перфокарт мог обработать 1200 перфокарт в минуту, принтеры печатали до 600 строк в минуту). **Вместо непосредственного чтения пакета заданий с перфокарт в память начинают использовать его предварительную запись сначала на магнитную ленту, а затем и на диск. Когда в процессе выполнения заданию требуется ввод данных, они читаются с диска. Точно так же выходная информация сначала копируется в системный буфер и записывается на ленту или диск, а реально печатается только после завершения задания.** Вначале действительные операции ввода-вывода осуществлялись в режиме off-line, то есть с использованием других, более простых, отдельно стоящих компьютеров. В дальнейшем они начинают выполняться на том же компьютере, который производит вычисления, то есть в режиме on-line. Такой прием получает название *spooling* (сокращение от Simultaneous Peripheral Operation On Line) или *подкачки-откачки данных*. Введение техники подкачки-откачки в пакетные системы позволило совместить реальные операции ввода-вывода одного задания с выполнением другого задания, но потребовало появления аппарата прерываний для извещения процессора об окончании этих операций.

Магнитные ленты были устройствами последовательного доступа, то есть информация считывалась с них в том порядке, в каком была записана. Появление магнитного диска, для которого не важен порядок чтения информации, то есть устройства прямого доступа, привело к дальнейшему развитию вычислительных систем. При обработке пакета заданий на магнитной ленте очередность запуска заданий определялась порядком их ввода. **При обработке пакета заданий на магнитном диске появляется возможность**

выбора очередного выполняемого задания. Пакетные системы начинают заниматься *планированием заданий*: в зависимости от наличия запрошенных ресурсов, срочности вычислений и т.д. на счет выбирается то или иное задание.

Дальнейшее повышение эффективности использования процессора было достигнуто с помощью *мультипрограммирования*. Идея мультипрограммирования заключается в следующем: пока одна программа выполняет операцию ввода-вывода, процессор не простаивает, как это происходило при однопрограммном режиме, а выполняет другую программу. Когда операция ввода-вывода заканчивается, процессор возвращается к выполнению первой программы. Эта идея напоминает поведение преподавателя и студентов на экзамене. Пока один студент (программа) обдумывает ответ на поставленный вопрос (операция ввода-вывода), преподаватель (процессор) выслушивает ответ другого студента (вычисления). Естественно, что такая ситуация требует наличия в комнате нескольких студентов. Точно также мультипрограммирование требует наличия в памяти нескольких программ одновременно. При этом каждая программа загружается в свой участок оперативной памяти, называемый разделом, и не должна влиять на выполнение другой программы. (Студенты сидят за отдельными столами и не подсказывают друг другу.)

Мультипрограммные пакетные системы дают окружение, в котором различные системные ресурсы (например, процессор, память, периферийные устройства) используются эффективно. И все же пользователь не мог непосредственно взаимодействовать с заданием и должен был предусмотреть с помощью управляющих карт все возможные ситуации. Отладка программ по-прежнему занимала много времени и требовала изучения многостраничных распечаток содержимого памяти и регистров или использования отладочной печати.

Появление электроннолучевых дисплеев и переосмысление возможностей применения клавиатур поставили на очередь решение этой проблемы. **Логическим расширением систем мультипрограммирования стали *time-sharing системы* или *системы разделения времени* **. В них процессор переключается между задачами не только на время операций ввода-вывода, но и просто по прошествии определенного интервала времени. Эти переключения происходят столь часто, что пользователи могут взаимодействовать со своими программами во время их выполнения, то есть интерактивно. В результате появляется возможность одновременной работы многих пользователей на одной компьютерной системе.** У

каждого пользователя для этого должна быть хотя бы одна программа в памяти. Чтобы уменьшить ограничения на количество работающих пользователей, была внедрена идея неполного нахождения исполняемой программы в оперативной памяти. Основная часть программы находится на диске и необходимый для ее дальнейшего выполнения кусок может быть легко загружен в оперативную память, а ненужный выкачан обратно на диск. Это реализуется с помощью механизма ***виртуальной памяти*.** Основным достоинством такого механизма является создание иллюзии неограниченной оперативной памяти ЭВМ.

В системах разделения времени пользователь получил возможность легко и эффективно вести отладку своей программы в интерактивном режиме, записывать информацию на диск, не используя перфокарты, а непосредственно с клавиатуры. Появление on-line файлов привело к необходимости разработки развитых *файловых систем*.

Параллельно внутренней эволюции вычислительных систем в этот период наблюдается и внешняя их эволюция. До начала этого периода вычислительные комплексы были, как правило, несовместимы. Каждый имела свою собственную специальную операционную систему, свою систему команд и т.д. В результате программу, успешно работающую на одном типе машин, необходимо было полностью переписать и заново отладить для другого типа компьютеров. В начале третьего периода появилась идея создания семейств программно-совместимых машин, работающих под управлением одной и той же операционной системы. Первым семейством программно-совместимых машин, построенных на интегральных микросхемах, явилась серия машин IBM/360. Построенное в начале 60-х годов это семейство значительно превосходило машины второго поколения по критерию цена/производительность. За ней последовала линия компьютеров PDP, несовместимых с линией IBM, кульминацией которой стала PDP-11.

Сила одной семьи была одновременно и ее слабостью. Широкие возможности этой концепции (наличие всех моделей: от миникомпьютеров до гигантских машин; обилие разнообразной периферии; различное окружение; различные пользователи) порождали сложную и огромную операционную систему. Миллионы строчек ассемблера, написанные тысячами программистов, содержали множество ошибок, что вызывало непрерывный поток публикаций о них и попыток их исправления. Только в операционной системе OS/360 содержалось более 1000 известных ошибок. Тем не менее, идея стандартизации операционных систем была широко внедрена в сознание пользователей и в дальнейшем получила активное развитие.

Четвертый период (1980-настоящее время). Персональные компьютеры. Классические, сетевые и распределенные системы.

Следующий период в эволюции вычислительных систем связан с появлением больших интегральных схем (БИС). В эти годы произошло резкое возрастание степени интеграции и удешевление микросхем. Компьютер, не отличающийся по архитектуре от PDP-11, по цене и простоте эксплуатации стал доступен отдельному человеку, а не отделу предприятия или университета. Наступила эра персональных компьютеров. Первоначально персональные компьютеры предназначались для использования одним пользователем в однопрограммном режиме, что повлекло за собой деградацию архитектуры этих ЭВМ и их операционных систем (в частности, пропала необходимость защиты файлов и памяти, планирования заданий и т.п.).

Компьютеры стали широко использоваться неспециалистами, что потребовало разработки "дружественного" программного обеспечения, это положило конец кастовости программистов.

Однако рост сложности и разнообразия задач, решаемых на персональных компьютерах, необходимость повышения надежности их работы привели к возрождению практически всех черт, характерных для архитектуры больших вычислительных систем.

В середине 80-х стали бурно развиваться сети компьютеров, в том числе персональных, работающих под управлением *сетевых* или *распределенных* операционных систем.

В сетевых операционных системах пользователи, при необходимости воспользоваться ресурсами другого сетевого компьютера, должны знать о его наличии и уметь это сделать. Каждая машина в сети работает под управлением своей локальной операционной системы, отличающейся от операционной системы автономного компьютера наличием дополнительных средств (программной поддержкой для сетевых интерфейсных устройств и доступа к удаленным ресурсам), но эти дополнения существенно не меняют структуру операционной системы.

Распределенная система, напротив, внешне выглядит как обычная автономная система. Пользователь не знает и не должен знать, где его файлы хранятся на локальной или удаленной машине, и где его программы выполняются. Он может вообще не знать, подключен ли его компьютер к сети. Внутреннее строение распределенной операционной системы имеет существенные отличия от автономных систем.

В дальнейшем автономные операционные системы мы будем называть *классическими* операционными системами.

Что мы вынесли из истории развития вычислительных систем?

Просмотрев этапы развития вычислительных систем, мы можем выделить **шесть основных функций**, которые выполняли классические операционные системы в процессе своей эволюции:

1. Планирование заданий и использования процессора.
2. Обеспечение программ средствами коммуникации и синхронизации.
3. Управление памятью.
4. Управление файловой системой.
5. Управление вводом-выводом.
6. Обеспечение безопасности

Каждая из приведенных функций обычно реализована в виде **подсистемы, являющейся структурным компонентом ОС.** В каждой конкретной операционной системе эти функции, конечно, реализовывались посвоему, в различном объеме. Они не были придуманы как составные части деятельности операционных систем изначально, а появились в процессе развития, по мере того, как вычислительные системы становились удобнее, эффективнее и безопаснее.

Эволюция вычислительных систем, созданных человеком пошла по такому пути, но никто еще не доказал, что это единственно возможный путь их развития. Операционные системы существуют потому, что на настоящий момент их существование - это разумный способ использования вычислительных систем.

Контрольные вопросы:

1. Дать определение понятию «программное обеспечение». Приведите классификацию ПО
2. Пояснить, на какие классы делится системное ПО
3. Пояснить, какие программы входят в состав базового системного ПО
4. Перечислить и охарактеризовать подходы к определению ОС
5. Дать определение понятию «операционная система»
6. Перечислить функции ОС

Лекция 2
Классификация операционных систем

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами.

Ниже приведена классификация ОС по нескольким наиболее основным признакам.

I. Особенности алгоритмов управления ресурсами

От эффективности алгоритмов управления локальными ресурсами компьютера во многом зависит эффективность всей сетевой ОС в целом. Поэтому, характеризуя сетевую ОС, часто приводят важнейшие особенности реализации функций ОС по управлению процессорами, памятью, внешними устройствами автономного компьютера. Так, например, **в зависимости от особенностей использованного алгоритма управления процессором, операционные системы делят на:**

- ✓ многозадачные и однозадачные,
- ✓ многопользовательские и однопользовательские,
- ✓ на системы, поддерживающие многонитевую обработку и не поддерживающие ее,
- ✓ на многопроцессорные и однопроцессорные системы.

Поддержка многозадачности

По числу одновременно выполняемых задач операционные системы могут быть разделены на два класса:

- ✓ однозадачные (например, MS-DOS, MSX) и
- ✓ многозадачные (ОС ЕС, OS/2, UNIX, Windows 95).

Однозадачные ОС в основном выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером.

Однозадачные ОС включают:

- ✓ средства управления периферийными устройствами,

- ✓ средства управления файлами,
- ✓ средства общения с пользователем.

Многозадачные ОС, кроме вышеперечисленных функций, **управляют разделением совместно используемых ресурсов,** таких как процессор, оперативная память, файлы и внешние устройства.

Вытесняющая и невытесняющая многозадачность. Важнейшим разделяемым ресурсом является **процессорное время**. Способ распределения процессорного времени между несколькими одновременно существующими в системе процессами (или нитями) во многом определяет специфику ОС. Среди множества существующих вариантов реализации многозадачности можно выделить две группы алгоритмов:

- ✓ невытесняющая многозадачность (NetWare, Windows 3.x);
- ✓ вытесняющая многозадачность (Windows NT, OS/2, UNIX).

Основным различием между **вытесняющим** и **невытесняющим** вариантами многозадачности является степень централизации механизма планирования процессов. В первом случае механизм планирования процессов целиком сосредоточен в операционной системе, а во втором - распределен между системой и прикладными программами.

При невытесняющей многозадачности активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению процесс. **(Т.е. это такой режим работы ОС, когда активный процесс, которому ОС выделила центральный процессор, монопольно использует его до тех пор, пока ему не потребуется выполнить какие-либо операции с внешними устройствами)**

При вытесняющей многозадачности решение о переключении процессора с одного процесса на другой принимается операционной системой, а не самим активным процессом.

Поддержка многопользовательского режима.

По числу одновременно работающих пользователей ОС делятся на:

- ✓ однопользовательские (MS-DOS, Windows 3.x, ранние версии OS/2);
- ✓ многопользовательские (UNIX, Windows NT).

Главным отличием **многопользовательских систем от однопользовательских** является **наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей.** Следует заметить, что не всякая многозадачная система является многопользовательской, и не всякая однопользовательская ОС является однозадачной.

Поддержка многонитевости

Важным свойством операционных систем является возможность распараллеливания вычислений в рамках одной задачи. **Многонитевая ОС разделяет процессорное время не между задачами, а между их отдельными ветвями (нитями).**

Многопроцессорная обработка

Другим важным свойством ОС является отсутствие или н**аличие в ней средств поддержки многопроцессорной обработки - мультипроцессирование. Мультипроцессирование** приводит к усложнению всех алгоритмов управления ресурсами.

В наши дни становится общепринятым введение в ОС функций поддержки многопроцессорной обработки данных. Такие функции имеются в операционных системах Solaris 2.x фирмы Sun, Open Server 3.x компании Santa Crus Operations, **OS/2** фирмы **IBM, Windows NT** фирмы **Microsoft и NetWare 4.1** фирмы **Novell.**

Многопроцессорные ОС могут классифицироваться по способу организации вычислительного процесса в системе с многопроцессорной архитектурой:

- ✓ **асимметричные ОС**
- ✓ **симметричные ОС.**

Асимметричная ОС целиком выполняется только на одном из процессоров системы, распределяя прикладные задачи по остальным процессорам.

Симметричная ОС полностью децентрализована и использует весь пул процессоров, разделяя их между системными и прикладными задачами.

Выше были рассмотрены характеристики ОС, связанные с управлением только одним типом ресурсов - **процессором**. Важное влияние на облик операционной системы в целом, на возможности ее использования в той или иной области оказывают особенности и других подсистем управления ло-

кальными ресурсами - подсистем управления памятью, файлами, устройствами ввода-вывода.

Специфика ОС проявляется и в том, каким образом она реализует сетевые функции:

- ✓ распознавание и перенаправление в сеть запросов к удаленным ресурсам,
- ✓ передача сообщений по сети,
- ✓ выполнение удаленных запросов.

При реализации сетевых функций возникает комплекс задач, связанных с распределенным характером хранения и обработки данных в сети:

- ✓ ведение справочной информации обо всех доступных в сети ресурсах и серверах,
- ✓ адресация взаимодействующих процессов,
- ✓ обеспечение прозрачности доступа,
- ✓ тиражирование данных,
- ✓ согласование копий,
- ✓ поддержка безопасности данных.

II. Особенности аппаратных платформ

На свойства операционной системы непосредственное влияние оказывают аппаратные средства, на которые она ориентирована.

По типу аппаратуры различают операционные системы:

- ✓ персональных компьютеров,
- ✓ мини-компьютеров,
- ✓ мейнфреймов,
- ✓ кластеров и сетей ЭВМ.

Среди перечисленных типов компьютеров могут встречаться как **однопроцессорные варианты**, так и **многопроцессорные**. В любом случае специфика аппаратных средств, как правило, отражается на специфике операционных систем.

Очевидно, что ОС большой машины является более сложной и функциональной, чем ОС персонального компьютера. Так в ОС больших машин функции по планированию потока выполняемых задач, очевидно, реализуются путем использования сложных приоритетных дисциплин и требуют боль-

шей вычислительной мощности, чем в ОС персональных компьютеров. Аналогично обстоит дело и с другими функциями.

Сетевая ОС имеет в своем составе средства передачи сообщений между компьютерами по линиям связи, которые совершенно не нужны в **автономной ОС**. На основе этих сообщений сетевая ОС поддерживает разделение ресурсов компьютера между удаленными пользователями, подключенными к сети. Для поддержания функций передачи сообщений сетевые ОС содержат специальные программные компоненты, реализующие популярные коммуникационные протоколы, такие как **IP, IPX, Ethernet** и другие.

Многопроцессорные системы требуют от операционной системы особой организации, с помощью которой сама операционная система, а также поддерживаемые ею приложения могли бы выполняться параллельно отдельными процессорами системы. Параллельная работа отдельных частей ОС создает дополнительные проблемы для разработчиков ОС, так как в этом случае гораздо сложнее обеспечить согласованный доступ отдельных процессов к общим системным таблицам, исключить эффект гонок и прочие нежелательные последствия асинхронного выполнения работ.

Другие требования предъявляются к **операционным системам кластеров. Кластер - слабо связанная совокупность нескольких вычислительных систем, работающих совместно для выполнения общих приложений, и представляющихся пользователю единой системой.** Наряду со специальной аппаратурой для функционирования кластерных систем необходима и программная поддержка со стороны операционной системы, которая **сводится в основном к синхронизации доступа к разделяемым ресурсам, обнаружению отказов и динамической реконфигурации системы.** Одной из первых разработок в области кластерных технологий были решения компании Digital Equipment на базе компьютеров VAX. Недавно этой компанией заключено соглашение с корпорацией Microsoft о разработке кластерной технологии, использующей Windows NT. Несколько компаний предлагают кластеры на основе UNIX-машин.

Наряду с ОС, ориентированными на совершенно определенный тип аппаратной платформы, **существуют операционные системы, специально разработанные таким образом, чтобы они могли быть легко перенесены с компьютера одного типа на компьютер другого типа, так называемые мобильные ОС.** Наиболее ярким примером такой ОС является популярная

система **UNIX**. **В этих системах аппаратно-зависимые места тщательно локализованы, так что при переносе системы на новую платформу переписываются только они. Средством, облегчающем перенос остальной части ОС, является написание ее на машинно-независимом языке, например, на С, который и был разработан для программирования операционных систем.**

III. Особенности областей использования

Многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

- ✓ системы пакетной обработки (например, ОС ЕС),
- ✓ системы разделения времени (UNIX, VMS),
- ✓ системы реального времени (QNX, RT/11).

Системы пакетной обработки предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является **максимальная пропускная способность, то есть решение максимального числа задач в единицу времени.** Для достижения этой цели в системах пакетной обработки используются следующая **схема функционирования**: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммная смесь, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие отличающиеся требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины; так, например, в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с интенсивным вводом-выводом. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается "выгодное" задание. **Следовательно, в таких ОС невозможно гарантировать выполнение того или иного задания в течение определенного периода времени.** В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказывается от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач. Таким образом,

взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что он приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя.

Системы разделения времени призваны исправить основной недостаток систем пакетной обработки - изоляцию пользователя-программиста от процесса выполнения его задач. **Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой. Так как в системах разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым. Если квант выбран достаточно небольшим, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них единолично использует машину.** Ясно, что системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе, и, кроме того, имеются накладные расходы вычислительной мощности на более частое переключение процессора с задачи на задачу. Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

Системы реального времени применяются **для управления различными техническими объектами**, такими, например, как станок, спутник, научная экспериментальная установка или технологическими процессами, такими, как гальваническая линия, доменный процесс и т.п. Во всех этих случаях существует предельно допустимое время, в течение которого должна быть выполнена та или иная программа, управляющая объектом, в противном случае может произойти авария: спутник выйдет из зоны видимости, экспериментальные данные, поступающие с датчиков, будут потеряны, толщина гальванического покрытия не будет соответствовать норме. Таким образом, **критерием эффективности для систем реального времени является их способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия). Это время называется временем реакции системы, а соответствующее свойство системы - реактивностью. Для этих систем мультипрограммная смесь представляет собой фиксированный набор заранее**

разработанных программ, а выбор программы на выполнение осуществляется исходя из текущего состояния объекта или в соответствии с расписанием плановых работ.

Некоторые операционные системы могут совмещать в себе свойства систем разных типов, например, часть задач может выполняться в режиме пакетной обработки, а часть - в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

IV. Особенности методов построения

При описании операционной системы часто указываются особенности ее структурной организации и основные концепции, положенные в ее основу.

К таким базовым концепциям относятся:

Способы построения ядра системы - монолитное ядро или микроядерный подход.

Большинство ОС использует **монолитное ядро, которое компонуется как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую, не требующие переключения из привилегированного режима в пользовательский и наоборот.**

Альтернативой является построение ОС на базе **микроядра**, **работающего также в привилегированном режиме и выполняющего только минимум функций по управлению аппаратурой,** в то время как функции ОС более высокого уровня выполняют специализированные компоненты ОС - серверы, работающие в пользовательском режиме. **При таком построении ОС работает более медленно, так как часто выполняются переходы между привилегированным режимом и пользовательским, зато система получается более гибкой** - ее функции можно наращивать, модифицировать или сужать, добавляя, модифицируя или исключая серверы пользовательского режима. Кроме того, серверы хорошо защищены друг от друга, как и любые пользовательские процессы.

Построение ОС на базе объектно-ориентированного подхода *дает возможность использовать все его достоинства, хорошо зарекомендовавшие себя на уровне приложений, внутри операционной системы, а именно: аккумуляцию удачных решений в форме стандартных объектов, возмож-*

ность создания новых объектов на базе имеющихся с помощью механизма наследования, хорошую защиту данных за счет их инкапсуляции во внутренние структуры объекта, что делает данные недоступными для несанкционированного использования извне, структуризованность системы, состоящей из набора хорошо определенных объектов.

Наличие нескольких прикладных сред дает возможность в рамках одной ОС одновременно выполнять приложения, разработанные для нескольких ОС. Многие современные операционные системы поддерживают одновременно **прикладные среды MS-DOS, Windows, UNIX (POSIX), OS/2** или хотя бы некоторого подмножества из этого популярного набора. Концепция множественных прикладных сред наиболее просто реализуется в ОС на базе микроядра, над которым работают различные серверы, часть которых реализуют прикладную среду той или иной операционной системы.

Распределенная организация операционной системы позволяет упростить работу пользователей и программистов в сетевых средах. **В распределенной ОС** реализованы механизмы, которые дают возможность пользователю представлять и воспринимать сеть в виде традиционного однопроцессорного компьютера. **Характерными признаками распределенной организации ОС являются:**

- ✓ наличие единой справочной службы разделяемых ресурсов, единой службы времени,
- ✓ использование механизма вызова удаленных процедур (RPC) для прозрачного распределения программных процедур по машинам, многонитевой обработки, позволяющей распараллеливать вычисления в рамках одной задачи и выполнять эту задачу сразу на нескольких компьютерах сети,
- ✓ наличие других распределенных служб.

Контрольные вопросы:

1. Охарактеризовать классификацию ОС по особенностям алгоритмов управления ресурсами: однозадачные и многозадачные
2. Охарактеризовать классификацию ОС по особенностям аппаратных платформ

3. Охарактеризовать классификацию ОС по особенностям алгоритмов управления ресурсами: однопользовательские и многопользовательские; однопроцессорные и многопроцессорные
4. Охарактеризовать классификацию ОС по особенностям методов построения
5. Охарактеризовать классификацию ОС по особенностям областей использования

Лекция 3
Организация данных в ОС

Для организации и хранения данных на машинных носителях используются файлы.

Файл – это последовательность произвольного числа байтов, обладающая уникальным собственным именем или поименованная область на машинных носителях.

Для обращения к файлу используется имя файла.
ИМЯ ФАЙЛА: ***имя.тип***

По способам именования файлов различают «короткое» и «длинное» имя.

Согласно соглашению, принятому в MS DOS, способом именования файлов на компьютерах IBM PC было **соглашение 8.3,** т.е. **имя файла состоит из двух частей: собственно имени и расширения имени. На имя файла отводится 8 символов, а на его расширение – 3 символа.**

Имя от расширения отделяется точкой. Как имя, так и расширение могут включать только алфавитно-цифровые символы латинского алфавита. Имена файлов, записанные в соответствии с соглашением 8.3, считаются «короткими».

Примеры имен файлов в MS DOS: **doom.exe, referat.doc.**

С появлением операционной системы Windows 95 было введено понятие «длинного» имени. Такое имя **может содержать до 256 символов.** Этого вполне достаточно для создания содержательных имен файлов.

«**Длинное» имя может содержать любые символы, кроме девяти специальных:**

\ / : * ? “ < > |

В имени разрешается использовать пробелы и несколько точек. Имя файла заканчивается расширением, состоящим из трех символов. Расширение используется для классификации файлов по типу.

Расширения имен файлов определяют их тип, то есть принадлежности к тем или иным программам, способы создания и назначения. То есть, в

большинстве случаев, по расширению файла можно понять, какого рода информацию он содержит. Например:

- ✓ **exe, bat, com, msi** – как правило, такие расширения имеют программы и исполняемые файлы.
- ✓ **sys, dll** – системные файлы и библиотеки.
- ✓ **txt –** файлы, содержащие внутри себя текст.
- ✓ **doc, docx–** текстовые файлы, созданные в Word.
- ✓ **xls, xlsx –** файлы Excel.
- ✓ **jpg, tif, bmp, gif, png –** графические файлы (фотографии, картинки).
- ✓ **avi, mov, wmv, mkv –** видеофайлы (фильмы, ролики).
- ✓ **mp3, wav, wma–** звуковые файлы (музыкальные композиции, звуковые дорожки).

Атрибуты файла — это дополнительная информация, относящаяся к данному файлу. Исторически атрибуты были введены в самых ранних версиях операционной системы MS DOS.

Существуют следующие атрибуты файла:

- ✓ **Только чтение (R)** - указывает, что данный **файл не подлежит изменению.** При установке этого атрибута для файла в него невозможно внести изменения. Например, вы сможете открыть текстовый документ с таким атрибутом в Блокноте, внести в него изменения, но при попытке сохранения появится сообщение об ошибке. Атрибут Только чтение застрахует файлы от изменения, но не спасет их от удаления.
- ✓ **Архивный (A)** - может использоваться программами резервного копирования, как знак того, что **данный файл изменялся** Данный атрибут не имеет особого значения для системы, а используется некоторыми программами архивации.
- ✓ **Скрытый (H)** - указывает, что **файл не следует отображать в обычном окне папки.**

  Файлы с таким атрибутом не отображаются в окне **Проводника**. Некоторые системные файлы Windows являются скрытыми, чтобы пользователи не могли их переместить или удалить. Хотя по умол-

чанию скрытые файлы и не видны, вы можете разрешить их отображение на вкладке **Вид** окна **Параметры папок**, установив переключатель **Скрытые файлы и папки** в положение **Показывать скрытые файлы, папки и диски**.

Как снять атрибут «Скрытый» для файлов и папок в Windows 7

В основном атрибуты скрытых файлов присваиваются системным файлам и скрыты они от пользователя в целях безопасности, например, от случайного удаления, и т. д. Но также присваивать атрибуты «скрытых» могут и вирусы. Рассмотрим несколько способов снятия атрибутов скрытых файлов.

Способ 1

Данный способ подойдет если вам нужно отредактировать скрытые системные файлы или просмотреть скрытые папки, но в случае если атрибуты изменены вирусами, может не помочь.

1. В меню **Пуск** открыть **Панель управления,** далее открыть **Параметры папок** (рис. 1).
2. В открывшемся окне **Параметры папок** (рис. 2), открыть вкладку **Вид** и снять флажок с параметра **Скрывать защищенные системные файлы**.

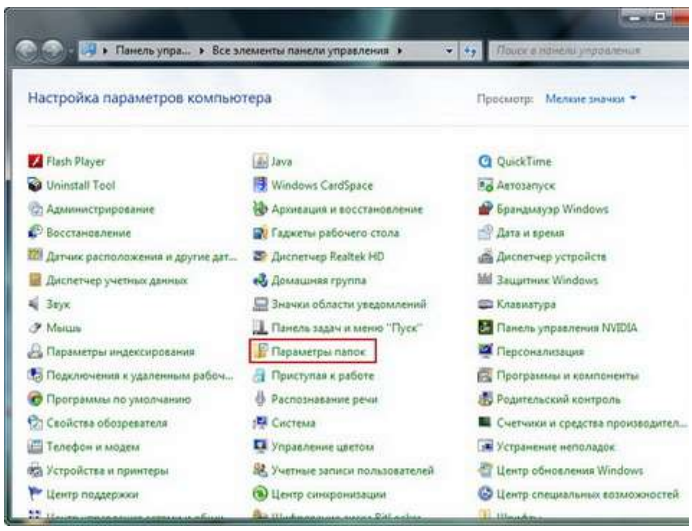

Рис. 1. Изменение параметров файлов и папок

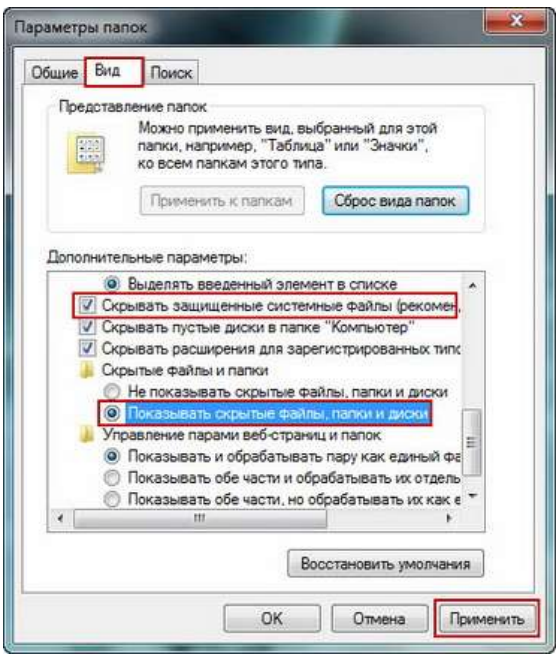

Рис. 2. Снятие атрибута «Скрытый»

3. После чего откроется окно с предупреждением (рис. 3), что вы хотите изменить параметр отображения для защищенных файлов системы, нажимаете кнопку **Да.**

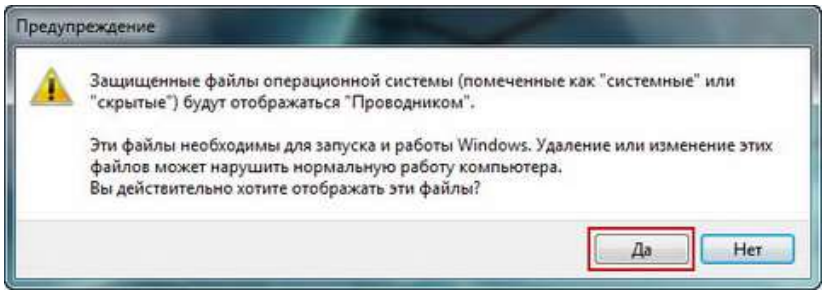

Рис. 3. Предупреждение

4. После чего (рис. 2) отметить параметр (поставить точку) **Показывать скрытые файлы, папки и диски**, и нажать кнопку **Применить.**

- ✓ **Системный (S)** - как бы **сочетает** в себе свойства атрибутов **Только чтение** и **Скрытый**, одновременно указывая на особую важность файла, снабженного таким атрибутом.
  Он автоматически устанавливается для важных системных файлов Windows, и вы не можете изменить его стандартными средствами. Для файлов с атрибутом **Системный** обычно также установлен атрибут **Скрытый**. Эти файлы будут отображаться в окнах папок, если разрешен показ скрытых файлов и дополнительно снят флажок **Скрывать защищенные системные файлы** на вкладке **Вид** окна **Свойства папки**.

Примечание
Изменить атрибут **Системный** можно в командной строке с помощью команды **attrib**, но удобнее это делать в файловом менеджере, например, Total Commander.

Внимание!
Не рекомендуется без особой необходимости включать отображение скрытых и системных файлов. Вряд ли они понадобятся вам в повседневной работе, а вот риск переместить или удалить по неосторожности важный системный файл существенно возрастает.

Если файл (или папка) расположен на разделе с файловой системой NTFS, то в окне его свойств будет присутствовать кнопка **Другие**, с помощью которой открывается окно установки дополнительных атрибутов (рис. 4).

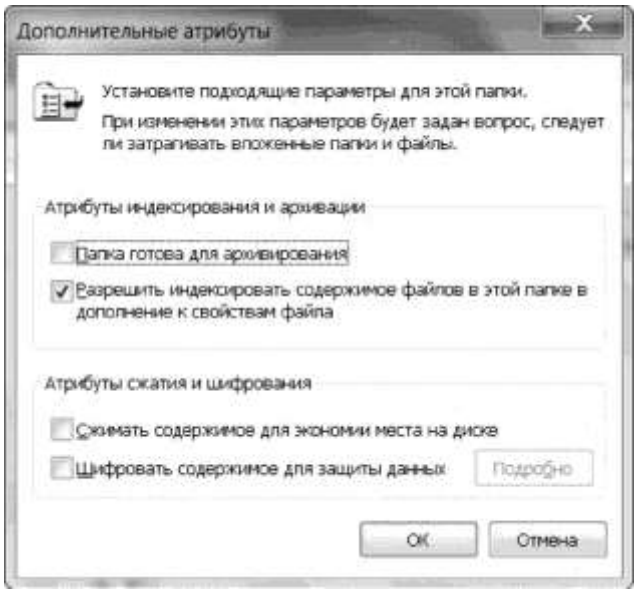

Рис. 4. Окно установки дополнительных атрибутов

Установка этих атрибутов позволяет задействовать соответствующие возможности файловой системы:

- ✓ **Разрешить индексировать содержимое** этого файла в дополнение к свойствам файла. Разрешает индексирование текста файла службой индексирования.
- ✓ **Сжимать содержимое** для экономии места на диске. Разрешает сжимать файл средствами NTFS.
- ✓ **Шифровать содержимое для защиты данных.** Надежный способ защитить конфиденциальные данные, даже если компьютер попадет в руки злоумышленников.

К атрибутам также относятся:

- ✓ Время создания.
- ✓ Дата создания.
- ✓ Дата последнего доступа.
- ✓ Время последней модификации.
- ✓ Дата последней модификации.
- ✓ Номер начального кластера файла в таблице расположения файлов.
- ✓ Размер файла.

Файловая структура

Вся совокупность файлов на диске и взаимосвязей между ними называется файловой структурой. Развитые операционные системы имеют **иерархическую — многоуровневую файловую структуру, организованную в виде дерева**.

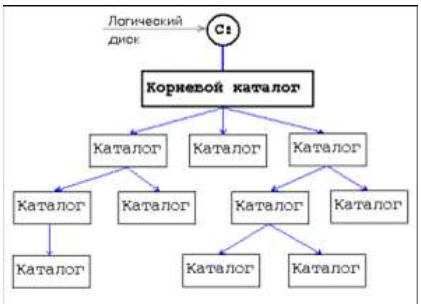

Рис. 5. Файловая структура в виде дерева

Рис. 6. Дерево каталогов диска G

Используется древовидная структура каталогов – **дерево каталогов**. Заимствована у Unix. **Иерархическая структура** – структура системы, части (компоненты) которой связаны отношениями включения или подчинения.

Иерархическая структура изображается ориентированным деревом, в котором вершины соответствуют компонентам, а дуги – связям.

Ориентированное дерево – это граф с выделенной вершиной (корнем), в котором между корнем и любой вершиной существует единственный путь. При этом возможны два варианта ориентации: либо все пути ориентированы от корня к листьям, либо все пути ориентированы от листьев к корню.

Каталог (директория) - группа файлов, объединенных по какому-либо признаку.

Различают **два состояния каталога** – текущее (активное) и пассивное.

Текущий (активный) каталог – каталог, в котором работа пользователя производится в текущее машинное время.

Пассивный каталог – каталог, с которым в данный момент времени нет связи.

Типы каталогов:

Корневой (главный) каталог – имеется на каждом диске, создается при форматировании диска, имеет ограниченный размер и не может быть удален средствами MS DOS. В корневой каталог входят другие каталоги и файлы, созданные командами ОС.

Родительский каталог – каталог, имеющий подкаталоги.

Подкаталог – каталог, который входит в другой каталог.

Для того, чтобы найти имеющиеся файлы в данной иерархической файловой системе надо указать **путь к файлу.** В путь к файлу входят записываемые через разделитель \ логическое имя диска и последовательность имен воженных друг в друга каталогов, в последнем из которых содержится нужный файл.

Например, пути к файлам можно записать следующим образом:

C:\TEXT\ Proba.txt

C:\GAMES\CHESS\Chess.exe

Путь файла вместе с именем файла называют полным именем файла.

Полное имя файла:

имя диска\путь\имя файла

Путь – цепочка соподчиненных каталогов, которую необходимо пройти к каталогу, где зарегистрирован искомый файл.

Файловая система – часть операционной системы, управляющая размещением и доступом к файлам и каталогам на диске.

Всякая операционная система создает на каждом томе (диске, пакете дисков, CD-ROM и пр.) совокупность системных данных, которая называется **файловой системой** (файловой структурой).

Файловая система (пустая) создается при инициализации (разметке) тома, затем корректируется ОС (подсистемой управления данными) при текущей работе, в процессе создания, удаления, модификации (увеличения или уменьшения объема) файлов пользователя, содержащих программы или данные.

Специальное системное программное обеспечение, реализующее работу с файлами по принятым спецификациям файловой системы, часто называют **системой управления файлами.** Именно системы управления файлами отвечают за создание, уничтожение, организацию, чтение, запись, модифика-

цию и перемещение файловой информации, а также за управление доступом к файлам и за управление ресурсами, которые используются файлами.

Назначение системы управления файлами — предоставление более удобного доступа к данным, организованным как файлы, то есть вместо низкоуровневого доступа к данным с указанием конкретных физических адресов нужной нам записи используется логический доступ с указанием имени файла и записи в нем.

Благодаря системам управления файлами пользователям предоставляются следующие возможности:

- ✓ создание, удаление, переименование (и другие операции) именованных наборов данных (файлов) из своих программ или посредством специальных управляющих программ, реализующих функции интерфейса пользователя с его данными и активно использующих систему управления файлами;
- ✓ работа с недисковыми периферийными устройствами как с файлами;
- ✓ обмен данными между файлами, между устройствами, между файлом и устройством (и наоборот);
- ✓ работа с файлами путем обращений к программным модулям системы управления файлами (часть API ориентирована именно на работу с файлами);
- ✓ защита файлов от несанкционированного доступа.

Как правило, все современные операционные системы имеют соответствующие системы управления файлами. А некоторые операционные системы имеют возможность работать с несколькими файловыми системами (либо с одной из нескольких, либо сразу с несколькими одновременно). В этом случае говорят о **монтируемых файловых системах** (монтируемую систему управления файлами можно установить как дополнительную), и в этом смысле они самостоятельны.

Для каждой операционной системы должна быть разработана соответствующая система управления файлами. И эта система управления файлами будет работать только в той операционной системе, для которой создана, но при этом обеспечит доступ к файлам, созданным с помощью системы управ-

ления файлами другой операционной системы, но работающей по тем же основным принципам файловой системы.

Таким образом, **файловая система — это множество именованных наборов данных, организованное по принятым спецификациям, которые определяют способы получения адресной информации, необходимой для доступа к этим файлам.**

Принцип организации файловой системы – табличный. Данные о том, в каком месте на диске записан файл, хранится в **таблице размещения файлов** (File Allocation Table, **FAT**).

Файловая система FAT

Файловая система FAT (File Allocation Table — таблица размещения файлов) получила свое название благодаря простой таблице, в которой указываются:

- ✓ непосредственно адресуемые участки логического диска, отведенные для размещения в них файлов или их фрагментов;
- ✓ свободные области дискового пространства;
- ✓ дефектные области диска (эти области содержат дефектные участки и не гарантируют чтение и запись данных без ошибок).

В файловой системе FAT дисковое пространство любого логического диска делится на две области:

- ✓ системную область
- ✓ область данных.

Рис. 7. Структура логического диска в FAT

Системная область логического диска создается и инициализируется **при форматировании**, а в последующем обновляется при работе с файловой структурой.

Область данных логического диска содержит обычные файлы и файлы-каталоги; эти объекты образуют иерархию, подчиненную корневому каталогу. Элемент каталога описывает файловый объект, который может быть либо обычным файлом, либо файлом-каталогом. Область данных, в отличие от системной области, доступна через пользовательский интерфейс операционной системы.

Системная область состоит из следующих компонентов (расположенных в логическом адресном пространстве друг за другом):

- ✓ загрузочной записи (Boot Record, BR);
- ✓ зарезервированных секторов (Reserved Sectors, ResSec);
- ✓ таблицы размещения файлов (File Allocation Table, FAT) - в целях защиты тома на нем хранятся две копии FAT. В случае повреждения первой копии FAT дисковые утилиты могут воспользоваться второй копией для восстановления тома;
- ✓ корневого каталога (Root Directory, RDir).

Область данных — совокупность блоков на диске, идентифицируемых своими номерами/адресами. Обычно адрес блока состоит из трех чисел — **№ цилиндра** (совокупность дорожек, доступных при фиксированном положении блока головок считывающего устройства), **№ поверхности** (дорожки в цилиндре), **№ блока на дорожке**.

Всю область данных разбивают на так называемые **кластеры**. Кластер представляет собой один или несколько смежных секторов в логическом дисковом адресном пространстве (точнее — только в области данных). **Кластер — это минимальная адресуемая единица дисковой памяти, выделяемая файлу (или некорневому каталогу).** Кластеры введены для того, чтобы уменьшить количество адресуемых единиц в области данных логического диска.

Таблица содержания. Пример простейшей (абстрактной) таблицы содержания, оглавления тома (диска, пакета дисков), которая в разных ОС имеет различные наименования — VTOC — Volume Table of Content (Таблица Содержания Тома), **FAT** — File Allocation Table (Таблица Размещения Фай-

лов), **FDT** — File Definition Table (Таблица Определения Файлов) и т. п., приведен в таблице:

Простейшая таблица оглавления тома

Имя файла (заглавная запись)	Номера блоков, выделенных для размещения файлов				
File 1	1	**3**	**7**	5	**13**
File 2	41	8			
File 3					
File 4	**3**				
Область переполнения					
File 1	23				
Список свободных блоков					
2	4	6	9	10	11	**12**
13						
Список сбойных блоков						
12	24	**7**				

Таблица состоит из трех областей:

- ✓ **область файлов.** Это таблица, имеющая обычно ограниченное (в приведенном примере *N = 6*) число строк *N* (в MS-DOS, например, *N = 500*, т. е. число файлов не более 500). Количество столбцов *M* (в примере *M = 5*) обычно выбирается из тех соображений, чтобы 85—95 % файлов, создаваемых пользователями, содержало бы не более *M* блоков, что зависит как от размера блока и типа пользователя, так и от общего уровня развития информационного и программного обеспечения. **Первый столбец таблицы в каждой строке *(заглавная запись — Title Record)* содержит данные о файле, в данном примере — имя файла;**

- ✓ **область переполнения —** дополнительная таблица аналогичной структуры, в которую записываются номера блоков особо длинных файлов (в примере — File_l). Организация таблицы размещения в форме области файлов и области переполнения, очевидно, позволяет сэкономить на объеме таблицы в целом, не ограничивая в то же время вероятной длины файла;

- ✓ **список свободных блоков —** необходимая информация для размещения создаваемых или расширяемых файлов. Список создается при инициализации и включает все блоки, кроме поврежденных, а затем корректируется при создании, удалении, модификации файлов;

- ✓ **список сбойных блоков.** Это таблица, создаваемая при инициализации (разметке) тома (диска), пополняемая программами диагностики (примером которых может служить хорошо известный пользователям NDD — Norton Disk Doctor) и предотвращающая распределение испорченных областей на магнитном носителе под файлы данных.

Здесь не указаны такие известные атрибуты файлов, как длина (в байтах), время создания, тип (архивный, скрытый, только для чтения, не для исполнения и пр.), которые могут содержаться в заглавной записи таблицы.

В развитых системах коллективного пользования такие данные содержатся в специальных таблицах разделения полномочий, поскольку перечисленные и другие атрибуты должны быть соотнесены с конкретными пользователями.

Кроме того, где-то должны быть размещены метка тома (имя и тип/объем), количество занятого и свободного пространства и прочая совокупная информация по тому данных.

Перечислим особенности ситуации, зафиксированной в приведенном примере (в простейшей (искусственной) файловой системе).

File 1 занимает 6 блоков, это число больше максимального, поэтому адрес блока № 6 (23) размещен в таблице переполнения;

File 2 занимает 2 блока, что меньше ограничения, поэтому вся информация сосредоточена в области файлов.

Имеются следующие конфликтные ситуации:

- ✓ **File 3** не содержит ни одного блока (следовательно, **файл был удален**, но заглавная запись сохранилась);

- ✓ **File_4** и **File 1** ссылаются на блок № 3. Это ошибка, поскольку каждый блок должен быть закреплен за единственным файлом;

- ✓ **File 1** содержит ссылку на блок №7, помеченный как сбойный (нечитаемый). Это приведет к невозможности корректно полностью прочитать данный файл — ситуация, знакомая каждому, работав-

шему с НГМД;

- ✓ в списке свободных блоков содержатся номера блоков № 12 (помеченный как сбойный) и № 13 (распределенный под File_l).

Это очевидные свидетельства начавшегося **разрушения файловой системы.** Перечисленные конфликты могут иметь своими источниками сбои, программные ошибки (разработчиков ОС), некорректное завершение ОС или целенаправленную деятельность вирусных или иных злонамеренных программ.

Файловые системы VFAT и FAT32

Одной из важнейших характеристик исходной файловой системы FAT было использование имен файлов формата 8.3. К стандартной системе FAT (имеется в виду прежде всего реализация FAT16) добавились еще две разновидности, используемые в ОС от Microsoft (конкретно — в Windows 95 и Windows NT): **VFAT** (виртуальная система FAT) и система **FAT32**, используемая в одной из редакций ОС Windows 95 и Windows 98. Файловая система **FAT32** поддерживается и такими последними системами, как Windows Millennium Edition, Windows 2000 и Windows XP. Имеются реализации FAT32 и для Windows NT, и для Linux.

Основными недостатками файловых систем FAT и VFAT, которые привели к разработке новой реализации файловой системы, основанной на той же идее (таблице размещения файлов), являются **большие потери на кластеризацию при больших размерах логического диска и ограничения на сам размер логического диска.**

Файловая система **FAT32** является полностью самостоятельной 32-разрядной файловой системой и содержит многочисленные усовершенствования и дополнения по сравнению с предыдущими реализациями FAT. Самое принципиальное отличие заключается в том, что FAT32 намного эффективнее расходует дисковое пространство.

Файловая система HPFS

Файловая система **HPFS** (High Performance File System — высокопроизводительная файловая система) впервые появилась в операционных системах OS/2 1.2 и LAN Manager. Она была разработана совместными усилиями лучших специалистов компаний IBM и Microsoft на основе опыта IBM по созданию файловых систем MVS, VM/CMS и виртуального метода доступа.

Архитектура HPFS разрабатывалась, как файловая система для многозадачного режима и была призвана обеспечить высокую производительность при работе с файлами на дисках большого размера.

Принципы размещения файлов на диске, положенные в основу **HPFS,** увеличивают как производительность файловой системы, так и ее надежность и отказоустойчивость.

Для достижения этих целей предложено несколько идей:

- ✓ размещение каталогов в середине дискового пространства;
- ✓ использование методов бинарных сбалансированных деревьев для ускорения поиска информации о файле;
- ✓ рассредоточение информации о местоположении файловых записей по всему диску, при этом записи каждого конкретного файла размещаются (по возможности) в смежных секторах и поблизости от данных об их местоположении.

Файловая система NTFS

В название файловой системы **NTFS** (New Technology File System — файловая система новой технологии) входят слова «новая технология». Действительно, файловая система **NTFS** содержит ряд значительных усовершенствований и изменений. С точки зрения пользователей файлы по-прежнему хранятся в каталогах, при работе в среде Windows часто называемых папками (folders). Однако в ней появилось много новых особенностей и возможностей.

Основные возможности файловой системы NTFS

1. **Надежность**

Высокопроизводительные компьютеры и системы совместного использования должны обладать повышенной надежностью, которая является ключевым элементом структуры и функционирования NTFS. **Система NTFS обладает определенными средствами самовосстановления. Она поддерживает различные механизмы проверки целостности системы, включая ведение журналов транзакций, позволяющих воспроизвести файловые операции записи по специальному системному журналу. При протоко-**

лировании файловых операций система управления файлами фиксирует в специальном служебном файле (журнале) происходящие изменения. В начале операции, связанной с изменением файловой структуры, делается соответствующая пометка. Если во время файловых операций происходит какой-нибудь сбой, то из-за упомянутой отметки операция остается помеченной как незавершенная. При выполнении процедуры проверки целостности файловой системы после перезагрузки машины эти незавершенные операции отменяются, и файлы возвращаются в исходное состояние. Если же операция изменения данных в файлах завершается нормальным образом, то в файле журнала эта операция отмечается как завершенная.

Поскольку NTFS разрабатывалась как файловая система для серверов, для которых очень важно обеспечить бесперебойную работу без перезагрузок, в ней, как и в HPFS, для повышения надежности был введен **механизм аварийной замены дефектных секторов резервными**. Другими словами, **если обнаруживается сбой при чтении данных, то система постарается прочесть эти данные, переписать их в специально зарезервированное для этой цели пространство диска, а дефектные сектора пометить как плохие и более к ним не обращаться.**

2. Ограничения доступа к файлам и каталогам

Файловая система **NTFS** поддерживает объектную модель безопасности операционной системы **Windows NT** и рассматривает все тома, каталоги и файлы как самостоятельные объекты. Система **NTFS** обеспечивает **безопасность на уровне файлов и каталогов**. Это означает, что **разрешения доступа к томам, каталогам и файлам могут зависеть от учетной записи пользователя и тех групп, к которым он принадлежит.** Каждый раз, когда пользователь обращается к объекту файловой системы, его разрешения на доступ проверяются по уже упоминавшемуся **списку управления доступом (ACL) для данного объекта**. Если пользователь обладает необходимым уровнем разрешений, его запрос удовлетворяется; в противном случае запрос отклоняется. Эта модель безопасности применяется как при локальной регистрации пользователей на компьютерах с Windows NT, так и при удаленных сетевых запросах.

3. Расширенная функциональность

Система NTFS проектировалась с учетом возможного расширения. В ней были воплощены многие дополнительные возможности:

- ✓ **повышенная отказоустойчивость,**
- ✓ **эмуляция других файловых систем,**
- ✓ **мощная модель безопасности,**
- ✓ **параллельная обработка потоков данных и создание файловых атрибутов, определяемых пользователем**
- ✓ **сжатие как отдельных файлов, так и целых каталогов**
- ✓ в последней, пятой, версии NTFS введена возможность **шифрования хранимых файлов**. Здесь следует, однако, заметить, что у шифрующей файловой системы пока больше недостатков, чем достоинств, поэтому на практике ее применять не рекомендуется.

4. Поддержка дисков большого объема

Система **NTFS** создавалась с расчетом на работу с большими дисками. Она уже достаточно хорошо проявляет себя при работе с томами объемом 300-400 Мбайт и выше. Чем больше объем диска и чем больше на нем файлов, тем больший выигрыш мы получаем, используя NTFS вместо FAT16 или FAT32.

5. Разрешения NTFS

Разрешения NTFS (NTFS permissions) - это набор специальных расширенных атрибутов файла или каталога (папки), заданных для ограничения доступа пользователей к этим объектам. Они имеются только на томах, где установлена файловая система NTFS. Разрешения обеспечивают гибкую защиту, так как их можно применять и к каталогам, и к отдельным файлам; они распространяются как на локальных пользователей (работающих на компьютерах, где находятся защищенные папки и файлы), так и на пользователей, подключающих к ресурсам по сети.

Разрешения NTFS служат, прежде всего, для защиты ресурсов от локальных пользователей, работающих за компьютером, на котором

располагается ресурс. *Однако их можно использовать и для удаленных пользователей, подключающихся к общей папке по сети. Очевидно, что в этом случае на пользователей действуют два механизма ограничения в доступе к ресурсам: сначала сетевой, а уже затем локальный, файловый. Поэтому итоговые разрешения на доступ будут определяться как минимальные из сетевых и файловых разрешений. Здесь необходимо сказать, что итоговые сетевые разрешения на доступ к ресурсам, которыми будет обладать пользователь при работе в сети, вычисляются как максимум разрешений в списке разрешений доступа, поскольку пользователь может быть членом нескольких групп, которые упомянуты в списке. Аналогично и для разрешений NTFS: пользователь получает максимальные разрешения, перечисленные в списке управления доступом, и только разрешение No Access (нет доступа) может перечеркнуть все остальные разрешения.*

Разрешения NTFS обеспечивают высокую избирательность защиты: *для каждого файла в папке можно установить свои разрешения. Например, одному пользователю можно позволить считывать и изменять содержимое файла, другому только считывать, третьему вообще запретить доступ. Рекомендуется устанавливать разрешения в списках ACL, используя не учетные записи отдельных пользователей, а учетные записи групп пользователей.*

Контрольные вопросы:

1. Дать определение «файл». Пояснить понятие «длинное» имя файла, привести пример «длинного» имени файла. Какие символы нельзя использовать в имени файла?
2. Пояснить понятие «короткое» имя файла, привести пример «короткого» имени файла. Пояснить, что такое расширение файла, привести примеры расширений.
3. Пояснить, что такое «атрибуты файла». Перечислить известные атрибуты файла. Пояснить атрибут Скрытый.
4. Пояснить атрибут Только для чтения.
5. Пояснить атрибут Системный.
6. Пояснить атрибут Архивный.
7. Пояснить, что такое «файловая структура». Перечислить и охарактеризовать типы и состояния каталогов
8. Перечислить функции файловых систем
9. Пояснить, что такое «файловая система». На какие области делится файловая система FAT? Пояснить, из каких компонентов состоит системная область

10. На какие области делится файловая система FAT? Охарактеризовать область данных. Пояснить, что такое «кластер»
11. Перечислить и охарактеризовать области таблицы содержания
12. Перечислить и охарактеризовать возможности файловой системы NTFS

Лекция 4
Работа с дисками

Для того чтобы на диске можно было хранить информацию, диск должен быть отформатирован, то есть должна быть создана физическая и логическая структура диска.

Формирование физической структуры диска состоит в создании на диске концентрических дорожек, которые, в свою очередь, делятся на сектора. Для этого в процессе форматирования магнитная головка дисковода расставляет в определенных местах диска метки дорожек и секторов.

Логическая структура гибких дисков

Логическая структура магнитного диска представляет собой совокупность секторов (емкостью 512 байтов), каждый из которых имеет свой порядковый номер (например, 100). Сектора нумеруются в линейной последовательности от первого сектора нулевой дорожки до последнего сектора последней дорожки.

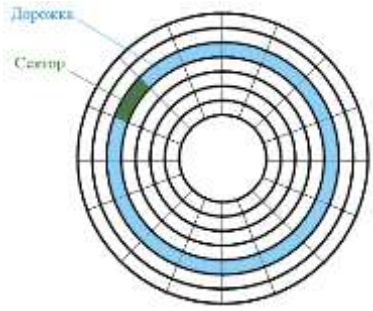

Рис. 8. Структура гибкого диска

У гибкого диска две стороны, на которых создается по **80 дорожек**. На каждой дорожке по **18 секторов**. **Общая емкость гибкого диска составляет:**

2 * 80 * 18 * 512 = 1474560 байт ≈ 1.44 Мбайт.

Самая первая дорожка магнитного диска (нулевая) считается служебной – там хранится служебная информация. Например, на этой дорожке хранится так называемая **таблица размещения файлов.**

На гибком диске минимальным адресуемым элементом является сектор.

При записи файла на диск будет занято всегда целое количество секторов, соответственно минимальный размер файла — это размер одного сектора, а максимальный соответствует общему количеству секторов на диске.

Логическая структура жестких дисков

Логическая структура жестких дисков несколько отличается от логической структуры гибких дисков.

Минимальным адресуемым элементом жесткого диска является кластер, который может включать в себя несколько секторов. Размер кластера зависит от типа используемой таблицы FAT и от емкости жесткого диска.

Таблица FAT16 может адресовать 2^{16} = 65 536 кластеров.

Для диска объемом 40 Гбайт размер кластера будет равен:
40 Гбайт/65536 = 655 360 байт = 640 Кбайт.

Файлу всегда выделяется целое число кластеров.

Например, текстовый файл, содержащий слово «информатика», составляет всего 11 байтов, но на диске этот файл будет занимать целиком кластер, то есть 640 Кбайт дискового пространства для диска емкостью 150 Гбайт. При размещении на жестком диске большого количества небольших по размеру файлов они будут занимать кластеры лишь частично, что приведет к большим потерям свободного дискового пространства.

Эта проблема частично решается с помощью использования таблицы FAT32, в которой **объем кластера принят равным 8 секторам или 4 килобайтам для диска любого объема.**

В целях более надежного сохранения информации о размещении файлов на диске хранятся **две идентичные копии таблицы FAT**.

В последнее время в основном на компьютерах с ОС Windows используется **файловая система NTFS.**

Файловая система NTFS – улучшенная файловая система, обеспечивающая уровень быстродействия и безопасности, а также дополнительные возможности, недоступные ни в одной версии файловой системы FAT.

Например, для обеспечения целостности данных тома в файловой системе NTFS используются стандартные технологии записи и восстановления транзакций.

В случае сбоя компьютера целостность файловой системы восстанавливается с помощью файла журнала NTFS и данных о контрольных точках.

В операционных системах Windows 2000 и Windows XP файловая система NTFS также обеспечивает такие дополнительные возможности, как разрешения для файлов и папок, шифрование и сжатие.

Операции с дисками

1. **Форматирование диска**

Форматирование диска — процесс разметки устройств хранения или носителей информации: жёстких дисков, устройств хранения на основе флеш-памяти, оптических носителей и др.

Существуют разные способы этого процесса.

Само форматирование заключается в создании (формировании) структур доступа к данным, например структур файловой системы. При этом, вся находящаяся на носителе информация теряется или уничтожается. В процессе форматирования также может проверяться целостность носителя.

Форматирование жесткого диска включает в себя три этапа:

1. **Форматирование диска на низком уровне (низкоуровневое форматирование).** Это единственный «настоящий» метод форматирования диска (физический уровень – сам магнитный слой).

При этом процессе на жестком диске создаются физические структуры: треки, сектора, управляющая информация. Этот процесс выполняется заводом-изготовителем на пластинах, которые не содержат еще никакой информации.

2. **Разбиение на разделы.** Этот процесс разбивает объем винчестера на логические диски (C, D, и т.д.). Этим обычно занимается операционная система, и метод разбиения сильно зависит от операционной системы.

3. **Высокоуровневое форматирование.** Этот процесс также контролируется операционной системой и зависит как от типа операционной системы, так и от утилиты, используемой для форматирования. Процесс записывает логические структуры, ответственные за правильное хранение файлов, а также, в некоторых случаях, системные загрузочные файлы в начало диска.

Это форматирование можно разделить на два вида: быстрое и полное.

При быстром форматировании перезаписывается лишь таблица файловой системы, **при полном** — сначала производится верификация (проверка) поверхности накопителя, а уже потом производится запись таблицы файловой системы.

После завершения процесса низкоуровневого форматирования винчестера, мы получаем диск с треками и секторами, но содержимое секторов будет заполнено случайной информацией.

Высокоуровневое форматирование — это процесс записи структуры файловой системы на диск, которая позволяет использовать диск в операционной системе для хранения программ и данных. В случае использования операционной системы DOS, для примера, команда format выполняет эту работу, записывая в качестве такой структуры главную загрузочную запись и таблицу размещения файлов. **Высокоуровневое форматирование выполняется после процесса разбивки диска на партиции (разделы), даже если будет использоваться только один раздел во весь объем накопителя.** В современных операционных системах процесс разбиения винчестера на разделы и форматирования может выполняться как в процессе установки операционной системы, так и на уже установленной системе, используя графический интуитивно понятный интерфейс. **Например, в случае Windows XP, это можно сделать следующим образом:**

- ✓ Щелкнуть правой кнопкой на значке **Мой компьютер** и выбрать управление,
- ✓ затем, раскрыв вкладку **Запоминающие устройства** выбрать пункт **Управление дисками**,
- ✓ после чего можно разбивать, форматировать, переразбивать разделы жесткого диска. Однако следует помнить, что изменения, внесенные как в разбивку диска, так и в форматирование, ведут к потере данных, находящихся на изменяемом диске.

Различие между высокоуровневым и низкоуровневым форматированием огромно. **Нет необходимости производить низкоуровневое форматирование для стирания информации с жесткого диска т.к. высокоуровневое форматирование подходит для большинства случаев.** Оно **перезаписывает служебную информацию файловой системы, делая винчестер чистым, однако, сами файлы при этом процессе не стираются, стирается только информация о местонахождении файла.**

Т.е. после высокоуровневого форматирования винчестера, содержавшего файлы, мы будем иметь чистый диск, свободный от каких-либо файлов, но, используя различные способы восстановления данных, можно добраться до старых файлов, которые были на диске до его форматирования. Единственным условием успеха в восстановлении данных является то, что файлы на диске перед форматированием не должны были быть фрагментированы.

2. Проверка диска

Проверка диска (Scan Disk) - исправляет физические и логические дефекты на диске, помечает сбойные блоки, чтобы данные не записывались в них.

Она позволяет избежать множества проблем, связанных с потерей информации.

Программа обнаруживает и устраняет ошибки, связанные со структурой файлов и каталогов, внутренней структурой сжатых дисков, с длинными именами файлов.

К ошибкам файловой структуры относятся потери фрагментов файлов или наличие фрагментов старых файлов. Возникают они обычно при попытке разместить два файла на одном участке диска либо при отказе в освобождении места при удалении файла.

Физические дефекты, связанные с поверхностью диска, выявляются в режиме полной проверки, которая занимает значительно больше времени, чем стандартная поверка. Зато позволит записи данных в такие участки диска, из которых невозможно будет считать. Необходимо корректно завершать работу - поможет избежать множества ошибок.

3. Дефрагментация диска

Дефрагментация диска (Defrag) - перераспределяет информацию на жестком диске, оптимизируя ее организацию.

Когда в область данных незаполненного диска производится запись первых файлов, эти файлы помещаются в непрерывные последовательности кластеров. Затем когда, в процессе работы происходит удаление некоторых файлов и запись на их место новых, файлы неизбежно разбиваются на фрагменты. Такая ситуация складывается вследствие интенсивной эксплуатации диска, когда часто записываются и удаляются большие программы или файлы данных.

Если файлы записаны на диске не непрерывно, а фрагментами, то доступ к нему занимает больше времени, чем могло бы быть при записи его непрерывным потоком.

Нельзя допускать чрезмерной фрагментации жесткого диска, которая замедляет чтение данных и ограничивает производительность ПК. При излишней фрагментации быстрее обычного изнашивается узел жесткого диска.

Перед запуском программы дефрагментации необходимо завершить работу всех остальных программ. Это связано с тем, что любое изменение информации на обрабатываемом диске, а это может случиться, если другие программы работают, приводит к повторному запуску дефрагментации.

Чтобы запустить программу дефрагментации диска в ОС Windows XP, нужно:

1. Открыть папку **Мой компьютер** и выбрать диск, на котором нужно выполнить дефрагментацию, щелкнув на нем правой кнопкой мыши.
2. В появившемся контекстном меню выбрать команду **Свойства**, чтобы открыть диалоговое окно настройки параметров диска.
3. Перейти на вкладку **Сервис**, на которой расположены кнопки запуска служебных программ. На этой вкладке нажать кнопку **Выполнить дефрагментацию**.

Также можно запустить программу дефрагментации, выбрав соответствующую команду в списке стандартных служебных программ главного меню **(Пуск – Программы – Стандартные – Служебные – Дефрагментация диска).**

В Windows 7 не нужно дефрагментировать диск, так как дефрагментация диска производится в фоновом режиме во время работы компьютера. При желании можно всегда проверить необходимость дефрагментации диска.

4. Очистка диска

Очистка диска - программа помогает очистить пространство на жестком диске.

Программа очистки диска проверяет диск и выводит перечень временных файлов, файлов Интернета, а также ненужных программных файлов, удаление которых не приведет к негативным последствиям. Можно выбрать удаление некоторых или всех этих файлов.

5. Уплотнение диска

Уплотнение диска (DriveSpace) - сжимает данные на жестком диске и дискетах.

Сжатые диски, создаваемые с целью экономии места на реальных физических дисках, сами не являются настоящими дисковыми устройствами.

Сжатый диск - это логическая структура, представляющая собой сжатый файл, который хранится на одном из обычных несжатых дисков. Этот файл называется CVF-файлом (файл сжатого тома) и располагается в корневом каталоге обычного диска. CVF-файл имеет атрибуты скрытого, системного и защищенного от записи файла.

Суть уплотнения с помощью DriveSpace заключается в том, что создаваемый CVF-файл содержит объем информации больший, чем размер занимаемого им пространства на несущем диске. Например, CVF-файл может занимать 100 Мбайт дискового пространства, в то время как объем размещенных в нем данных может составлять 200 Мбайт.

6. Архивация данных

Архивация данных (Backup) - создает резервные копии файлов на жестком диске для хранения файлов и программ, что позволяет восстановить случайные потери этих объектов из-за сбоев питания, вирусов и т.д. При архивации эти файлы не сжимаются, а занимают такие объемы, как и исходные.

Контрольные вопросы:

1. Пояснить, что такое «форматирование диска». Перечислить виды форматирования

2. Пояснить, что такое «форматирование диска». Пояснить, что такое низкоуровневое форматирование
3. Пояснить, что такое «форматирование диска». Пояснить, что такое высокоуровневое форматирование
4. Пояснить, что такое «дефрагментация диска»

Лекция 5
Состав операционных систем

Программы ОС постоянно (резидентно) занимают в оперативной памяти (ОП) объем, установленный при конфигурировании системы. Остальные части операционной системы по мере необходимости вызываются из внешней памяти на МД.

Операционная система обеспечивает осуществление в вычислительной системе следующих процессов:

- ✓ обработки задач;
- ✓ работы системы в режиме диалога и квантования времени;
- ✓ работы системы в реальном масштабе времени в составе многопроцессорных и многомашинных комплексов;
- ✓ связи оператора с системой;
- ✓ протоколирования хода выполнения вычислительных работ;
- ✓ обработки данных, поступающих по каналам связи;
- ✓ функционирования устройств ввода-вывода;
- ✓ использования широкого набора средств отладки и тестирования программ;
- ✓ планирования прохождения задач в соответствии с их приоритетами;
- ✓ ведения учета и контроля за использованием данных, программ и ресурсов ЭВМ.

Основные компоненты операционных систем — управляющие и обрабатывающие программы.

Управляющие программы управляют работой вычислительной системы, обеспечивая в первую очередь автоматическую смену заданий для поддержания непрерывного режима работы ЭВМ при переходе от одной программы к другой без вмешательства оператора.

Управляющая программа определяет порядок выполнения обрабатывающих программ и обеспечивает необходимым набором услуг для их выполнения.

Основные функции управляющей программы:

- ✓ последовательное или приоритетное выполнение каждой работы (управление задачами);
- ✓ хранение, поиск и обслуживание данных независимо от их организации и способа хранения (управление данными).

Программы управления задачами:

- ✓ считывают входные потоки задач, обрабатывают их в зависимости от приоритета, инициируют одновременное выполнение нескольких заданий;
- ✓ вызывают процедуры;
- ✓ ведут системный журнал.

Программы управления данными:

- ✓ обеспечивают способы организации, идентификации, хранения, каталогизации и выборки обрабатываемых данных
- ✓ управляют вводом-выводом данных с различной организацией, объединением записей в блоки и разделением блоков на записи, обработкой меток томов и наборов данных.

Программы управления восстановлением после сбоя:

- ✓ обрабатывают прерывания от систем контроля,
- ✓ регистрируют сбои в процессоре и внешних устройствах,
- ✓ формируют записи о сбое в журнале,
- ✓ анализируют возможность завершения затронутой сбоем задачи
- ✓ переводят систему в состояние ожидания, если завершение задачи невозможно.

В соответствии с рассмотренным подходом выделяют основные составляющие современной ОС:

- ✓ ядро,
- ✓ подсистема ввода-вывода,
- ✓ командный процессор,
- ✓ файловая система.

Состав операционной системы MS DOS

MS DOS -- первая операционная система для персональных компьютеров, которая получила широкое распространение. Со временем она была практически вытеснена новыми, современными операционными системами, типа Windows и Linux, но в ряде случаев MS DOS остается удобной и единственно возможной для работы на ЭВМ (устаревшая техника, давно написанное программное обеспечение и т. п.).

Дисковая операционная система – это программа, которая управляет всей работой ПК:

- ✓ Взаимодействием процессора с внешними устройствами разного типа
- ✓ Взаимодействием ПК с пользователем
- ✓ Запуском, выполнением и взаимодействием различных программ

Так как MS DOS была создана довольно давно (**1981 г.**), **она совершенно не соответствует требованиям, предъявляемым к современным операционным системам:**

- ✓ Она не может напрямую использовать большие объемы памяти, устанавливаемые в современные ЭВМ.
- ✓ В файловой системе используются только короткие имена файлов (8 символов в имени и 3 в расширении)
- ✓ Плохо поддерживаются разные устройства типа звуковых карт, видео-ускорителей и т. д.
- ✓ В MS DOS совершенно не реализована мультизадачность, т. е. она не может естественным образом выполнять несколько задач (работающих программ) одновременно. Поэтому она не может использоваться в качестве основной операционной системы для полноценной многопользовательской работы в сети.
- ✓ MS DOS не имеет никаких средств контроля и защиты от несанкционированных действий программ и пользователя, что привело к появлению огромного количества так называемых вирусов.

Состав MS DOS

В операционную систему MS DOS входят следующие основные модули:

1. Базовая система ввода – вывода (**BIOS**);
2. Блок начальной загрузки (**Boot Record**);
3. Модуль расширения BIOS (**IO.SIS**);
4. Модуль обработки прерываний (**MS DOS.SYS**);
5. Командный процессор (**COMMAND.COM**);
6. Файлы-драйверы, которые после их загрузки в память обеспечивают работу таких устройств, как мышь, CD-ROM и др.
7. Утилиты ОС, выполняющие различные сервисные функции (форматирование дисков и др.).

Базовая система BIOS (Basic Input/Output System) аппаратно зависима и находится в ПЗУ ПК

Она реализует следующие основные функции:

- ✓ Автоматическую проверку (тестирование) аппаратных компонентов при включении ПК;
- ✓ Вызов блока начальной загрузки ОС (загрузка в память программы операционной системы происходит в два этапа: сначала загружается блок начальной загрузки (Boot Record) и на него передается управление, затем с помощью этого блока - остальные модули).

Все программы BIOS расположены в ПЗУ компьютера => с одной стороны BIOS - часть компьютера, а с другой - компонент любой ОС, запускаемой на данном компьютере.

Блок начальной загрузки (Boot Record) – это очень короткая программа (около 512 байт), находящаяся в первом секторе каждого диска с операционной системой DOS. **Boot Record** загружает в память еще два модуля ОС (системные файлы **io.sys, msdos.sys**), которые завершают процесс загрузки DOS.

Дисковые файлы IO.SYS и MSDOS.SYS помещаются в оперативную память при загрузке и остаются в ней постоянно.

Файл IO.SYS вместе с файлом MSDOS.SYS составляют системное ядро ОС.

Файл **IO.SYS** представляет собой дополнение к базовой системе ввода-вывода (содержит подпрограммы ввода/вывода для конкретной реализации, которые используют или заменяют программы, находящиеся в BIOS), которое выполняет следующие функции:

- ✓ в процессе загрузки ОС выполняет логическую замену драйверов, хранящихся в BIOS, и подключение, если требуется, новых драйверов.
- ✓ организация интерфейса с BIOS.

Модуль расширения BIOS хранится на системном диске в виде файла с именем **IO.SYS** и является неотъемлемой частью MS DOS.

MSDOS.SYS реализует основные высокоуровневые услуги операционной системы, отвечает за:

- ✓ управление файлами;
- ✓ управление ресурсами сети;
- ✓ обработку ошибок;
- ✓ запуск и завершение выполнения программы

Командный процессор DOS обрабатывает команды, вводимые пользователем. Командный процессор находится в дисковом файле

COMMAND.COM на диске, с которого загружается операционная система.

Некоторые команды пользователя, например ***type, dir*** или ***copy***, командный процессор выполняет сам. Такие команды называются **внутренними** или **встроенными**.

Для выполнения остальных (**внешних**) команд пользователя командный процессор ищет на дисках программу с соответствующим именем и, если находит ее, загружает в память и передает ей управление. По окончании работы программы командный процессор удаляет программу из памяти и выводит сообщение о готовности к выполнению команд (приглашение DOS).

Драйверы устройств -- это специальные программы, которые дополняют систему ввода - вывода DOS и обеспечивают обслуживание новых или нестандартное использование имеющихся устройств. *Например, с помощью драйвера DOS* ***RAMDRIVE.SYS*** *возможна работа с "электронным диском", т.е. частью памяти компьютера, с которой можно работать так же, как с диском.* Драйверы помещаются в память компьютера при загрузке операционной системы, их имена указываются в специальном файле **CONFIG.SYS**. Такая схема облегчает добавление новых устройств и позволяет делать это, не затрагивая системные файлы DOS.

Файл автозагрузки AUTOEXEC.BAT служит для загрузки прикладных программ (например, Norton Commander) сразу же после загрузки ОС

Утилиты MS DOS реализуют выполнение внешних команд.

Внешние команды DOS -- это программы, поставляемые вместе с операционной системой в виде отдельных файлов (для их выполнения программа подгружается в оперативную память по требованию пользователя (format, copydisk))**.** Эти программы выполняют **действия обслуживающего характера**, например форматирование дискет (**format.com**), проверку состояния дисков (**scandisk.exe**) и т. д.

Утилиты могут находиться в виде программных файлов на любом диске. При этом имя файла, как правило, дублирует имя команды.

Загрузка MS DOS

Загрузка MS DOS – это считывание операционной системы из внешнего запоминающего устройства в оперативную память, ее настройка и запуск.

После включения питания компьютера, на котором установлена операционная система MS DOS, автоматически происходят следующие процессы:

- ✓ **Тестирование ПК** (BIOS выполняет комплекс программ начального тестирования компьютера);
- ✓ **Загрузка MS DOS** (считывание операционной системы из внешнего запоминающего устройства в оперативную память);
- ✓ **Настройка MS DOS** (настройка ОС выполняется по командам, записанным в файлах config.sys и autoexec.bat.).

После загрузки ОС на экране монитора высвечивается приглашение пользователю на ввод команд, которое состоит из имени диска и символов:

A:\> или C:\>.

Это означает, что DOS готова к приему команд.

Приглашение DOS содержит информацию о текущем дисководе и о текущем каталоге. Например,

A:\> - дисковод A:, корневой каталог:

C:\windows> - дисковод C:, каталог \windows.

Контрольные вопросы:

1. Перечислите программные модули, входящие в состав MS DOS
2. Укажите назначение модуля Boot Record
3. Дайте определение понятию «драйверы устройств». Для чего предназначен файл CONFIG.SYS?
4. Укажите назначение модуля MSDOS.SYS
5. Перечислите программные модули, входящие в состав ядра MS DOS. Для чего предназначен файл AUTOEXEC.BAT
6. Укажите назначение модуля IO.SYS
7. Перечислите виды команд в MS DOS. Дайте определения каждого вида команд. Приведите примеры команд

Приложение к лекции 5

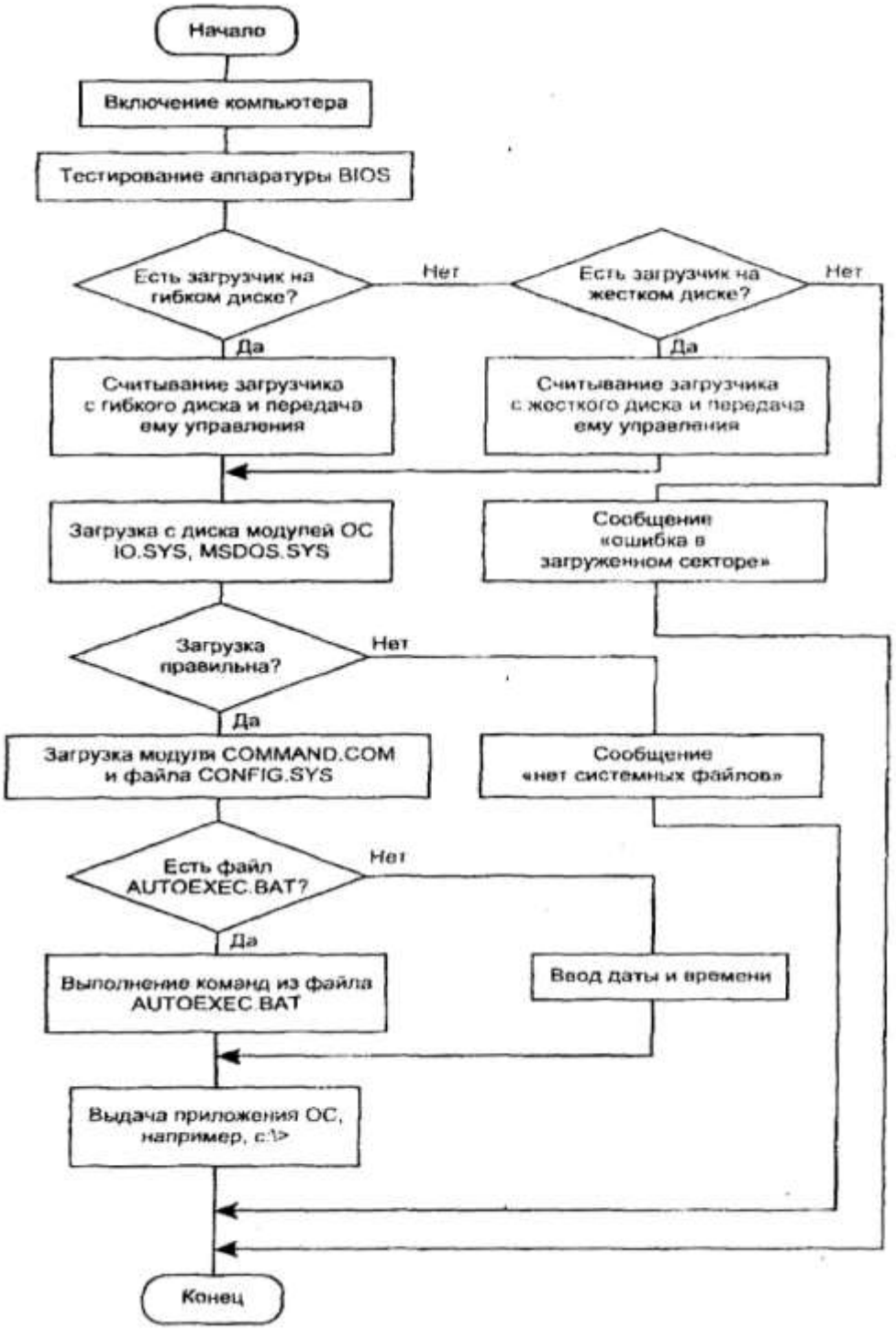

Рис. 9 Алгоритм загрузки ОС MS DOS

Лекция 6
Принципы работы ОС

Как любое техническое устройство, компьютер обменивается информацией с человеком посредством набора определенных правил, обязательных как для машины, так и для человека. Эти правила в компьютерной литературе называются интерфейсом. Интерфейс может быть понятным и непонятным, дружественным и нет. К нему подходят многие прилагательные. Но в одном он постоянен: он есть и никуда от него не денешься.

Интерфейс – это способ общения пользователя с персональным компьютером, пользователя с прикладными программами и программ между собой.

Интерфейс служит для удобства управления программным обеспечением компьютера.

От интерфейса зависит технология общения человека с компьютером.

Классификация интерфейсов

Интерфейс — это, прежде всего, набор правил. Как любые правила, их можно обобщить, собрать в «кодекс», сгруппировать по общему признаку. Таким образом, мы пришли к понятию «вид интерфейса» как объединение по схожести способов взаимодействия человека и компьютеров. Вкратце можно предложить следующую схематическую классификацию различных интерфейсов общения человека и компьютера (рис. 10).

Рис. 10. Типы интерфейсов, операционных систем и технологий реализации

Командный интерфейс

Командный интерфейс называется так потому, что в этом виде интерфейса человек подает «команды» компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки.

Пакетная технология

Исторически этот вид технологии появился первым. Она существовала уже на релейных машинах Зюса и Цюзе (Германия, 1937 г.). Идея ее проста: на вход компьютера подается последовательность символов, в которых по определенным правилам указывается последовательность запущенных на выполнение программ. После выполнения очередной программы запускается следующая и т.д. Машина по определенным правилам находит для себя команды и данные. В качестве этой последовательности может выступать, например, перфолента, стопка перфокарт, последовательность нажатия клавиш электрической пишущей машинки (типа CONSUL). Машина также выдает свои сообщения на перфоратор, алфавитно-цифровое печатающее устройство (АЦПУ), ленту пишущей машинки.

Такая машина представляет собой «черный ящик» (точнее «белый шкаф»), в который постоянно подается информация и которая также постоянно «информирует» мир о своем состоянии. Человек здесь имеет малое влияние на работу машины — он может лишь приостановить работу машины, сменить программу и вновь запустить ЭВМ. Впоследствии, когда машины стали помощнее и могли обслуживать сразу нескольких пользователей, вечное ожидание пользователей типа: «Я послал данные машине. Жду, что она ответит. И ответит ли вообще?» — стало, мягко говоря, надоедать. К тому же вычислительные центры, вслед за газетами, стали вторым крупным производителем макулатуры. Поэтому с появлением алфавитно-цифровых дисплеев началась эра по-настоящему пользовательской технологии — командной строки.

Технология командной строки

При этой технологии в качестве единственного способа ввода информации от человека к компьютеру служит клавиатура, а компьютер выводит информацию человеку с помощью алфавитно-цифрового дисплея (монитора). Эту комбинацию (монитор + клавиатура) стали называть терминалом, или консолью.

Команды набираются в командной строке. Командная строка представляет собой символ приглашения и мигающий прямоугольник

— курсор. При нажатии клавиши на месте курсора появляются символы, а сам курсор смещается вправо. Это очень похоже на набор команды на пишущей машинке. Однако в отличие от нее буквы отображаются на дисплее, а не на бумаге, и неправильно набранный символ можно стереть. Команда заканчивается нажатием клавиши **Enter** (или Return.) После этого осуществляется переход в начало следующей строки. Именно с этой позиции компьютер выдает на монитор результаты своей работы. Затем процесс повторяется.

Технология командной строки уже работала на монохромных алфавитно-цифровых дисплеях. Поскольку вводить позволялось только буквы, цифры и знаки препинания, то технические характеристики дисплея были не существенны. В качестве монитора можно было использовать телевизионный приемник и даже трубку осциллографа. Обе эти технологии реализуются в виде командного интерфейса — машине подаются на вход команды, а она как бы «отвечает» на них.

Преобладающим видом файлов при работе с командным интерфейсом стали текстовые файлы — их и только их можно было создать при помощи клавиатуры. На время наиболее широкого использования интерфейса командной строки приходится появление операционной системы **Unix** и появление первых восьмиразрядных персональных компьютеров с многоплатформенной операционной системой **CP/M**.

Графический интерфейс

Его идея зародилась в середине 70-х годов, когда в исследовательском центре Xerox Palo Alto Research Center (PARC) была разработана концепция визуального интерфейса. Предпосылкой графического интерфейса явилось уменьшение времени реакции компьютера на команду, увеличение объема оперативной памяти, а также развитие технической базы компьютеров. Аппаратным основанием концепции, конечно же, явилось появление алфавитно-цифровых дисплеев на компьютерах, причем на этих дисплеях уже имелись такие эффекты, как «мерцание» символов, инверсия цвета (смена начертания белых символов на черном фоне обратным, т.е. черных символов на белом фоне), подчеркивание символов. Эти эффекты распространились не на весь экран, а только на один или более символов. Следующим шагом явилось создание цветного дисплея, позволяющего выводить, вместе с этими эффектами, символы в 16 цветах на фоне с палитрой (т.е. цветовым набором) из 8 цветов. После появления графических дисплеев с возможностью вывода любых графических изображений в виде множества точек на экране различного цвета фантазии в использовании экрана вообще не стало границ! Первая система с графическим интерфейсом **8010 Star Infor-**

mation System группы **PARC**, таким образом, появилась за четыре месяца до выхода в свет первого компьютера фирмы **IBM** в **1981 г**. Первоначально визуальный интерфейс использовался только в программах. Постепенно он стал переходить и на операционные системы, используемые сначала на компьютерах **Atari** и **Apple Macintosh**, а затем и на **IBM**-совместимых компьютерах.

С более раннего времени, и под влиянием также и этих концепций, проходил процесс по унификации в использовании клавиатуры и мыши прикладными программами. Слияние этих двух тенденций и привело к созданию того пользовательского интерфейса, с помощью которого, при минимальных затратах времени и средств на переучивание персонала, можно работать с любыми программным продуктом.

Графический интерфейс пользователя (с 1974 г.) за время своего развития прошел две стадии.

Простой графический интерфейс

На первом этапе графический интерфейс очень походил на технологию командной строки.

Отличия от технологии командной строки заключались в следующем:

1. при отображении символов допускалось выделение части символов цветом, инверсным изображением, подчеркиванием и мерцанием. Благодаря этому повысилась выразительность изображения;

2. в зависимости от конкретной реализации графического интерфейса курсор может представляться не только мерцающим прямоугольником, но и некоторой областью, охватывающей несколько символов и даже часть экрана. Эта выделенная область отличается от других, невыделенных частей (обычно цветом);

3. нажатие клавиши **Enter** не всегда приводит к выполнению команды и переходу к следующей строке. Реакция на нажатие любой клавиши во многом зависит от того, в какой части экрана находился курсор;

4. кроме клавиши **Enter**, на клавиатуре все чаще стали использоваться «серые» клавиши управления курсором;

5. уже в этой редакции графического интерфейса стали использоваться манипуляторы (типа мыши, трекбола и т.п.). Они позволяли быстро выделять нужную часть экрана и перемещать курсор.

Подводя итоги, можно привести следующие особенности этого интерфейса:

- ✓ выделение областей экрана;
- ✓ переопределение клавиш клавиатуры в зависимости от контекста;
- ✓ использование манипуляторов и серых клавиш клавиатуры для управления курсором;
- ✓ широкое использование цветных мониторов.

Появление этого типа интерфейса совпадает с широким распространением операционной системы **MS DOS**. Именно она внедрила этот интерфейс в массы, благодаря чему **80-е годы** прошли под знаком совершенствования этого типа интерфейса, улучшения характеристик отображения символов и других параметров монитора.

Типичным примером использования этого вида интерфейса является файловая оболочка **Norton Commander** и текстовый редактор **Multi-Edit**. А текстовые редакторы **Лексикон**, **Chi Writer** и текстовый процессор **Microsoft Word for DOS** являются примером, как этот интерфейс превзошел сам себя.

WlMP-интерфейс

Вторым этапом в развитии графического интерфейса стал «чистый» интерфейс WIMP (Window — окно, Image — образ, Menu — меню, Pointer — указатель)

Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью **графических образов** — меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается «опосредственно», через графические образы.

Этот интерфейс характеризуется следующими особенностями:

1. **вся работа с программами, файлами и документами происходит в окнах** — определенных очерченных рамкой частях экрана;
2. **все программы, файлы, документы, устройства и другие объекты представляются в виде значков** — иконок (пиктограмм). При открытии иконки превращаются в окна;
3. **все действия с объектами осуществляются с помощью меню.** Хотя меню появилось на первом этапе становления графического интерфейса, оно не имело в нем главенствующего значения, а

служило лишь дополнением к командной строке. В чистом WIMP-интерфейсе меню становится основным элементом управления;

4. **широкое использование манипуляторов для указания на объекты.** Манипулятор перестает быть просто игрушкой — дополнением к клавиатуре, а становится основным элементом управления. С помощью манипулятора УКАЗЫВАЮТ на любую область экрана, окна или иконки, ВЫДЕЛЯЮТ ее, а уже потом через меню или с использованием других технологий осуществляют управление ими.

Основные элементы графических интерфейсов:

- ✓ управляющие кнопки
- ✓ поле ввода
- ✓ список
- ✓ раскрывающийся список
- ✓ поле ввода с раскрывающимся списком
- ✓ поле ввода со счетчиком
- ✓ флажок (независимый переключатель)
- ✓ зависимые переключатели (радиокнопки)
- ✓ регулятор

Следует отметить, что **WIMP требует для своей реализации цветной растровый дисплей с высоким разрешением и манипулятор.** Также программы, ориентированные на этот вид интерфейса, предъявляют повышенные требования к производительности компьютера, объему его памяти, пропускной способности шины и т.п. Однако этот вид интерфейса наиболее прост в усвоении и интуитивно понятен. Поэтому сейчас **WIMP-интерфейс** стал стандартом де-факто.

Ярким примером программ с графическим интерфейсом является операционная система **Microsoft Windows**.

SILK-интерфейс

С середины 90-х годов, после появления недорогих звуковых карт и широкого распространения технологий распознавания речи, появилась так называемая **«речевая технология» — SILK-интерфейс** (Speech — речь, Image — образ, Language - язык, Knowlege — знание)

Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный «разговор» чело-

века и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.

Речевая технология

При этой технологии команды подаются голосом путем произнесения **специальных зарезервированных слов — команд**, например:

«Проснись» — включение голосового интерфейса;

«Отдыхай» — выключение речевого интерфейса;

«Открыть» — переход в режим вызова той или иной программы. Имя программы называется в следующем слове;

«Буду диктовать» — переход из режима команд в режим набора текста голосом;

«Режим команд» — возврат в режим подачи команд голосом и некоторые другие.

Слова должны выговариваться четко, в одном темпе. Между словами обязательна пауза. Из-за неразвитости алгоритма распознавания речи такие системы требуют индивидуальной предварительной настройки на каждого конкретного пользователя.

Биометрическая технология

Эта технология возникла в конце 90-х годов XX века и на момент написания книги еще разрабатывается. Для управления компьютером используется выражение лица человека, направление его взгляда, размер зрачка и другие признаки. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация. Изображения считываются с цифровой видеокамеры, а затем с помощью специальных программ распознавания образов из этого изображения выделяются команды. Эта технология, по-видимому, займет свое место в программных продуктах и приложениях, где важно точно идентифицировать пользователя компьютера.

Семантический интерфейс

Этот вид интерфейса возник в конце **70-х годов XX века** с развитием **искусственного интеллекта**. Его трудно назвать самостоятельным видом интерфейса — **он включает в себя и интерфейс командной строки, и графический, и речевой, и мимический интерфейс.** Основная его отличитель-

ная черта — это отсутствие команд при общении с компьютером. Запрос формируется на естественном языке, в виде связанного текста и образов. По своей сути это трудно называть интерфейсом — это уже моделирование «общения» человека с компьютером.

С середины 90-х годов XX века автор уже не встречал публикаций, относящихся к семантическому интерфейсу. Похоже, что в связи с важным военным значением этих разработок (например, для автономного ведения современного боя машинами — роботами, для «семантической» криптографии) эти направления были засекречены. Информация о том, что эти исследования продолжаются, иногда появляется в периодической печати (обычно в разделах компьютерных новостей).

В 1992 году Япония представила миру ***действующие компьютеры пятого поколения, которые имели семантический интерфейс****, были основаны на логических выводах из базы знаний и должны были самообучаться, но реализация компьютеров оказалась непродуктивной — система, по свидетельству очевидцев, переходя через определенную точку, скатывалась в состояние потери надежности и утраты цельности, резко «глупела» и становилась неадекватной. Кроме того, специалисты считают ошибочным выбор языка типа Пролог для создания базы знаний и манипулирования данными. В 1980-е годы эта система программирования использовалась в экспертных системах, однако эксплуатация показала, что приложения оказываются малонадежными и плохо отлаживаемыми по сравнению с системами, разработанными обычными технологиями. Скорее всего, именно «фальстарт» японских инженеров по созданию интеллектуального компьютера и семантического интерфейса, в частности, наиболее существенно повлиял на снижение интереса к подобным разработкам.*

Интерфейс прикладного программирования API

Прежде всего необходимо однозначно разделить общий термин **API (application program interface, интерфейс прикладного программирования) на следующие направления**:

- ✓ API как интерфейс высокого уровня, принадлежащий к библиотекам RTL **(RTL (run time library)** — библиотека времени выполнения; она включает в себя те стандартные подпрограммы, которые система программирования подставляет на этапе компиляции. В общем случае RTL включает в себя не только модули из системы программирования, но и модули самой ОС);

- ✓ API прикладных и системных программ, входящих в поставку операционной системы;
- ✓ прочие API.

Интерфейс прикладного программирования, как это и следует из названия, предназначен для использования прикладными программами системных ресурсов ОС и реализуемых ею функций. API описывает совокупность функций и процедур, принадлежащих ядру или надстройкам ОС.

Итак, API представляет собой набор функций, предоставляемых системой программирования разработчику прикладной программы и ориентированных на организацию взаимодействия результирующей прикладной программы с целевой вычислительной системой. Целевая вычислительная система представляет собой совокупность программных и аппаратных средств, в окружении которых выполняется результирующая программа. Сама результирующая программа порождается системой программирования на основании кода исходной программы, созданного разработчиком, а также объектных модулей и библиотек, входящих в состав системы программирования.

В принципе API используется не только прикладными, но и многими системными программами как в составе ОС, так и в составе системы программирования.

Но дальше речь пойдет только о функциях API с точки зрения разработчика прикладной программы. Для системной программы существуют некоторые дополнительные ограничения на возможные реализации API.

Функции API позволяют разработчику строить результирующую прикладную программу так, чтобы использовать средства целевой вычислительной системы для выполнения типовых операций. При этом разработчик программы избавлен от необходимости создавать исходный код для выполнения этих операций. Программный интерфейс API включает в себя не только сами функции, но и соглашения об их использовании, которые регламентируются операционной системой (ОС), архитектурой целевой вычислительной системы и системой программирования.

Существует несколько вариантов реализации API:

- ✓ реализация на уровне ОС;
- ✓ реализация на уровне системы программирования;
- ✓ реализация на уровне внешней библиотеки процедур и функций.

Система программирования в каждом из этих вариантов предоставляет разработчику средства для подключения функций API к исходному коду программы и организации их вызовов. Объектный код функций API подключается к результирующей программе компоновщиком при необходимости.

Возможности API можно оценивать со следующих позиций:

- ✓ эффективность выполнения функций API — включает в себя скорость выполнения функций и объем вычислительных ресурсов, потребных для их выполнения;
- ✓ широта предоставляемых возможностей;
- ✓ зависимость прикладной программы от архитектуры целевой вычислительной системы.

В идеале хотелось бы иметь набор функций API, выполняющихся с наивысшей эффективностью, предоставляющих пользователю все возможности современных ОС и имеющих минимальную зависимость от архитектуры вычислительной системы (еще лучше — лишенных такой зависимости).

Добиться наивысшей эффективности выполнения функций API практически трудно по тем же причинам, по которым невозможно добиться наивысшей эффективности выполнения для любой результирующей программы. Поэтому об эффективности API можно говорить только в сравнении его характеристик с другим API.

Что касается двух других показателей, то в принципе нет никаких технических ограничений на их реализацию. Однако существуют организационные проблемы и узкие корпоративные интересы, тормозящие создание такого рода библиотек.

Реализация функций API на уровне ОС

При реализации функций API на уровне ОС за их выполнение ответственность несет ОС. Объектный код, выполняющий функции, либо непосредственно входит в состав ОС (или даже ядра ОС), либо поставляется в со-

ставе динамически загружаемых библиотек, разработанных для данной ОС. Система программирования ответственна только за то, чтобы организовать интерфейс для вызова этого кода.

В таком варианте результирующая программа обращается непосредственно к ОС. Поэтому достигается наибольшая эффективность выполнения функций API по сравнению со всеми другими вариантами реализации API.

Недостатком организации API по такой схеме является практически полное отсутствие переносимости не только кода результирующей программы, но и кода исходной программы. Программа, созданная для одной архитектуры вычислительной системы, не сможет исполняться на вычислительной системе другой архитектуры даже после того, как ее объектный код будет полностью перестроен. Чаще всего система программирования не сможет выполнить перестроение исходного кода для новой архитектуры вычислительной системы, поскольку многие функции API, ориентированные на определенную ОС, будут в новой архи-тектуре просто отсутствовать.

Таким образом, в данной схеме для переноса прикладной программы с одной целевой вычислительной системы на другую будет требоваться изменение Исходного кода программы.

Переносимости можно было бы добиться, если унифицировать функции API в различных ОС. С учетом корпоративных интересов производителей ОС такое направление их развития представляется практически невозможным. В лучшем случае при приложении определенных организационных усилий удается добиться стандартизации смыслового (семантического) наполнения основных функ-ций API, но не их программного интерфейса.

Хорошо известным примером API такого рода может служить набор функций, предоставляемых пользователю со стороны ОС типа Microsoft Windows — WinAPI (Windows API). Надо сказать, что даже внутри этого корпоративного API существует определенная несогласованность, которая несколько ограничивает переносимость программ между различными ОС типа Windows. Еще одним примером такого API можно считать набор сервисных функций ОС типа MS-DOS, реализованный в виде набора подпрограмм обслуживания программных прерываний.

Реализация функций API на уровне системы программирования

Если функции API реализуются на уровне системы программирования, они предоставляются пользователю в виде библиотеки функций соответствующего языка программирования. Обычно речь идет о библиотеке времени исполнения — RTL (run time library). Система программирования предоставляет пользователю библиотеку соответствующего языка программирования и обеспечивает подключение к результирующей программе объектного кода, ответственного за выполнение этих функций.

Очевидно, что эффективность функций API в таком варианте будет несколько ниже, чем при непосредственном обращении к функциям ОС. Так происходит, поскольку для выполнения многих функций API библиотека RTL языка программирования должна все равно выполнять обращения к функциям ОС. Наличие всех необходимых вызовов и обращений к функциям ОС в объектном коде RTL обеспечивает система программирования.

Однако переносимость исходного кода программы в таком варианте будет самой высокой, поскольку синтаксис и семантика всех функций будут строго регламентированы в стандарте соответствующего языка программирования. Они зависят от языка и не зависят от архитектуры целевой вычислительной системы. Поэтому для выполнения прикладной программы на новой архитектуре вычислительной системы достаточно заново построить код результирующей программы с помощью соответствующей системы программирования.

Единообразное выполнение функций языка обеспечивается системой программирования. При ориентации на различные архитектуры целевой вычислительной системы в системе программирования могут потребоваться различные комбинации вызовов функций ОС для выполнения одних и тех же функций исходного языка. Это должно быть учтено в коде RTL. В общем случае для каждой архитектуры целевой вычислительной системы будет требоваться свой код RTL языка программирования. Выбор того или иного объектного кода RTL для подключения к результирующей программе система программирования обеспечивает автоматически.

Реализация функций API с помощью внешних библиотек

При реализации функций API с помощью внешних библиотек они предоставляются пользователю в виде библиотеки процедур и функций, созданной сторонним разработчиком. Причем разработчиком такой библиотеки может выступать тот же самый производитель.

Система программирования ответственна только за то, чтобы подключить объектный код библиотеки к результирующей программе. Причем внешняя библиотека может быть и динамически загружаемой (загружаемой во время выполнения программы).

С точки зрения эффективности выполнения этот метод реализации API имеет самые низкие результаты, поскольку внешняя библиотека обращается как к функциям ОС, так и к функциям RTL языка программирования. Только при очень высоком качестве внешней библиотеки ее эффективность становится сравнимой с библиотекой RTL.

Если говорить о переносимости исходного кода, то здесь требование только одно — используемая внешняя библиотека должна быть доступна в любой из архитектур вычислительных систем, на которые ориентирована прикладная программа. Тогда удается достигнуть переносимости. Это возможно, если используемая библиотека удовлетворяет какому-то принятому стандарту, а система программирования поддерживает этот стандарт.

Например, библиотеки, удовлетворяющие стандарту POSIX (см. следующий подраздел), доступны в большинстве систем программирования для языка C. И если прикладная программа использует только библиотеки этого стандарта, то ее исходный код будет переносимым. Еще одним примером является широко известная библиотека графического интерфейса XLib, поддерживающая стандарт графической среды X Window.

Для большинства специфических библиотек отдельных разработчиков это не так. Если пользователь использует какую-то библиотеку, то она ориентирована на ограниченный набор доступных архитектур целевой вычислительной системы. Примерами могут служить библиотеки MFC (Microsoft foundation classes) фирмы Microsoft и VCL (visual controls library) фирмы Borland, жестко ориентированные на архитектуру ОС типа Windows.

Тем не менее многие фирмы-разработчики сейчас стремятся создать библиотеки, которые бы обеспечивали переносимость исходного кода приложений между различными архитектурами целевой вычислительной системы. Пока еще такие библиотеки не получили широкого распространения, но есть несколько попыток их реализации — например, библиотека CLX производства фирмы Borland ориентирована на архитектуру ОС типа Linux и ОС типа Windows.

В целом развитие функций прикладного API идет в направлении попытки создать библиотеки API, обеспечивающие широкую переносимость исходного кода. Однако, учитывая корпоративные интересы различных производителей и сложившуюся ситуацию на рынке системного программного обеспечения, в ближайшее время вряд ли удастся достичь значительных успехов в этом направлении. Разработка широко применимого стандарта API пока еще остается делом будущего.

Как правило, API не стандартизированы. В каждом конкретном случае набор вызовов API определяется, прежде всего, архитектурой ОС и ее назначением. В то же время принимаются попытки стандартизировать некоторый базовый набор функций, поскольку это существенно облегчает перенос приложений с одной ОС в другую. Таким примером может служить очень известный и, пожалуй, один из самых распространенных стандартов — стандарт POSIX. В этом стандарте перечислен большой набор функций, их параметров и возвращаемых значений. Стандартизированными, согласно POSIX, являются не только обращения к API, но и файловая система, организация доступа к внешним устройствам, набор системных команд1. Использование в приложениях этого стандарта позволяет в дальнейшем легко переносить такие программы с одной ОС в другую путем простейшей перекомпиляции исходного текста.

Частным случаем попытки стандартизации API является внутренний корпоративный стандарт компании Microsoft, известный как WinAPI. Он включает в себя следующие реализации: Winl6, Win32s, Win32, WinCE. С точки зрения WinAPI (в силу ряда идеологических причин — обязательный графический «оконный» интерфейс пользователя), базовой задачей является окно. Таким образом, WinAPI изначально ориентирован на работу в графической среде. Однако базовые понятия дополнены традиционными функциями, в том числе частично поддерживается стандарт POSIX.

Платформенно-независимый интерфейс POSIX

POSIX (Portable Operating System Interface for Computer Environments) — платформенно - независимый системный интерфейс для компьютерного окружения. Это стандарт IEEE **(IEEE (Institute of Electrical and Electronical Engineers)** — американский Институт инженеров по электротехнике и радиоэлектронике), описывающий системные интерфейсы для открытых операционных систем, в том числе оболочки, утилиты и инструментарии. Помимо этого, согласно POSIX, стандартизированными являются задачи обеспечения безопасности, задачи реального времени, процессы администрирования, сетевые функции и обработка транзакций. Стандарт базируется на UNIX-системах, но допускает реализацию и в других ОС.

В данном контексте под системными командами следует понимать некий набор программ, позволяющих управлять вычислительными процессами. Например, pstat, kill, dir и др.

POSIX возник как попытка всемирно известной организации IEEE пропагандировать переносимость приложений в UNIX-средах путем разработки абстрактного, платформенно-независимого стандарта. Однако POSIX не ограничивается только UNIX-системами; существуют различные реализации этого стандарта в системах, которые соответствуют требованиям, предъявляемым стандартом IEEE Standard 1003.1-1990 (POSIX.1). Например, известная ОС реального времени QNX соответствует спецификациям этого стандарта, что облегчает перенос приложений в эту систему, но UNIX-системой не является ни в каком виде, ибо ее архитектура использует абсолютно иные принципы.

Этот стандарт подробно описывает VMS (virtual memory system, систему виртуальной памяти), многозадачность (MPE, multiprocess executing) и технологию переноса операционных систем (CTOS). Таким образом, на самом деле POSIX представляет собой множество стандартов, именуемых POSIX. I — POSIX. 12. Следует также особо отметить, что POSIX. 1 предполагает язык С как основной язык описания системных функций API.

Таким образом, программы, написанные с соблюдением данных стандартов, будут одинаково выполняться на всех POSIX-совместимых системах.

Контрольные вопросы:

1. Перечислить и охарактеризовать типы интерфейсов ОС
2. Пояснить, что такое интерфейс прикладного программирования API
3. Перечислить функции API и принципы его реализации
4. Пояснить, что такое интерфейс POSIX

Лекция 7
Режимы работы операционных систем

Режимы обработки данных

Порядок представления прикладной программе перечисленных средств определяется **режимом обработки данных,** реализованных в операционной системе ЭВМ. Различают **однопрограммные и мультипрограммные режимы обработки данных** и, соответственно, работы ОС.

К однопрограммным режимам относятся:

- ✓ режим непосредственного доступа (РНД);
- ✓ пакетный однопрограммный режим (П1П).

Мультипрограммными режимами обработки данных являются:

- ✓ пакетный мультипрограммный режим (ПМП);
- ✓ режим разделения времени **(РРВ).**

Однопрограммные режимы обработки данных

Режим непосредственного доступа широко применялся в ЭВМ первого поколения и используется при работе с современными персональными компьютерами. Режим РНД характерен тем, что ЭВМ предоставляется только одному пользователю, который осуществляет взаимодействие с машиной посредством пульта управления (сейчас — клавиатура, мыши и дисплеи). Время решения каждой задачи в режиме РНД складывается из времени $T_{вв}$ ввода программы и данных в ЭВМ, времени $T_{цп}$ работы процессора над решением задачи, времени $T_{ву}$ обмена данными с внешними устройствами (включая вывод результатов в обработки), времени $T_{оп}$ обслуживания ЭВМ и задачи оператором ЭВМ при ее подготовке к запуску и по окончании решения задачи:

$$T_{нд} = T_{вв} + T_{цп} + T_{ву} + T_{оп}.$$

Коэффициент загрузки процессора при одной задаче составляет

$$\eta_{нд} = T_{цп}/T_{нд}.$$

Полное время решения N задач и коэффициент загрузки:

$$T_{\text{нд}}(N) = \sum_{i=1}^{N} [T_{\text{вв}}(i) + T_{\text{цп}}(i) + T_{\text{ву}}(i) + T_{\text{оп}}(i)];$$

$$\eta_{\text{нд}}(N) = \frac{1}{T_{\text{нд}}} \sum_{i=1}^{N} T_{\text{цп}}(i),$$

где i — номер задачи.

В РНД наличие ОС необязательно.

Недостатками режима РНД являются:

- ✓ аппаратура и программы ЭВМ используются неэффективно;
- ✓ велики затраты времени программиста на управление машиной;
- ✓ предъявляются высокие требования к подготовке пользователя как оператора вычислительной машины.

Пакетный однопрограммный режим (П1П) применяется в ВС, начиная с ЭВМ второго поколения. **Несколько заданий для решения задач обработки собираются в один пакет, называемый пакет заданий (ПЗ).** Пакет заданий оператор ЭВМ вводит в ЭВМ, где ПЗ сначала записывается во внешнюю память (магнитные диски, магнитные барабаны и т.п.). Затем операционная система машины последовательно считывает задания, входящие в ПЗ, и осуществляет выполнение необходимых в соответствиями с заданиями действий для решения задач пользователей. После завершения очередного задания происходит обращение к ОС, которая активирует начало выполнения следующего. После завершения последнего задания пакета оператор ЭВМ загружает в машину новый пакет заданий.

Режим П1П обладает следующими положительными чертами:

- ✓ более высокая пропускная способность;
- ✓ отсутствие специальных требований к аппаратуре ЭВМ;
- ✓ возможна его реализация на любой ЭВМ.

К недостаткам режима П1П относятся:

- ✓ необходимо наличие операционной системы;
- ✓ пользователь физически отделен от ЭВМ и решаемой им задачи;
- ✓ увеличивается реакция пользователя на полученные результаты решения;
- ✓ последовательный порядок выполнения заданий пакетов не позволяет увеличить загрузку оборудования вычислительной системы.

Многопрограммные режимы обработки данных

Пакетный мультипрограммный режим широко применяется в ЭВМ третьего и последующих поколений. ПМП является режимом классического мультипрограммирования, при котором в вычислительной системе находятся в обработке сразу несколько заданий. На входе в систему формируется набор пакетов заданий, которые оператор ЭВМ загружает в систему. После окончания ввода первого ПЗ операционная система начинает его обработку, не дожидаясь до ввода второго и последующих ПЗ. Задания, принадлежащие одному пакету, выполняются последовательно (т.е. в режиме П1П). Задания, принадлежащие разным пакетам, выполняются параллельно. Первым начинает выполняться первое задание первого пакета. По мере освобождения ресурсов ОС активизирует выполнение заданий из других пакетов в порядке их следования внутри ПЗ.

Пакетный мультипрограммный режим обеспечивает наивысшую пропускную способность вычислительной системы, что достигается при наличии в ЭВМ следующих аппаратных средств:

- ✓ автономно управляемые внешние устройства;
- ✓ развитая система прерывания программ;
- ✓ средства защиты памяти от взаимного влияния программ.

Основным недостатком режима ПМП является практически полное устранение пользователя из системы и, как следствие, отсутствие связи пользователя со своей задачей.

Режим разделения времени существенно отличается от классического мультипрограммирования, реализованного в ПМП, и является в настоящее время основным режимом функционирования операционных систем. **Главное в режиме разделения времени — это предоставление каждой задаче (или пользователю, работающему в диалоге с машиной) ресурсов ЭВМ на некоторый ограниченный интервал времени (квант).** По истечении кванта времени данная программа свертывается операционной системой, развертывается следующая по очереди программа (или подключается следующий терминал пользователя), которой предоставляются ресурсы ЭВМ, и т.д.

Режимы и дисциплины обслуживания

Порядок обслуживания заданий (заявок на работу) в операционных системах с мультипрограммированием, т.е. реализующих режимы ПМП или

РРВ, определяются принятыми в них **режимами обслуживания и дисциплинами обслуживания.**

Режимом обслуживания называется правило отбора заявок на обслуживание.

Режимы обслуживания делятся на три вида:

- ✓ режим одиночного отбора заявок:
- ✓ режим группового отбора, когда на обслуживание отбирается вся очередь заявок определенного типа;
- ✓ смешанный режим отбора, когда для одних классов заявок производится одиночный отбор, а для других групповой.

Дисциплиной обслуживания называется правило отбора заявок на обслуживание при заданном режиме обслуживания.

Для каждого из режимов обслуживания может быть применен один из следующих видов дисциплин обслуживания:

- ✓ бесприоритетное обслуживание;
- ✓ обслуживание с приоритетом;
- ✓ обслуживание по расписанию.

Разновидности дисциплины бесприоритетного обслуживания:

ОПП — обслуживание в порядке поступления («первый пришел — первый обслужен», FIFO);

ООП — обслуживание в обратном порядке («первый пришел — последний обслужен», LIFO);

ОСП — обслуживание в случайном порядке.

При бесприоритетном обслуживании считается, что все заявки имеют равное право на обслуживание.

Если требуется, чтобы заявки некоторого типа имели преимущества перед другими на их обслуживание операционной системой, то применяется дисциплина обслуживания с приоритетами:

ДОП — дисциплина обслуживания с относительными приоритетами, когда приоритет заявки влияет только на ее место в очереди заявок на обслуживание;

ДАП — дисциплина с абсолютными приоритетами, когда высоко приоритетная заявка получает преимущества не только перед заявками, стоящими в очереди, но и перед заявкой, получающей обслуживание;

ДСП — дисциплина со смешанными приоритетами, при которой к од-

ним группам заявок применяются относительные приоритеты, а к другим — абсолютные;

ДДП — дисциплина обслуживания с динамическими приоритетами, когда значение приоритетов заявок может изменяться (расти) по мере их нахождения в очереди, обеспечивая тем самым первоочередное обслуживание заявок, долго находящихся в системе.

Дисциплина обслуживания по расписанию обеспечивает заданный пользователем порядок обработки заданий независимо от очередности их поступления в систему. Она применяется в тех случаях, когда результаты решения одной задачи являются входными данными для другой.

Контрольные вопросы:

1. Пояснить, что такое режим обработки данных, охарактеризовать известные режимы.
2. Пояснить, что такое режим обслуживания, перечислить виды режимов обслуживания
3. Пояснить, что такое дисциплины обслуживания, перечислить виды дисциплин обслуживания

Лекция 8
Основные принципы построения ОС

1. Частотный принцип. Для действий, которые часто встречаются при работе с ОС, обеспечиваются условия их быстрого выполнения. *Основан на выделении в алгоритмах программ, а в обрабатываемых массивах действий и данных по частоте использования. Действия и данные, которые часто используются, располагаются в операционной памяти, для обеспечения наиболее быстрого доступа. Основным средством такого доступа является организация многоуровневого планирования. На уровень долгосрочного планирования выносятся редкие и длинные операции управления деятельностью системы. К краткосрочному планированию подвергаются часто используемые и короткие операции. Система инициирует или прерывает исполнение программ, предоставляет или забирает динамически требуемые ресурсы, и прежде всего центральный процессор и память.*

2. Принцип модульности. Модуль - это функционально законченный элемент системы, выполненный в соответствии с принятыми межмодульными интерфейсами. Модуль по определению предполагает возможность замены его на любой другой при наличии соответствующих интерфейсов. Чаще всего при построении ОС разделение на модули происходит по функциональному признаку. Важное значение при построении ОС имеют привилегированные, повторно входимые и реентерабельные модули. Привилегированные модули функционируют в привилегированном режиме, при котором отключается система прерываний, и никакие внешние события не могут нарушить последовательность вычислений. Реентерабельные модули допускают повторное многократное прерывание исполнения и повторный запуск из других задач. Для этого обеспечивается сохранение промежуточных вычислений и возврат к ним с прерванной точки. Повторно входимые модули допускают многократное параллельное использование, однако не допускают прерываний. Они состоят из привилегированных блоков и повторное обращение к ним возможно после завершения какого-либо из этих блоков. Принцип модульности отражает технологические и эксплуатационные свойства системы. Максимальный эффект от использования достигается, если принцип распространяется и на ОС, и на прикладные программы, и на аппаратуру.

3. Принцип функциональной избирательности. Этот принцип подразумевает выделение некоторых модулей, которые должны постоянно находиться в оперативной памяти для повышения производительности вы-

числений. Эту часть ОС называют **ядром**. С одной стороны, чем больше модулей в ОЗУ, тем выше скорость выполнения операций. С другой стороны, объем памяти, занимаемой ядром, не должен быть слишком большим, поскольку в противном случае обработка прикладных задач будет низкоэффективной. В состав ядра включают модули по управлению прерываниями, модули для обеспечения мультизадачности и передачи управления между процессами, модули по распределению памяти и т.д.

4. Принцип генерируемости ОС. Этот принцип определяет такой способ организации архитектуры ядра ОС, который позволял бы настраивать его, исходя из конкретной конфигурации вычислительного комплекса и круга решаемых задач. *Эта процедура выполняется редко, перед достаточно протяженным периодом эксплуатации ОС. Процесс генерации осуществляется с помощью специальной программы-генератора и соответствующего входного языка. В результате генерации получается полная версия ОС, представляющая собой совокупность системных наборов модулей и данных. Принцип модульности существенно упрощает генерацию. Наиболее ярко этот принцип используется в ОС Linux, которая позволяет не только генерировать ядро ОС, но указывать состав подгружаемых, т.н. транзитных модулей. В остальных ОС конфигурирование выполняется в процессе инсталляции.*

5. Принцип функциональной избыточности. Принцип учитывает возможность проведения одной и той же операции различными средствами. В состав ОС могут входить несколько разных мониторов, управляющих тем или иным видом ресурса, несколько систем управления файлами и т.д. *Это позволяет быстро и достаточно адекватно адаптировать ОС к определенной конфигурации вычислительной системы, обеспечить максимально эффективную загрузку технических средств при решении конкретного класса задач и получить при этом максимальную производительность.*

6. Принцип умолчания. Применяется для облегчения организации связи с системами, как на стадии генерации, так и при работе с системой. Принцип основан на хранении в системе некоторых базовых описаний, структур процесса, модулей, конфигураций оборудования и данных, определяющих прогнозируемые объемы требуемой памяти, времени счета программы, потребности во внешних устройствах, которые характеризуют пользовательские программы и условия их выполнения. *Эту информацию пользовательская система использует в качестве заданной, если она не будет за-*

данна или сознательно не конкретизирована. В целом применение этого принципа позволяет сократить число параметров устанавливаемых пользователем, когда он работает с системой.

7. Принцип перемещаемости. Предусматривает построение модулей, исполнение которых не зависит от места расположения в оперативной памяти. Настройка текста модуля в соответствии с его расположением в памяти осуществляется либо специальными механизмами, либо по мере ее выполнения. Настройка заключается в определении фактических адресов, используемых в адресных частях команды, и определяется применяемым способом адресации и алгоритмом распределения оперативной памяти, принятой для данной ОС. Она может быть распределена и на пользовательские программы.

8. Принцип виртуализации. Принцип позволяет представить структуру системы в виде определенного набора планировщиков процессов и распределителей ресурсов (мониторов), используя единую централизованную схему. *Концепция виртуальности выражается в понятии виртуальной машины. Любая ОС фактически скрывает от пользователя реальные аппаратные и иные ресурсы, заменяя их некоторой абстракцией. В результате пользователи видят и используют виртуальную машину как достаточно абстрактное устройство, способное воспринимать их программы, выполнять их и выдавать результат. Пользователю совершенно не интересна реальная конфигурация вычислительной системы и способы эффективного использования ее компонентов. Он работает в терминах используемого им языка и представленных ему виртуальной машиной ресурсов. Для нескольких параллельных процессов создается иллюзия одновременного использования того, что одновременно в реальной системе существовать не может. Виртуальная машина может воспроизводить и реальную архитектуру, однако элементы архитектуры выступают с новыми, либо улучшенными, характеристиками, зачастую упрощающими работу с системой. Идеальная, с точки зрения пользователя, машина должна иметь:*

- ✓ *единообразную по логике работы виртуальную память практически неограниченного объема;*
- ✓ *произвольное количество виртуальных процессоров, способных функционировать параллельно и взаимодействовать во время работы;*
- ✓ *произвольное количество виртуальных внешних устройств, способных получать доступ к памяти виртуальной машины после-*

довательно или параллельно, синхронно или асинхронно. Объемы информации не ограничиваются.

Чем больше виртуальная машина, реализуемая ОС, приближена к идеальной, т.е. чем больше ее архитектурно-логические характеристики отличны от реальных, тем большая степень виртуальности достигнута. ОС строится как иерархия вложенных друг в друга виртуальных машин. Нижним уровнем программ является аппаратные средства машин. Следующим уровнем уже является программным, который совместно с нижним уровнем обеспечивает достижение машиной новых свойств. Каждый новый уровень дает возможность расширять функции возможности по обработке данных и позволяет достаточно просто производить доступ к низшим уровням. Применение метода иерархического упорядочивания виртуальных машин наряду с достоинствами: систематичность проекта, возрастание надежности программных систем, уменьшение сроков разработки имеет проблемы. Основная из них: определение свойств и количества уровней виртуализации, определения правил внесения на каждый уровень необходимых частей ОС.

Свойства отдельных уровней абстракции (виртуализации):

1. *На каждом уровне ничего не известно о свойствах и о существовании более высоких уровней.*
2. *На каждом уровне ничего не известно о внутреннем строении других уровней. Связь между ними осуществляется только через жесткие, заранее определенные сопряжения.*
3. *Каждый уровень представляет собой группу модулей, некоторые из них являются внутренними для данного и доступны для других уровней. Имена остальных модулей известны на следующим, более высоком уровне, и представляют собой сопряжение с этим уровнем.*
4. *Каждый уровень располагает определенными ресурсами и либо скрывает от других уровней, либо представляет другим уровням их абстракции (виртуальные ресурсы).*
5. *Каждый уровень может обеспечивать некоторую абстракцию данных в системе.*
6. *Предположения, что на каждом уровне делается относительно других уровней, должны быть минимальными.*
7. *Связь между уровнями ограничена явными аргументами, передаваемыми с одного уровня на другой.*
8. *Недопустимо совместное использование несколькими уровнями глобальных данных.*

9. *Каждый уровень должен иметь более прочное и слабое сцепление с другими уровнями.*
10. *Всякая функция, выполняемая уровнем абстракции должна иметь единственный вход.*

9. Принцип независимости ПО от внешних устройств. Принцип заключается в том, что связь программы с конкретными устройствами производится не на уровне трансляции программы, а в период планирования ее использования. При работе программы с новым устройством, перекомпиляция не требуется. Принцип реализуется в подавляющем большинстве ОС.

10. Принцип совместимости. Этот принцип определяет возможность выполнения ПО, написанного для другой ОС или для более ранних версий данной ОС. Различают совместимость на уровне исполняемых файлов и на уровне исходных текстов программ. В первом случае готовую программу можно запустить на другой ОС. Для этого требуется совместимость на уровне команд микропроцессора, на уровне системных и библиотечных вызовов. Как правило, используются специально разработанные эмуляторы, позволяющие декодировать машинный код и заменить его эквивалентной последовательностью команд в терминах другого процессора. Совместимость на уровне исходных текстов требует наличия соответствующего транслятора и также совместимости на уровне системных вызовов и библиотек.

11. Принцип открытости и наращиваемости. Открытость подразумевает возможность доступа для анализа как системным специалистам, так и пользователям. Наращиваемость подразумевает возможность введения в состав ОС новых модулей и модификации существующих. Построение ОС по принципу клиент-сервер с использованием микроядерной структуры обеспечивает широкие возможности по наращиваемости. В этом случае ОС строится как совокупность привилегированной управляющей программы и непривилегированных услуг-серверов. Основная часть остается неизменной, тогда как серверы могут быть легко заменены или добавлены.

12. Принцип мобильности (переносимости). Подразумевает возможность перенесения ОС с аппаратной платформы одного типа на платформу другого типа. При разработке переносимой ОС следуют следующим правилам: большая часть ОС пишется на языке, который имеет трансляторы на всех платформах, предназначенных для использования. Это язык высокого уровня, как правило, С. Программа на ассемблере в общем случае не являет-

ся переносимой. Далее, минимизируют или исключают те фрагменты кода, которые непосредственно взаимодействуют с аппаратными ресурсами. Аппаратно-зависимый код изолируется в нескольких хорошо локализуемых модулях.

13. Принцип безопасности. Подразумевает защиту ресурсов одного пользователя от другого, а также предотвращения захвата всех системных ресурсов одним пользователем, включая и защиту от несанкционированного доступа. Согласно стандарту NCSC (National Computer Security Center) 1985 года, т.н. Оранжевой книге, системы подразделяются на 7 категорий: D, C1, C2, B1, B2, B3, A1, где A является классом с максимальной защитой. **Большинство современных ОС отвечают требованиям уровня C2. Он обеспечивает:**

- ✓ средства секретного входа, позволяющие идентифицировать пользователя путем ввода уникального имени и пароля при входе в систему;
- ✓ избирательный контроль доступа, позволяющий владельцу ресурса определить, кто имеет доступ к ресурсу и его права;
- ✓ средства учета и наблюдения (аудита), обеспечивающие возможность обнаружения и фиксации событий, связанных с безопасностью системы и доступом к системным ресурсам;
- ✓ защита памяти, подразумевающая инициализацию перед повторным использованием.

На этом уровне система не защищена от ошибок пользователя, но его действия легко отслеживаются по журналу. Системы уровня B распределяют пользователей по категориям, присваивая определенный рейтинг защиты, и предоставляя доступ к данным только в соответствии с этим рейтингом. Уровень A требует выполнения формального, математически обоснованного доказательства соответствия системы определенным критериям безопасности. На уровне A управляющие безопасностью механизмы занимают до 90% процессорного времени. В ОС реализуется несколько подходов для обеспечения защиты. Одним из них является двухконтекстность работы процессора, т.е. в каждый момент времени процессор может выполнить либо программу из состава ОС, либо прикладную или служебную программу, не входящую в состав ОС. Для того, чтобы гарантировать невозможность непосредственного доступа к любому разделяемому ресурсу со стороны пользовательских и служебных программ, в состав машинных команд вводятся специальные привилегированные команды, управляющие распределением и использова-

нием ресурсов. Эти команды разрешается выполнять только ОС. Контроль за их выполнением производится аппаратно. При попытке выполнить такую команду возникает прерывание, и процессор переводится в привилегированный режим. Для реализации принципа защиты используется механизм защиты данных и текста программ, находящихся в ОЗУ. Самым распространенным подходом при этом является контекстная защита. Для программ и пользователей выделяется определенный участок памяти, и выход за его пределы приводит к прерыванию по защите. Механизм контроля реализуется аппаратным способом на основе ограниченных регистров или ключей памяти. Применяются различные способы защиты хранения данных в файлах. Самый простой способ защиты - парольный.

Контрольные вопросы:

1. Перечислить принципы построения ОС
2. Охарактеризовать один или несколько принципов по своему выбору
3. Охарактеризовать один или несколько принципов по выбору преподавателя

Лекция 9
Структура MS DOS

ОС MS DOS имеет модульную структуру.

Модуль – унифицированная самостоятельная функциональная часть, имеющая законченное оформление и средства сопряжения с другими функциональными узлами и модулями.

В операционную систему MS DOS входят следующие основные модули:

1. Базовая система ввода – вывода (**BIOS**);
2. Блок начальной загрузки (**Boot Record**);
3. Модуль расширения BIOS (**IO.SIS**);
4. Модуль обработки прерываний (**MS DOS.SYS**);
5. Командный процессор (**COMMAND.COM**);
6. Файлы-драйверы, которые после их загрузки в память обеспечивают работу таких устройств, как мышь, CD-ROM и др.
7. Утилиты ОС, выполняющие различные сервисные функции (форматирование дисков и др.).

Модульная структура MS-DOS значительно облегчает ее модификацию, т.е. она открыта для наращивания своих возможностей.

Уровни вложенности элементов MS-DOS:

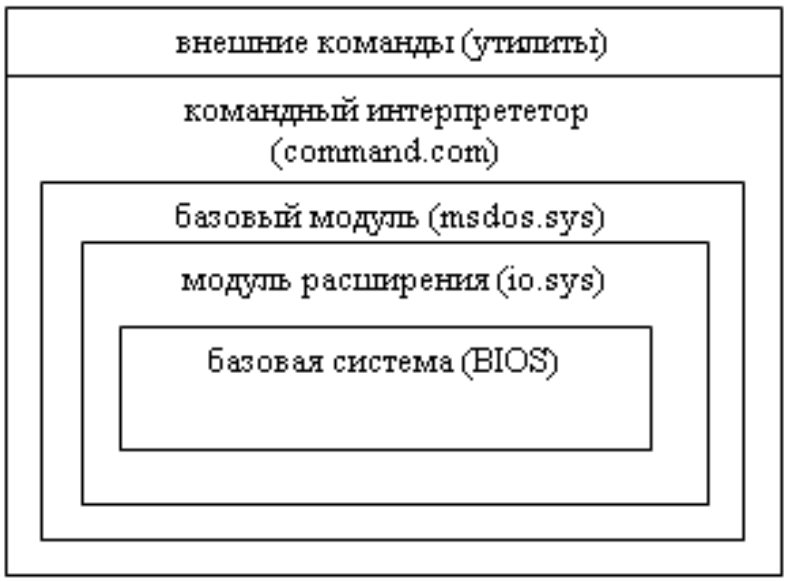

Рис. 11. Уровни вложенности

На внутреннем уровне находятся программы, управляющие аппаратурой, а **на внешнем** - средства для организации диалога с пользователями.

Основная часть MS-DOS - промежуточные уровни, которые управляют файловой системой, выполнением и взаимодействием программ, использованием памяти.

Многие операционные системы успешно работают на различных аппаратных платформах без существенных изменений в своем составе. Во многом это объясняется тем, что, несмотря на различия в деталях, средства аппаратной поддержки ОС большинства компьютеров приобрели сегодня много типовых черт, а именно эти средства в первую очередь влияют на работу компонентов операционной системы. В результате **в ОС можно выделить достаточно компактный слой машинно-зависимых компонентов ядра и сделать остальные слои ОС общими для разных аппаратных платформ.**

Структура MS DOS отображена на рисунке:

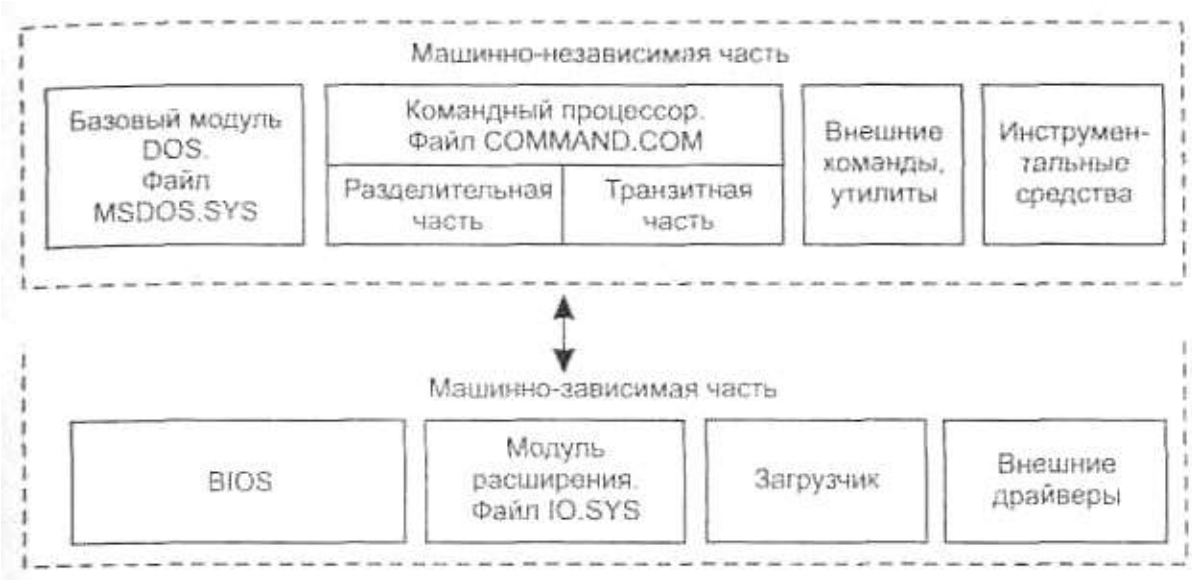

Рис. 12 Структура MS DOS

Машинно-зависимые компоненты ОС – модули, в которых отражается специфика аппаратной платформы компьютера; в идеале этот слой полностью экранирует вышележащие слои ядра от особенностей аппаратуры

Машинно-зависимые свойства операционных систем:

- ✓ **прерывания** *(это процедуры, которые компьютер вызывает для выполнения определенной задачи; или Прерывание - временное прекращение процесса, такого как выполнение программы*

вычислительной машины, вызванное событием, внешним по отношению к этому процессу, и совершенное таким образом, что процесс может быть продолжен)

- ✓ **планирование процессов** *(процесс - это система действий, реализующая определенную функцию в вычислительной системе)*
- ✓ **обслуживание ввода-вывода**
- ✓ **управление реальной и виртуальной памятью**

Одна и та же операционная система не может без каких-либо изменений устанавливаться на компьютерах, отличающихся типом процессора или/и способом организации всей аппаратуры. В модулях ядра ОС не могут не отразиться такие особенности аппаратной платформы, как количество типов прерываний и формат таблицы ссылок на процедуры обработки прерываний, состав регистров общего назначения и системных регистров, состояние которых нужно сохранять в контексте процесса, особенности подключения внешних устройств и многие другие.

Однако опыт разработки операционных систем показывает: ядро можно спроектировать таким образом, что только часть модулей будут машинно-зависимыми, а остальные не будут зависеть от особенностей аппаратной платформы. В хорошо структурированном ядре **машинно-зависимые модули локализованы и образуют программный слой, естественно примыкающий к слою аппаратуры,** как это и показано на рис. **Такая локализация машинно-зависимых модулей существенно упрощает перенос операционной системы на другую аппаратную платформу.**

Объем машинно-зависимых компонентов ОС зависит от того, насколько велики отличия в аппаратных платформах, для которых разрабатывается ОС.

Машинно-независимые свойства операционных систем:

- ✓ **работа с файлами**
- ✓ **управление заданиями**
- ✓ **распределение ресурсов** *(Под управлением ресурсами в ОС понимается распределение ресурсов системы между различными задачами и процессами, одновременно функционирующими в ней)*
- ✓ **защита**

Рассмотрим подробнее некоторые программные модули MS DOS

Командный процессор DOS обрабатывает команды, вводимые пользователем. Командный процессор находится в дисковом файле **COMMAND.COM** на диске, с которого загружается операционная система.

Некоторые команды пользователя, например ***type, dir*** или ***copy***, командный процессор выполняет сам. Такие команды называются **внутренними** или **встроенными**.

Для выполнения остальных (**внешних**) команд пользователя командный процессор ищет на дисках программу с соответствующим именем и, если находит ее, загружает в память и передает ей управление. По окончании работы программы командный процессор удаляет программу из памяти и выводит сообщение о готовности к выполнению команд (приглашение DOS).

Функции:

- ✓ Прием команд с клавиатуры или из bat-файлов и их выполнение;
- ✓ Выполнение команд файла **AUTOEXEC.BAT** при загрузке MS DOS
- ✓ Загрузка в оперативную память и запуск на выполнение прикладных программ в среде MS DOS

Командный процессор состоит из 3 частей:

1. **Резидентной** – она размещается в ОП сразу после MSDOS.SYS, включает процедуры обслуживания некоторых прерываний, процедуры обработки стандартных ошибок MS DOS, процедуры загрузки транзитной части командного процессора;
2. **Инициализирующей –** в ОП она следует сразу же за резидентной частью, во время загрузки ОС ей передается управление, она выполняет файл **AUTOEXEC.BAT** и некоторые другие действия. Эта часть командного процессора стирается из ОП первой же загруженной программой;
3. **Транзитной –** загружается в старшие адреса ОЗУ; обрабатывает все внутренние команды, команды с клавиатуры и из bat-файлов; выдает системную подсказку MS DOS, загружает в ОП программы и передает им управление (транзитный модуль может вытесняться из ОП на диск прикладной программой, если

ей для работы не хватает памяти. После окончания работы такой программы транзитный модуль вновь восстанавливается в ОП на прежнем месте).

Файл автозапуска программ при загрузке ОС (AUTOEXEC.BAT) – текстовый файл, содержащий дополнительную настроечную информацию. MS DOS выполняет этот файл автоматически, сразу после выполнения CONFIG.SYS.

Инструментальные средства MS DOS:

- ✓ Система программирования MS DOS QBASIC;
- ✓ Текстовый редактор MS DOS EDITOR;
- ✓ Отладчик DEBUG для тестирования и отлаживания исполняемых файлов

Размещение MS DOS в оперативной памяти приведено на рисунке:

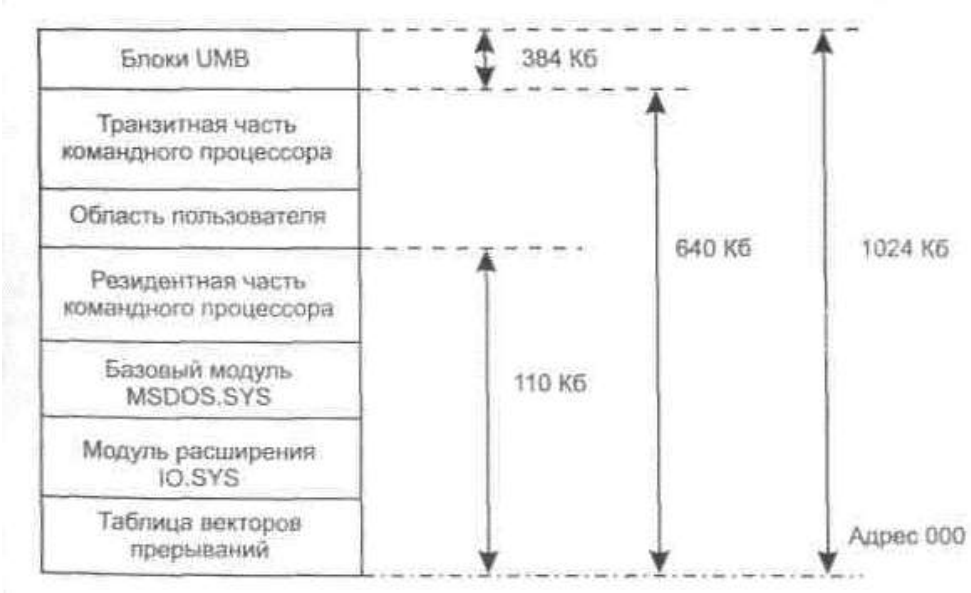

Рис. 13 Размещение MS DOS в оперативной памяти

Контрольные вопросы:

1. Нарисовать структуру ОС MS DOS
2. Перечислить машинно-зависимые свойства ОС
3. Перечислить машинно-независимые свойства ОС
4. Перечислить и охарактеризовать составные части Command.com
5. Указать назначение MSDOS.SYS

Лекция 10
Архитектура ОС Windows 95/98/ME

Windows 95 — 32-разрядная ОС со встроенной поддержкой сетевых функций. Эта ОС обеспечивает полную совместимость с программами, рассчитанными на MS DOS и Windows 3.x.

Архитектура **Windows 95** изображена на рис.

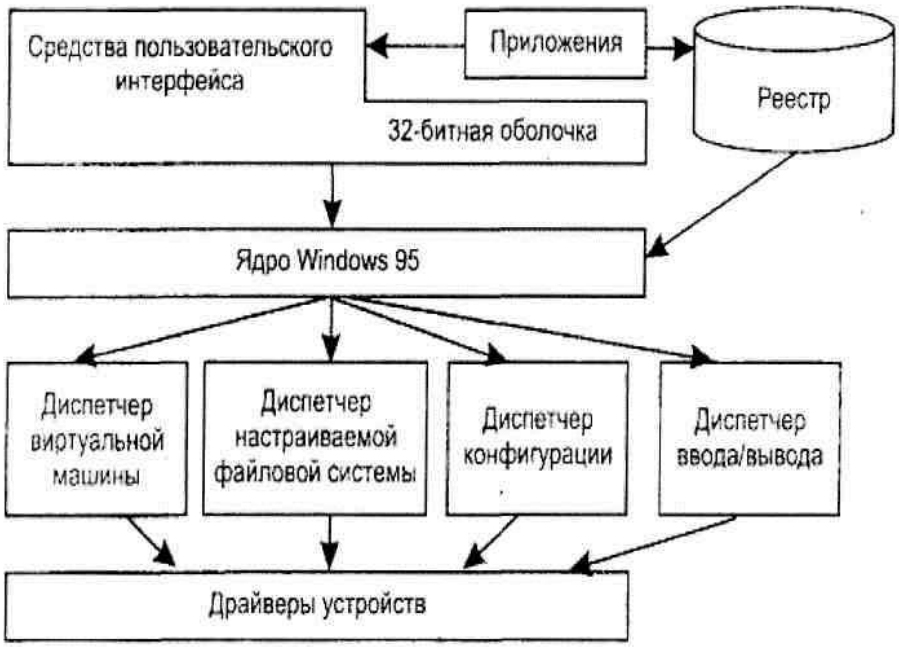

Рис. 14 Архитектура Windows 95/98

Реестр

Центральная инструментальная БД в Windows 95/98 называется **реестром** (Registry). **Основное назначение реестра — централизованное хранение информации о параметрах конфигурации системы.** С помощью реестра можно распределить ресурсы одного компьютера между несколькими пользователями и создать на нем несколько конфигураций.

Реестр – это иерархическая база данных, в которой централизованно хранится вся информация об аппаратных средствах, о конкретных приложениях **Windows95/98** и о настройках пользователя интерфейсной части ОС.

Драйверы устройств

В Windows 95/98 среди драйверов устройств различают **универсальные драйверы и мини-драйверы.**

Универсальный драйвер включает большую часть кода, необходимого конкретному классу устройств (например, принтеру или модему) для «общения» с соответствующими компонентами ОС (например, подсистемами

печати и связи).

Мини-драйвер — сравнительно небольшой и простой драйвер, содержащий какие-либо дополнительные инструкции, необходимые для управления определенным устройством, принадлежащим данному классу.

Во многих случаях универсальные драйверы реализуют практически все функции, которые необходимы для управления операциями ввода-вывода при обмене данными с периферийным устройством, и иметь дополнительный мини-драйвер не требуется.

Драйвер виртуального устройства (Virtual Device Driver, VxD) — 32-битный драйвер защищенного режима, управляющий каким-либо системным ресурсом (аппаратным или программным) и позволяющий использовать этот ресурс одновременно более чем одному приложению, **x** — подразумевает тип конкретного виртуального устройства (VDD — дисплея, VTD — таймера, VPD — принтера и т. д.). VxD — драйверы загружаются динамически.

Диспетчер конфигурации

Для поддержки функциональных возможностей технологии **Plug and Play** в архитектуре **Windows 95** включен **диспетчер конфигурации,** управляющий процессом конфигурирования системы. **Диспетчер конфигурации дает гарантию того, что каждое устройство сможет пользоваться линиями аппаратных прерываний (IRQ), адресами портов ввода/вывода и прочими ресурсами без конфликтов с другими устройствами.** При изменениях в составе оборудования диспетчер отслеживает эти изменения и при необходимости управляет процессом перенастройки оборудования, после чего уведомляет об этом приложения.

Диспетчер виртуальной машины

Диспетчер виртуальной машины (VMM – Virtual Machine Manager) выделяет ресурсы каждому приложению и системному процессу, выполняемому на компьютере. **Виртуальная машина — это некая среда в памяти, которая кажется приложению отдельным компьютером с теми же ресурсами, что и у физического компьютера.**

Диспетчер виртуальной машины включает в себя:

- ✓ планировщик процессов,
- ✓ блок подкачки страниц памяти
- ✓ интерфейс защищенного режима MS DOS.

Планировщик процессов — компонент, отвечающий за выделение системных ресурсов приложениям и другим выполняемым на компьютере про-

цессам, а также за распределение процессорного времени, что позволяет одновременно выполнять несколько процессов.

В **Windows 95** применяются **два метода для одновременного выполнения процессов:**

- ✓ кооперативная многозадачность (cooperative)
- ✓ вытесняющая многозадачность (preemptive multitasking).

Кооперативная многозадачность (используется в Windows 3.x) предусматривает, что процесс просматривает очередь сообщений и передает управление другим выполняемым приложениям (следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам). Программы, слишком редко проверяющие свою очередь, «захватывают» все процессорное время, не давая переключиться в другие приложения.

Вытесняющая многозадачность — ОС сама «решает», у какой программы отобрать управление и какой передать.

Каждой программе выделяется адресное пространство 4 Гбайта (верхние 2 Гбайта резервируются системой, нижние 2 Гбайта отдаются программе). При этом все виртуальное пространство разбивается на равные блоки или страницы.

Подкачка страниц по запросу — метод, при котором код и данные перемещаются из физической памяти в страничные файлы на диске и обратно по необходимости. Блок подкачки страниц памяти проектирует виртуальные адреса с адресного пространства процесса на физические страницы памяти компьютера. При этом физическая память линейной адресации памяти позволяет использовать все 4 Гбайта адресной памяти для 32-битных приложений.

Файл подкачки в **Windows 95/98** может теперь занимать фрагментированные области на жестком диске без специального снижения производительности и размещается даже на сжатом диске.

Хотя большинство MS DOS - программ хорошо работает под управлением Windows 95/98, какая-то часть таких программ требует монопольного доступа к системным ресурсам. В этом случае **диспетчер виртуальной машины** создает **отдельную операционную среду,** называемую **сеансом MS DOS**. В этом режиме системные ресурсы недоступны другим приложениям и процессам.

Диспетчер настраиваемой файловой системы

В **Windows 95/98** доступ к дисковым и переадресованным (через сеть) устройствам осуществляется через **диспетчер настраиваемой файловой системы IFS.** Он также отвечает за арбитраж доступа к устройствам файловой системы и их компонентам.

Этот диспетчер взаимодействует со следующими драйверами:

- ✓ 32-битный VFAT драйвер — для файловых операций на дисковых устройствах (VFAT-система Windows 9x похожа на файловую систему FAT с добавленными средствами обработки длинных имен файлов);
- ✓ 32-битный драйвер CDFS — для файловых операций на CD-ROM;
- ✓ 32-битные редиректоры для подключения к серверам Windows NT, Server, Novell NetWare и другим.

Диспетчер ввода/вывода

Диспетчер ввода/вывода — VxD-драйвер, обеспечивающий сервис для файловой системы и драйверов, а также отвечает за поддержку очередности запросов на файловый сервис и маршрутизацию запросов соответствующим драйверам. Он загружает и инициализирует драйверы устройств защищенного режима и обеспечивает сервис операций ввода/вывода.

Ядро Windows 95

Ядро Windows 95 состоит из трех компонентов: **User, Kernel** и **GDI,** каждый из которых включает две динамически подключаемые библиотеки (DDL): одну 32-битную и одну 16-битную, обеспечивающие сервис для выполняемых приложений.

Компонент ядра User управляет:

- ✓ вводом с клавиатуры, от мыши и других координатных устройств,
- ✓ выводом через интерфейс пользователя.
- ✓ кроме того, он управляет взаимодействием со звуковым драйвером, таймером и коммуникативными портами.

Kernel обеспечивает базовые функциональные возможности ОС, в том числе поддержку файлового ввода/вывода, управление виртуальной памятью и планирование задач. Кроме того, в момент запуска программы он загружает ее EXE- и DLL-файлы. **Kernel** отвечает за обработку прерываний — это обработка событий, возникающих при выполнении программы и требующих прервать в ней параллельный поток управления.

GDI (General Device Interface — интерфейс графического устройства)

— это графическая система, управляющая всем, что появляется на экране дисплея, и поддерживающая графический вывод на принтер и другие устройства. **Windows 95** отличается новой 32-битной оболочкой — интерфейс пользователя базируется на средствах **Windows Explorer (Проводник).** Все приложения и инструментальные средства способны пользоваться стандартными элементами управления, предлагаемыми оболочкой (например, стандартными файловыми окнами, списками и т. д.).

Windows 95/98 поддерживает 32- и 16-битные Windows-приложения, а также программы MS DOS.

Приложения **Win32** работают на основе алгоритма вытесняющей многозадачности в отдельных адресных пространствах.

Все приложения **Winl6** выполняются как единый процесс в общем адресном пространстве на основе алгоритма невытесняющей многозадачности. Библиотеки динамической компоновки **USER, USER32, GDI, GDI32, KERNEL и KERNEL32**, которые предоставляют системные сервисы всем приложениям, загружаются в системную VM и отображаются в адресные пространства каждого прикладного процесса.

DOS - программы работают в режиме вытесняющей многозадачности.

Контрольные вопросы:

1. Нарисовать структуру ОС Windows 95/98/ME
2. Дать определение понятию «реестр», указать назначение реестра Windows
3. Перечислить и охарактеризовать типы драйверов
4. Указать назначение диспетчера конфигурации
5. Перечислить и охарактеризовать составные части ядра ОС Windows 95/98/ME

Лекция 11
Архитектурные модули Windows NT/2000/XP/7

Windows NT была разработана группой разработчиков под руководством Дейва Катлера (**начиная с 1993 года).**

Windows NT - 32-разрядная ОС с приоритетной многозадачностью, в состав которой входят средства обеспечения безопасности и развитый сетевой сервис.

Многозада́чность *(англ. multitasking) — свойство операционной системы или среды программирования обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких процессов (одновременное выполнение нескольких программ, переключение с одной программы на другую, управление приоритетами выполняемых программ).*

Вытесняющая многозадачность *— ОС сама «решает», у какой программы отобрать управление и какой передать.*

Архитектура Windows NT создавалась заново, но она совместима с другими ОС Windows (сохранены основные принципы построения системы и функции), поддерживает существующие файловые системы, различные приложения.

Windows NT представляет собой модульную операционную систему, которая состоит из отдельных взаимосвязанных относительно простых модулей.

При обсуждении архитектуры ОС **Windows NT** постоянно используются понятия «режим пользователя» и «режим ядра», поэтому стоит определить, что это значит. Начнем с обсуждения разницы между пользовательским режимом и режимом ядра (user mode/kernel mode).

Пользовательский режим – наименее привилегированный режим, поддерживаемый NT; он не имеет прямого доступа к оборудованию и у него ограниченный доступ к памяти.

Режим ядра – привилегированный режим. Те части NT, которые исполняются в режиме ядра, такие как драйверы устройств и подсистемы типа **Диспетчера Виртуальной Памяти**, имеют прямой доступ ко всей аппаратуре и памяти.

Различия в работе программ пользовательского режима и режима ядра поддерживаются аппаратными средствами компьютера (а именно – процессором).

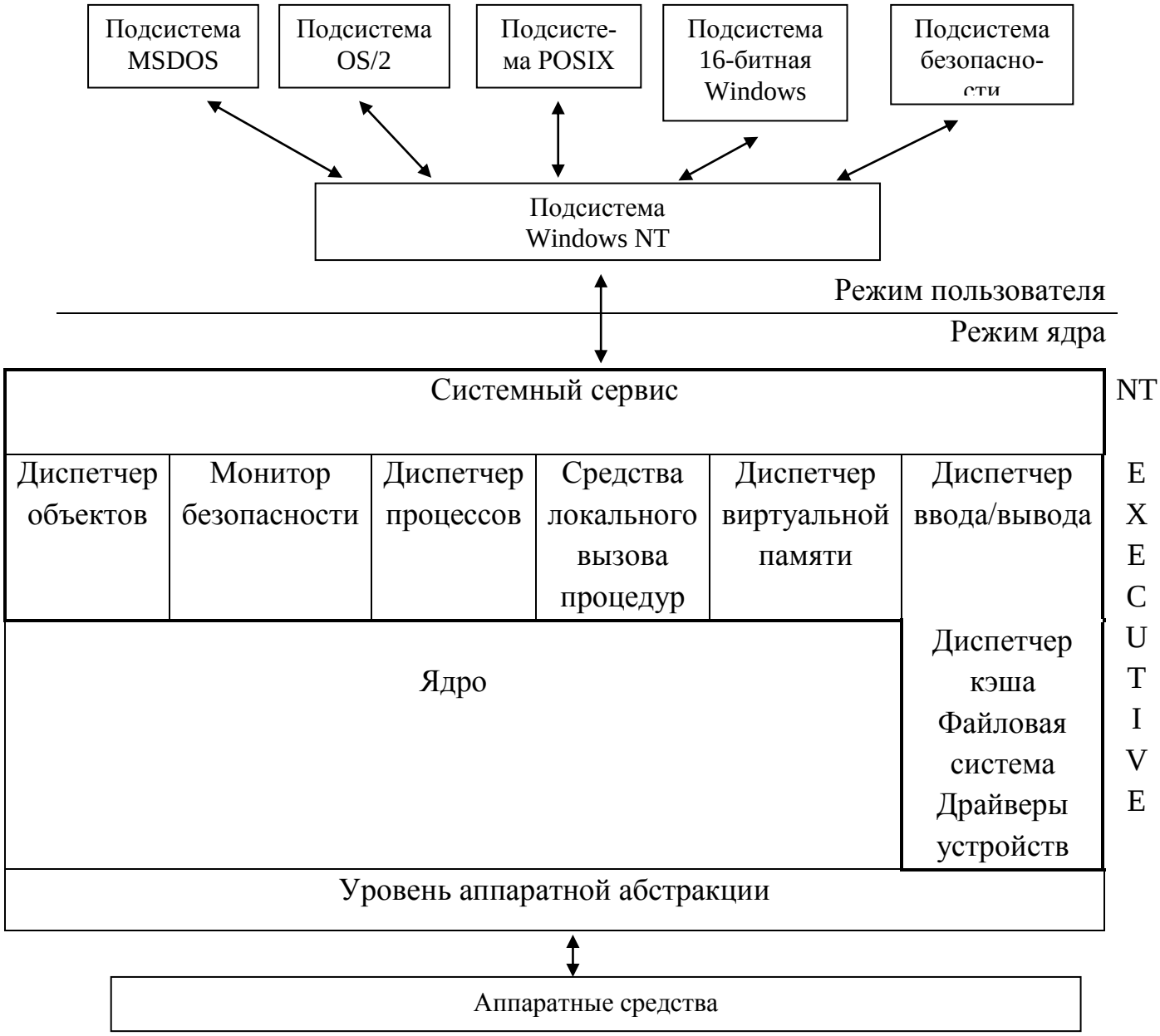

Рис. 15. Модульная структура Windows NT

Основными модулями Windows NT являются (перечислены в порядке следования от нижнего уровня архитектуры к верхнему):

- ✓ уровень аппаратных абстракций HAL (Hardware Abstraction Layer),
- ✓ ядро (Kernel),
- ✓ исполняющая система (Executive),
- ✓ защищенные подсистемы (protected subsystems) - вторую часть

Windows NT, работающую в режиме пользователя составляют **серверы** — так называемые **защищенные подсистемы**. **Серверы Windows NT называются защищенными подсистемами, так как каждый из них выполняется в отдельном процессе, память которого отделена от других процессов системой управления виртуальной памятью NT Executive.** Так как подсистемы автоматически не могут совместно использовать память, они общаются друг с другом посредством посылки сообщений. Сообщения могут передаваться как между клиентом и сервером, так и между двумя серверами. Все сообщения проходят через исполнительную часть Windows NT. Ядро Windows NT планирует нити защищенных подсистем точно так же, как и нити обычных прикладных процессов.

Поддержку защищенных подсистем обеспечивает исполняющая часть — Windows NT Executive, которая работает в пространстве ядра и никогда не сбрасывается на диск.

- ✓ подсистемы среды (environment subsystems).

Уровень аппаратных абстракций

Уровень аппаратных абстракций обеспечивает независимость остальной части операционной системы от конкретных аппаратных особенностей. Подобный подход позволяет обеспечить легкую переносимость Windows NT с одной аппаратной платформы на другую.

Уровень аппаратных абстракций (HAL) представляет собой создаваемый производителями аппаратных средств слой программного обеспечения, который скрывает (или абстрагирует) особенности и различия аппаратуры от верхних уровней операционной системы. Таким образом, благодаря так называемому HAL - фильтру, различные аппаратные средства выглядят аналогично с точки зрения операционной системы; снимается необходимость специальной настройки операционной системы под используемое оборудование.

HAL концентрирует в одном месте большую часть машинно-зависимых процедур. Он представляет собой низкоуровневый интерфейс, лежащий между исполняющей системой и аппаратными средствами компьютера. HAL располагается между NT Executive и аппаратным обеспечением и скрывает от системы такие детали, как контроллеры прерываний, интерфейсы ввода/вывода и механизмы взаимодействия между процессорами. Такое решение позволяет легко переносить Windows

NT с одной платформы на другую, заменяя только слой HAL.

При создании уровня аппаратных абстракций ставилась задача подготовки процедур, которые позволяли бы единственному драйверу конкретного устройства поддерживать функционирование этого устройства для всех платформ. **HAL ориентирован на большое число разновидностей аппаратных платформ** с однопроцессорной архитектурой; таким образом для каждого из аппаратных вариантов не требуется отдельной версии операционной системы.

Процедуры HAL *называются как средствами операционной системы (включая ядро), так и драйверами устройств.* **При работе с драйверами устройств уровень аппаратных абстракций обеспечивает поддержку различных технологий ввода-вывода** (вместо традиционной ориентации на одну аппаратную реализацию или требующей значительных затрат адаптации под каждую новую аппаратную платформу).

Ядро

Ядро является основой модульного строения системы и координирует выполнение большинства базовых операций Windows NT.

Ядро отвечает за:

- ✓ планирование выполнения потоков,
- ✓ синхронизацию работы нескольких процессоров,
- ✓ обработку аппаратных прерываний и исключительных ситуаций.

Ядро (Kernel) работает в тесном контакте с уровнем аппаратных абстракций. Этот модуль, в первую очередь, занимается планированием действий процессора. В случае, если компьютер содержит несколько процессоров, ядро синхронизирует их работу с целью достижения максимальной производительности системы.

Ядро осуществляет **диспетчеризацию потоков** (threads — нитей управления, иногда называются подзадачами, ответвлениями), которые являются основными объектами в планируемой системе.

Потоки определяются в контексте процесса; процесс включает адресное пространство, набор доступных процессу объектов и совокупность выполняемых в контексте процесса потоков управления. Объектами являются управляемые операционной системой ресурсы.

Ядро производит диспетчеризацию потоков управления таким образом, чтобы максимально загрузить процессоры системы и обеспечить первоочередную обработку потоков с более высоким приоритетом.

*Всего существует **32 значения приоритетов,** которые сгруппированы в два класса: **real-time** и **variable**. Подобный подход позволяет достичь максимальной эффективности операционной системы.*

Всего существует 32 уровня приоритетов — от 0 до 31. Распределение приоритетов между выполняющимися процессами и потоками осуществляется по следующим правилам:

Low — 4 (низкий приоритет);

BelowNormal — ниже среднего;

Normal — 8 (нормальный приоритет);

AboveNormal — выше среднего;

High — 16 (высокий приоритет);

RealTime — 24 (приоритет реального времени).

Собственно исполняемыми элементами процесса являются потоки. Поток получает базовый приоритет от своего процесса, а фактическое значение приоритета присваивается потоку операционной системой. Те потоки, которые выполняются на *переднем плане* (foreground), получают приращение приоритета относительно базового.

У потоков, выполняемых в *фоновом режиме* (background), приоритет уменьшается. По умолчанию все задачи запускаются с нормальным приоритетом. Обычный пользователь может изменить приоритет задачи вплоть до высокого. Приоритет реального времени может присвоить только администратор.

Ядро управляет двумя типами объектов.

***Объекты диспетчеризации** (dispatcher objects) характеризуются сигнальным состоянием (signaled или nonsignaled) и управляют **диспетчеризацией и синхронизацией системных операций**. Эти объекты включают события, мутанты, мутэксы, семафоры, потоки управления и таймеры (events, mutants, mutexes, semaphores, threads, timers).*

***Управляющие объекты** (control objects) используются для операций управления ядра, но не воздействуют на диспетчеризацию или синхронизацию.*

Управляющие объекты включают в себя асинхронные вызовы процедур, прерывания, уведомления и состояния источника питания, процессы и профили (asynhcronous procedure calls, in-terupts, power notifies, power statuses, processes, profiles).

Исполняющая система

Исполняющая система (Executive), в состав которой входит ядро и уровень аппаратных абстракций HAL, **обеспечивает общий сервис системы,** который могут использовать все подсистемы среды.

Каждая группа сервиса находится под управлением одной из отдельных составляющих исполняющей системы:

- ✓ диспетчера объектов (Object Manager)l
- ✓ диспетчера виртуальной памяти (Virtual Memory Manager);
- ✓ диспетчера процессов (Process Manager)
- ✓ средства вызова локальных процедур (Local Procedure Call Facility);
- ✓ диспетчера ввода-вывода (I/O Manager);
- ✓ монитора безопасности (Security Reference Monitor).

Верхний уровень исполняющей системы называется системным сервисом (System Services). Показанный на рис. системный сервис представляет собой интерфейс между подсистемами среды пользовательского режима и привилегированным режимом (режимом ядра).

Монитор безопасности

В многозадачной операционной системе, каковой является Windows NT, приложения совместно используют ряд ресурсов системы, включая память компьютера, устройства ввода-вывода, файлы и процессор(ы) системы. **Windows NT включает набор компонентов безопасности, которые гарантируют, что приложения не смогут обратиться к этим ресурсам без соответствующего разрешения.**

Монитор безопасности совместно с процессором входа в систему (Logon) и защищенными подсистемами **реализует модель безопасности Windows NT.**

Монитор безопасности отвечает за проведение в жизнь политики проверки правильности доступа и контроля, определенной локальной подсистемой безопасности. Монитор безопасности обеспечивает услуги по подтверждению доступа к объектам, проверке привилегий пользователя и генерации сообщений как для привилегированного режима, так и для режима пользователя. Монитор безопасности, подобно другим частям операционной системы, выполняется в привилегированном режиме.

Процесс входа в систему в Windows NT предусматривает обязательный вход в систему безопасности для идентификации пользователя. Каждый пользователь должен иметь бюджет и использовать пароль для обращения к этому бюджету.

Прежде чем пользователь сможет обратиться к любому ресурсу компьютера из Windows NT, он должен войти в систему через процесс входа в систему для того, чтобы подсистема безопасности могла распознать имя пользователя и пароль. Только после успешного установления подлинности монитор безопасности выполняет процедуру проверки правильности доступа для определения права пользователя на обращение к этому объекту.

Защищенность ресурсов — одна из особенностей, предоставляемая моделью безопасности. Задачи не могут обращаться к чужим ресурсам (типа памяти) иначе, чем через применение специальных механизмов совместного использования.

Windows NT также предоставляет средства контроля, которые позволяют администратору фиксировать действия пользователей.

Диспетчер кэша

Архитектура ввода-вывода содержит единственный **диспетчер кэша** (Cache Manager), который осуществляет кэширование для всей системы ввода-вывода.

Кэширование (Caching) — метод, используемый файловой системой для увеличения эффективности.

Вместо непосредственной записи и считывания с диска, часто используемые файлы временно сохраняются в кэш-памяти; таким образом, работа с этими файлами выполняется в памяти. Операции с данными, находящимися в памяти, производятся значительно быстрее операций с данными на диске.

Диспетчер кэша обеспечивает службу кэширования для всех файловых систем и сетевых компонентов, функционирующих под управлением диспетчера ввода-вывода. В зависимости от объема доступной оперативной памяти диспетчер кэша может динамически увеличивать или уменьшать размер кэш-памяти.

Драйверы файловой системы

В архитектуре ввода-вывода Windows NT управление драйверами файловой системы осуществляет **диспетчер ввода-вывода.** Windows NT допускает использование множества файловых систем:

- ✓ существующие файловые системы типа FAT.

- ✓ HPFS (для обеспечения совместимости снизу вверх с операционными системами MS-DOS, Windows 3.x и OS/2)
- ✓ NTFS — новую файловую систему, разработанную специально для использования с Windows NT. NTFS обеспечивает ряд возможностей, включая средства восстановления файловой системы, поддержку Unicode, длинных имен файлов и поддержку для POSIX.

Архитектура ввода-вывода Windows NT не только поддерживает традиционные файловые системы, но и обеспечивает **функционирование сетевых серверов в качестве драйверов файловой системы.** С точки зрения диспетчера ввода-вывода, нет разницы между работой с файлом, размещенным на удаленном компьютере сети, и работой с файлом на локальном жестком диске.

Сетевые драйверы

Следующим типом драйверов, присутствующих в качестве компонента в архитектуре ввода-вывода, являются **сетевые драйверы.** Windows NT включает интегрированные возможности работы с сетями и поддержку для распределенных приложений. Серверы функционируют как драйверы файловой системы и выполняются на уровне интерфейса поставщика или ниже.

Драйверы транспортного протокола общаются с серверами через уровень, называемый интерфейсом транспортного драйвера (TDI — Transport Driver Interface).

Windows NT включает следующие транспортные средства:

- ✓ *протокол управления передачей - межсетевой протокол TCP/IP, который обеспечивает возможность работы с широким диапазоном существующих сетей;*
- ✓ *NBF, потомок расширенного интерфейса пользователя NetBIOS (NetBEUI), который обеспечивает совместимость с существующими локальными вычислительными сетями на базе LAN Manager, LAN Server и MS-Net;*
- ✓ *управление передачей данных (DLC — Data Link Control), которое обеспечивает интерфейс для доступа к мэйнфрэймам и подключенным к сети принтерам;*
- ✓ *NWLink, реализация IPX/SPX, обеспечивающая связь с Nowell NetWare.*

В нижней части сетевой архитектуры находится драйвер платы се-

тевого адаптера.Windows NT в настоящее время поддерживает драйверы устройств, выполненные в соответствии со спецификацией NDIS (Network Device Inerface Specification) версии 3.0. NDIS предоставляет гибкую среду обмена данными между транспортными протоколами и сетевыми адаптерами. NDIS 3.0 позволяет отдельному компьютеру иметь несколько установленных в нем плат сетевых адаптеров. В свою очередь, каждая плата сетевого адаптера может поддерживать несколько транспортных протоколов для доступа к различным типам сетевых станций.

Подкомпоненты исполняющей системы, такие, как диспетчер ввода-вывода и диспетчер процессов, используют ядро для синхронизации действий.

*Они также взаимодействуют с ядром для более высоких уровней абстракции, называемых **объектами ядра;** некоторые из этих объектов экспортируются внутри пользовательских вызовов интерфейса прикладных программ (API).*

Подсистемы среды представляют собой защищенные серверы пользовательского режима (user-mode), которые обеспечивают выполнение и поддержку приложений, разработанных для различного операционного окружения (различных операционных систем). Примером подсистем среды могут служить подсистемы Win32 и OS/2.

Защищенные подсистемы Windows NT, в отличие от исполняющей системы и уровня абстрагирования от аппаратных средств, работают в пользовательском режиме, что закрывает им низкоуровневый доступ к аппаратным средствам и данным компьютера. Но операционная система от этого становится более надежной и безопасной, ведь **подсистемы могут обращаться к компьютеру только через посредника — исполняющую систему, которая контролирует работу всей машины.**

Контрольные вопросы:

1. Нарисовать структуру ОС Windows NT/2000/XP
2. Указать назначение HAL
3. Указать состав и назначение исполняющей системы
4. Указать назначение монитора безопасности
5. Указать назначение ядра ОС

Лекция 12
Краткий обзор современных операционных систем

Последние версии ОС Windows

Windows Vista (имеющая кодовое название Longhorn) — операционная система семейства Microsoft Windows NT, линейки операционных систем, используемых на пользовательских персональных компьютерах. Windows Vista, как и Windows XP, — исключительно клиентская система. Microsoft также выпустила серверную версию Windows Vista — Windows Server 2008.

В Windows Vista обновлена подсистема управления памятью и вводом-выводом. Новой функциональностью также является «Гибридный спящий режим», или режим «гибернации», при использовании которого содержимое оперативной памяти дополнительно записывается на HDD, но и из памяти также не удаляется. В результате если подача энергии не прекращалась, то компьютер восстанавливает свою работу, пользуясь информацией из ОЗУ. Если питание компьютера выключалось, операционная система использует сохраненную на HDD копию ОЗУ и загружает информацию с нее (аналог спящего режима). Режим реализован благодаря так называемым «файлам гибернации», которые занимают объем на жестком диске, равный объему установленной на компьютере оперативной памяти. Возможно пользовательское удаление этих файлов с утратой функции гибернации. При этом восстановление этих файлов без особых затруднений возможно путем вызова специальных команд из командной строки.

Финальная версия Windows Vista представлена в вариантах для 32- и для 64-разрядных процессоров.

Операционная система Windows 7

Windows 7 имеет шесть редакций:

1. Начальная (Starter);
2. Домашняя базовая (Home Basic);
3. Домашняя расширенная (Home Premium);
4. Профессиональная (Professional);
5. Корпоративная (Enterprise; для России не выпускается);
6. Максимальная (Ultimate).

Начальная редакция (Windows 7 Starter) не будет включать функциональной части для проигрывания H.264, AAC, MPEG-2. Домашняя базовая — предназначена исключительно для выпуска в развивающихся странах (в том

числе в России), в ней нет интерфейса Windows Aero с функциями Peek, Shake и предпросмотр$_a$ в панели задач, общего доступа к подключению в Интернет и некоторых других функций. Также в ней есть те же ограничения на просмотр, что и в начальной редакции. В профессиональной и максимальной версиях существует поддержка XP Mode (на некоторых процессорах).

Все 32-битные версии, кроме Начальной, поддерживают до 4 Гб ОЗУ. Начальная поддерживает до 2 Гб в 32 и 64-битвой версии. Однако на практике пользователю часто доступен меньший объем. Поддержка более крупных объёмов памяти доступна только при переходе на 64-битную версию. 64-битные версии поддерживают до 8 Гб (Домашняя базовая), до 16 Гб (Домашняя расширенная) и до 192 Гб памяти во всех остальных редакциях.

В Windows 7 есть возможность отключения или включения браузера Internet Explorer и проигрывателя Windows Media Player. Так же ОС обладает поддержкой multitouch-мониторов для ноутбуков.

Функция ReadyBoost позволяет использовать флэш-накопитель как дополнительную кэш-память для ускорения работы системы. Меню Пуск в Windows 7 стало короче и лишилось иконок.

В ОС также встроено около 120 фоновых рисунков, уникальных для каждой страны и языковой версии. Так, русская версия включает тему «Россия» с шестью уникальными обоями высокого разрешения. Все версии включают 50 новых шрифтов. Существующие шрифты доработаны для корректного отображения всех символов **Windows 7** — первая версия Windows, которая включает больше шрифтов для отображения нелатинских символов, чем для отображения латинских. Панель управления шрифтами также подверглась улучшению — по умолчанию в ней будут отображаться только те шрифты, раскладка для которых установлена в системе. Панель поиска Instant Search теперь распознаёт больше языков. К примеру, распознаются русские падежи, склонения, род, единственное и множественное числа.

Дополнительным преимуществом Windows 7 можно считать более тесную интеграцию с производителями драйверов. Большинство из них определяются автоматически, при этом в 90% случаев сохраняется обратная совместимость с драйверами для Windows Vista.

В Windows 7 используется DirectX 11 и Windows Media Player 12. Последний получил новый интерфейс и стал поистине «всеядным», в отличие от предшественника, которому требовалось большое количество кодеков для воспроизведения. Однако он не может воспроизводить лицензионные Blu-Ray диски с видео, но имеет возможность считывать и записывать на них данные. Windows Media Player получил новый прозрачный интерфейс и теперь им можно управлять с панели задач.

Несмотря на то что Центр мобильности Windows не претерпел значительных изменений со времён Windows Vista, Windows 7 работает дольше предшественницы на ноутбуках и потребляет меньше энергии, особенно при воспроизведении DVD.

В Windows 7 реализована более гибкая настройка User Account Control (UAC), которая, в отличие от Windows Vista, имеет ещё два промежуточных состояния между режимами «Всегда уведомлять» и «Никогда не уведомлять» — «Уведомлять, только при попытках программ внести изменения в компьютер» (положение по умолчанию), «Уведомлять, только при попытках программ внести изменения в компьютер (не затемнять рабочий стол)». Стоит заметить, что в отличие от Vista, затемнение происходит, только если программа активна и находится на переднем плане. Если вы совершили клик в момент открытия UAC и деактивировали программу, затем нения может и не произойти.

Внесены изменения в технологию шифрования BitLocker, добавлена функция шифрования съёмных носителей BitLocker to go, позволяющая шифровать съёмные носители.

Улучшения коснулись и брандмауэра Windows - вернулась функция уведомления пользователя о блокировке программы, которая пытается получить доступ к сети.

С помощью групповой политики и функции AppLocker можно запретить запуск определенных приложений.

Функция DirectAccess позволяет устанавливать безопасное соединение с сервером в фоновом режиме. Также DirectAccess может применять групповые политики до входа пользователя в систему.

Windows 7 использует sandbox-режим. Весь неуправляемый код запускается в среде (песочнице), в которой операционная система ограничивает доступ программы к аппаратной части компьютера и сети. Доступ к низкоуровневым сокетам, равно как и прямой доступ к файловой системе, уровню абстракции от оборудования (HAL), полному доступу к адресу памяти, запрещён. Весь доступ ко внешним приложениям, файлам и протоколам регулируется операционной системой.

В интерфейс Windows Aero добавлена новая функция Aero Shake, позволяющая свернуть все неактивные приложения движением мыши. Для ее активации достаточно захватить заголовок окна и немного «потрясти» влево-вправо.

Функция Aero Peek позволяет отображать уменьшенные копии окон при наведении мыши на значок панели задач, переключаться между окнами приложения простым кликом по значку, перетаскивать и фиксировать на па-

нели задач различные окна и приложения, просматривать рабочий стол одним наведением в специальную область экрана и многое другое.

Аналогично функции Shake функция Aero Snap позволяет движением мыши разворачивать окно на пол-экрана, весь экран или только по вертикальной оси.

Краткий обзор современных операционных систем семейства UNIX

Семейство операционных систем UNIX

UNIX является исключительно удачным примером реализации простой мультипрограммной и многопользовательской операционной системы. В свое время она проектировалась как инструментальная система для разработки программного обеспечения. Своей уникальностью система UNIX обязана во многом тому обстоятельству, что была, по сути, создана всего двумя разработчиками, которые делали ее исключительно для себя и первое время использовали на мини-ЭВМ с очень скромными вычислительными ресурсами. Первая версия этой системы занимала всего около 12 Кбайт и могла работать на компьютерах с очень небольшим объемом оперативной памяти. Поскольку при создании второй версии UNIX разработчики отказались от языка ассемблера и специально придумали язык высокого уровня, на котором можно было бы писать не только системные, но и прикладные программы (речь идет о языке С), то и сама система UNIX, и приложения, выполняющиеся в ней, стали легко переносимыми (мобильными). Компилятор с языка С для всех оттранслированных программ дает реентерабельный и разделяемый код, что позволяет эффективно использовать имеющиеся в системе ресурсы.

Общая характеристика и особенности архитектуры

Первой целью при разработке этой системы было стремление сохранить простоту и обойтись минимальным количеством функций. Все реальные сложности оставлялись пользовательским программам.

Второй целью была общность. Одни и те же методы и механизмы должны были использоваться во многих случаях:

- ✓ обращение к файлам, устройствам ввода-вывода и буферам межпроцессных сообщений выполняются с помощью одних и тех же примитивов;

- ✓ одни и те же механизмы именования, присвоения альтернативных имен и защиты от несанкционированного доступа применяются и к файлам с данными, и к каталогам, и к устройствам;
- ✓ одни и те же механизмы работают в отношении программно и аппаратно инициируемых прерываний.

Третья цель заключалась в том, чтобы сложные задачи можно было решать, комбинируя существующие небольшие программы, а не разрабатывая их заново.

Наконец, **четвертая цель состояла в создании мультитерминальной операционной системы с эффективными механизмами разделения не только процессорного времени, но и всех остальных ресурсов.** В мультитерминальной операционной системе на одно из первых мест по значимости выходят вопросы защиты одних вычислительных процессов от вмешательства других вычислительных процессов. Причем для реализации третьей цели необходимо было создать механизмы полноценного обмена данными между программными модулями, из которых предполагалось составлять конечные программы.

Операционная система UNIX обладает простым, но очень мощным командным языком и независимой от устройств файловой системой. Важным, хотя и простым с позиций реализации такой возможности, является тот факт, что система UNIX предоставляет пользователям средства направления выхода одной программы непосредственно на вход другой. В результате достигается четвертая цель — большие программные системы можно создавать путем композиции имеющихся небольших программ, а не путем написания новых, что в большинстве случаев упрощает задачу. UNIX -системы существуют уже 30 лет, и к настоящему времени имеется чрезвычайно большой набор легко переносимых из системы в систему отлично отлаженных и проверенных временем приложений.

В число системных и прикладных программ, поставляемых с UNIX -системами, входят редакторы текстов, программируемые интерпретаторы командного языка, компиляторы с нескольких популярных языков программирования, включая C, C++, ассемблер, PERL, FORTRAN и многие другие, компоновщики (редакторы межпрограммных связей), отладчики, многочисленные библиотеки системных и пользовательских программ, средства сортировки и ведения баз данных, многочисленные административные и обслуживающие программы. Для абсолютного большинства всех этих программ имеется документация, в том числе исходные тексты программ (как правило,

хорошо комментированные). Кроме того, описания и документация по большей части доступны пользователям в интерактивном режиме. Используется иерархическая файловая система с полной защитой, работа со съемными томами, обеспечивается независимость от устройств.

Центральной частью UNIX -систем является ядро (kernel).

Оно состоит из большого количества модулей и с точки зрения архитектуры считается **монолитным**. Однако **в ядре всегда можно выделить три основные подсистемы:**

- ✓ управления процессами,
- ✓ управления файлами,
- ✓ управления операциями ввода-вывода между центральной частью и периферийными устройствами.

Подсистема управления процессами организует выполнение и диспетчеризацию процессов, их синхронизацию и разнообразное межпроцессное взаимодействие. **Важнейшая функция подсистемы управления процессами — это распределение оперативной памяти и (для современных систем) организация виртуальной памяти.** Подсистема управления файлами тесно связана и с подсистемой управления процессами, и с драйверами. Ядро может быть перекомпилировано с учетом конкретного состава устройств компьютера и решаемых задач. Не все драйверы могут быть включены в состав ядра, часть из них может вызываться из ядра. Более того, очень большое количество системных функций выполняется системными программными модулями, не входящими непосредственно в ядро, но вызываемых из ядра. Основные системные функции, которые должно выполнять ядро совместно с остальными системными модулями, строго стандартизированы. За счет этого во многом достигается переносимость кода между разными версиями UNIX и абсолютно различным аппаратным обеспечением.

Виртуальная машина

Система UNIX многопользовательская. Каждому пользователю после регистрации (входа в систему) предоставляется виртуальный компьютер, в котором есть все необходимые ресурсы: процессор (процессорное время выделяется на основе круговой, или карусельной, диспетчеризации и с использованием динамических приоритетов, что позволяет обеспечить равенство в обслуживании), оперативная память, устройства, файлы. Текущее состояние такого виртуального компьютера, предоставляемого пользователю, называется *образом.*

Можно сказать, что процесс — это выполнение образа. Образ процесса состоит:

- ✓ из образа памяти;
- ✓ значений общих регистров процессора;
- ✓ состояния открытых файлов;
- ✓ текущего каталога файлов;
- ✓ другой информации.

Суперпользователь

Очевидно, что администратор системы, который тоже является зарегистрированным пользователем, чтобы управлять всей системой, должен обладать существенно большими, чем обычные пользователи, привилегиями. В операционных системах UNIX эта задача решается путем выделения единственного нулевого значения UID. **Пользователь с таким значением UID называется *суперпользователем* (superuser) и обозначается словом *root* (корень).** Он имеет неограниченные права на доступ к любому файлу и на выполнение любой программы. Кроме того, такой пользователь имеет возможность полного контроля над системой. Он может остановить ее и даже разрушить. По этой причине не рекомендуется работать под этой учетной записью. Администратор должен создать себе обычную учетную запись простого пользователя, а для выполнения действий, связанных с административными полномочиями, рекомендуется использовать команду **su**. Команда **su** запрашивает у пользователя пароль суперпользователя, и, если он указан правильно, операционная система переводит сеанс пользователя в режим работы суперпользователя. После выполнения необходимых действий, требующих привилегий суперпользователя, следует выполнить команду exit, которая и вернет администратору статус простого пользователя.

Еще одним важным отличием суперпользователя от обычного пользователя операционной системы UNIX является то, что на суперпользователя не распространяются ограничения на используемые ресурсы. Для обычных пользователей устанавливаются такие ограничения, как максимальный размер файла, максимальное число сегментов разделяемой памяти, максимально допустимое пространство на диске и т. д. Суперпользователь может изменять эти ограничения для других пользователей, но на него они не действуют.

Интерфейс пользователя

Традиционный способ взаимодействия пользователя с системой UNIX основывается на командных языках. После входа пользователя в систему для него запускается один из командных интерпретаторов (в зависимости от па-

раметров, сохраняемых в файле /etc/passwd). Обычно в системе поддерживается несколько командных интерпретаторов с похожими, но различающимися своими возможностями командными языками. Общее название для любого командного интерпретатора ОС UNIX — ***оболочка*** **(shell),** поскольку любой интерпретатор представляет внешнее окружение ядра системы. По умолчанию в системах **Linux** командным интерпретатором является **bash**. В принципе он может быть заменен другим, но практически никто этого не делает.

Вызванный командный интерпретатор выдает приглашение на ввод пользователем командной строки, которая может содержать простую команду, конвейер команд или последовательность команд. После выполнения очередной командной строки и выдачи на экран терминала или в файл соответствующих результатов интерпретатор команд снова выдает приглашение на ввод командной строки, и так до тех пор, пока пользователь не завершит свой сеанс работы и не выйдет из системы.

Командные языки, используемые в UNIX, достаточно просты, чтобы новые пользователи могли быстро начать работать, и достаточно мощны, чтобы можно было использовать их для написания сложных программ.

Поскольку в настоящее время все большее распространение получают графические интерфейсы, в операционных системах семейства UNIX стали все чаще работать в X-Window. X-Window — это графический интерфейс, позволяющий пользователям взаимодействовать со своими вычислениями и с системой в графическом режиме. В отличие от систем Windows компании Microsoft, графический интерфейс для UNIX -систем не является основным, в системе можно работать и без него.

Командная строка состоит из имени команды (а именно имени выполняемого файла), за которым следует список аргументов, разделенных пробелами. Оболочка разбивает командную строку на компоненты. Указанный в команде файл загружается, и ему обеспечивается доступ к заданным в команде аргументам.

Любой командный язык оболочки фактически состоит из трех частей:

- ✓ служебных конструкций, позволяющих манипулировать текстовыми строками и строить сложные команды на основе простых команд;
- ✓ встроенных команд, выполняемых непосредственно интерпретатором командного языка;
- ✓ команд, представляемых отдельными выполняемыми файлами.

В свою очередь, набор команд последнего вида включает стандартные команды (системные утилиты, такие как vi, cc и т. д.) и команды, созданные пользователями системы.

Операционная система Linux

Linux — это современная UNIX-подобная операционная система для персональных компьютеров и рабочих станций, удовлетворяющая стандарту POSIX.

Как известно, **Linux — это свободно распространяемая версия UNIX-систем, которая первоначально разрабатывалась Линусом Торвальдсом (torvalds@kruuna.helsinki.fi) в университете Хельсинки (Финляндия).** Он предложил разрабатывать ее совместно и выдвинул условие, согласно которому исходные коды являются открытыми, любой может их использовать и изменять, но при этом обязан оставить открытым и свой код, внесенный в тот или иной модуль системы. Все компоненты системы, включая исходные тексты, распространяются с лицензией на свободное копирование и установку для неограниченного числа пользователей. Таким образом, система **Linux** была создана с помощью многих программистов и энтузиастов UNIX - систем, общающихся между собой через Интернет. Изначально система Linux создавалась как «самодельная» UNIX -подобная реализация для машин типа IBM PC с процессором **i80386**. Однако вскоре Linux стала настолько популярна и ее поддержало такое большое число компаний, что в настоящее время имеются реализации этой операционной системы практически для всех типов процессоров и компьютеров на их основе. На базе **Linux** создаются и встроенные системы, и суперкомпьютеры. Система поддерживает кластеризацию и большинство современных интерфейсов и технологий.

Linux — это полноценная многозадачная многопользовательская операционная система (точно так же, как и все другие версии UNIX). Это означает, что одновременно много пользователей могут работать на одной машине, параллельно выполняя множество программ.

Система Linux достаточно хорошо совместима с рядом стандартов для UNIX (насколько можно говорить о стандартизации UNIX) на уровне исходных текстов, включая IEEE POSIX.1, System V и BSD. Она и создавалась с расчетом на такую совместимость, Большинство свободно распространяемых через Интернет программ для UNIX может быть откомпилировано для Linux практически без особых изменений. Кроме того, все исходные тек-

сты для Linux, включая ядро, драйверы устройств, библиотеки, пользовательские программы и инструментальные средства распространяются свободно.

Linux поддерживает различные типы файловых систем для хранения данных. Некоторые файловые системы, такие как EXT2FS, были созданы специально для Linux. Поддерживаются также другие типы файловых систем, например Minix-1 и Xenix. Кроме того, реализована система управления файлами на основе FAT, позволяющая непосредственно обращаться к файлам, находящимся в разделах с этой файловой системой. Поддерживается также файловая система ISO 9660 CD-ROM для работы с дисками CD-ROM. Имеются системы управления файлами и на томах с HPFS и NTFS, правда, они работают только на чтение файлов. Созданы варианты системы управления файлами и для доступа к FAT32; эта файловая система в операционной системе Linux называется VFAT.

Как и в классических UNIX -системах, Linux имеет макроядро, которое содержит уже известные три подсистемы.

Возможности ОС Android для мобильных устройств

Операционная система Андроид на данный момент является наиболее распространённой во всём мире и предлагает пользователям массу разнообразнейшего функционала. Некоторые из вас наверняка уже знают, что в каждой из версий системы Андроид умелыми разработчиками были заложены особые функции, по умолчанию спрятанные в настройках.

1. **Ярлычки контактов телефонной книги на рабочем столе Андроид**
2. **Поиск и голосовой набор**

Вы можете обнаружить виджет «Google Поиска» на главном рабочем столе вашего Андроид устройства. Кроме того, на стандартной Google клавиатуре есть кнопка с микрофоном. Микрофон будет использоваться для записи вашей речи и преобразования её в текст, в зависимости от предварительно настроенного языка в параметрах девайса.

3. **Меню быстрых настроек**

Для того, чтобы иметь быстрый доступ к параметрам и функциональности вашего Андроид аппарата, вам потребуется лишь удерживать палец на верхней части экрана, потянуть его вниз и вы увидите следующие настройки: контроль Wi-Fi, Bluetooth, управление громкостью и вибрацией, 3G, GPRS и многое другое.

4. **Количество рабочих экранов**

Пользователи Андроид устройств фирмы Samsung могут управлять количеством рабочих столов на своём смартфоне или планшете. Для этого вам нужно удерживать палец на свободном пространстве рабочего стола до появления дополнительного меню. После этого выберите пункт «Настройки», затем «Дополнительные настройки», чтобы установить нужное вам количество рабочих столов. Существуют также некоторые приложения в Play Маркете, которые помогут вам с этим, расширяя стандартный функционал.

5. Программы в фоне

Когда пользователь закрывает приложение, на самом деле оно не завершает свою работу, а уходит в фон. Эта функция полезна в некоторых случаях, однако системные ресурсы не резиновые, и множество таких «висяков» могут привести к ухудшению производительности, особенно на старых моделях устройств.

Следует добавить, что система сама закрывает запущенные процессы, которые не используются длительное время, но чаще всего происходит перезагрузка (в устройствах Samsung), что очень раздражает.

6. Работа с ярлыками и виджетами

Ярлыки и виджеты на рабочем столе могут быть удалены, перемещены, изменены в размерах. Чтобы сделать это, вам будет достаточно удерживать на них палец пару секунд и, затем, выбрать необходимый вариант.

7. Работа с сервисами Google

При первом запуске устройства, система Андроид предлагает вам создать новый Google аккаунт или войти в существующий, если у вас таковой имеется. Когда вы создадите новый контакт или добавите новое событие в календаре, то система спросит вас, куда вы желаете сохранить эту информацию: в память телефона или в Вашем Google аккаунте.

Если вы выберите первый вариант, то информация будет сохранена в памяти устройства. В противно случае она будет синхронизирована с Вашим аккаунтом Google. Последний вариант имеет множество преимуществ, которые вы ощутите не сразу. Однако, в конце концов, вы поймёте всю важность подобного хранения своих личных данных.

8. Синхронизация закладок браузера между системами Андроид и Windows

Если Вы в качестве интернет браузера используете Google Chrome, то вся история и все закладки, сохранённые на вашем Андроид устройстве, могут быть просмотрены и загружены на вашем ПК (соответственно тоже в Google Chrome) и наоборот. Данная функциональная особенность этого браузера весьма полезна.

9. Использование флэшек для хранения данных

Это, возможно, может быть для многих пользователей очевидным фактом, однако некоторые люди будут удивлены, узнав, что вместо карты MicroSD, можно использовать флэшки, подключив их к смартфону либо планшету через специальный OTG кабель.

10. Всё для всех

Возможность делиться своими файлами, новостями и программами также может показаться вам новой особенностью систему Андроид. На данный момент практически в каждой программе есть функция расшаривания данных. Она не просто добавляет ссылку на ваши данные кому-либо, но может быть использована как быстрая возможность создать заметку. Поэтому, если вам необходимо срочно создать заметку, просто выберите шаринг, найдите программу Google Keep и ссылка на ваш текст будет моментально создана.

Если вы хотите сохранить ваш файл на google диске – просто кликните «Sharing» > «Google Drive», и выберите файл для загрузки на сервер.

11. Управление интернет трафиком Андроид приложений

Статистика интернет трафика (потяните вниз панель уведомлений, нажмите «Настройки» > «Сеть») показывает, какая из ваших программ наиболее интенсивно выходит в интернет и загружает оттуда данные. Узнать это может очень полезно, ведь любая из них может стать причиной потери денег на вашем счету. Благодаря этим настройкам вы сможете запретить выбранным приложениям доступ в интернет.

12. **Разблокировка устройства с помощью распознавания лица пользователя,** а также многое другое.

Вы можете разблокировать свой смартфон или планшет обыкновенным свайпом, пин-кодом, графическим ключом или же распознаванием лица пользователя.

Контрольные вопросы:

1. Перечислить и охарактеризовать версии ОС Windows 7
2. Охарактеризовать особенности архитектуры ОС семейства UNIX
3. Охарактеризовать особенности интерфейса пользователя в ОС семейства UNIX
4. Охарактеризовать особенности ОС Linux
5. Охарактеризовать возможности ОС Android для мобильных устройств

Лекция 13
Понятие ресурса в ОС

В общем случае, всякий потребляемый объект (независимо от формы его существования), обладающий некоторой практической ценностью для потребителя, является ресурсом.

Ресурсы по запасу выделяемых единиц ресурса бывают ***исчерпаемые*** *и* ***неисчерпаемые****. Исчерпаемость ресурса, как правило, приводит к жизненным конфликтам в среде потребителей. Для регулирования конфликтов ресурсы должны распределяться между потребителями по каким-то правилам, в наибольшей степени их удовлетворяющим.*

Одной из важнейших задач, решаемых современными ОС, является обеспечение эффективного и бесконфликтного распределения ресурсов вычислительной системы между процессами.

Ресурс вычислительной системы – средство вычислительной системы, которое может быть выделено процессу обработки данных на определенный интервал времени.

Основными ресурсами вычислительной системы являются процессоры, области основной памяти, наборы данных, периферийные устройства, программы.

Общую классификацию ресурсов можно представить в виде:

1. **По реальности существования:** физический и виртуальный;

Под физическим понимают ресурс, который реально существует и при распределении его между пользователями обладает присущими ему физическими характеристиками.

Виртуальный ресурс - это некоторая модель физического ресурса. Виртуальный ресурс не существует в том виде, в котором он проявляет себя пользователю. Как модель виртуальный ресурс реализуется в некоторой программно-аппаратной форме. В этом смысле виртуальный ресурс существует. Однако виртуальный ресурс может предоставить пользователю при работе с ним не только часть тех свойств, которые присущи объекту моделирования, т.е. физическому ресурсу, но и свойства, которые ему не присущи.

2. **По возможности расширения свойств:** эластичный и жесткий

Признак **«возможность расширения свойств»** характеризует ресурс с точки зрения возможности построения на его основе некоторого виртуального ресурса. **Физический ресурс, который допускает «виртуализацию», т.е. воспроизведение и (или) расширение своих свойств, называют эластичным.**

Жестким называется физический ресурс, который по своим внутренним свойствам не допускает виртуализацию.

3. **По степени активности:** активный и пассивный

Активный ресурс способен выполнять действия по отношению к другим ресурсам (или даже в отношении самого себя) или процессам, которые в общем случае приводят к изменению последних. **Пассивный ресурс** не обладает таким свойством.

ЦП - активный ресурс; область памяти, выделяемая по требованию - пассивный ресурс.

4. **По времени существования:** постоянный, временный

Различие ресурсов по признаку **«время существования»** обусловлено динамикой ресурсов в отношении процессов, использующих их. **Если ресурс существует в системе до момента порождения процесса и доступен для использования на всем проведении интервала существования процесса, то такой ресурс является постоянным для данного процесса.**

Временный ресурс может появляться или уничтожаться в системе динамически в течение времени существования рассматриваемого процесса. Причем создание и уничтожение может проводиться как самим процессом, так и другими процессами - системными или пользовательскими. Очевидно, что ресурсы разделяются по определенным правилам системой взаимосвязанных процессов. Поэтому ресурсы, которые являются постоянными для одних процессов, могут быть временными для других, и наоборот.

5. **По степени важности:** главный и второстепенный

Необходимость различия ресурсов по признаку **«степень важности»** обусловлена двумя причинами: обеспечение должной работоспособности и

увеличение гибкости управления процессами и распределением ресурсов. Для этого различают **главные и второстепенные ресурсы. Ресурс является главным по отношению к конкретному процессу, если без его выделения процесс принципиально не может развиваться.** К таким ресурсам относятся, прежде всего, ЦП и ОП. **Ресурсы, которые допускают некоторую альтернативу развития процесса, если они не будут выделены, называются второстепенными** (например, МЛ, МД).

6. **По структуре:** простой, составной;

Структурный признак устанавливает наличие или отсутствие у ресурса некой структуры. **Ресурс является простым, если не содержит составных элементов и рассматривается при распределении как единое целое.**

Составной ресурс характеризуется некоторой структурой. Он содержит в своем составе ряд однотипных элементов, обладающих с точки зрения пользователей, одинаковыми характеристиками. Процессам-пользователям безразлично, какой или какие из элементов среди прочих из составного ресурса будут выделяться им при удовлетворении их запросов на ресурс. Простой и составной ресурсы отличаются числом состояний. **Простой ресурс может быть либо «занят», когда он выделен для пользования какому-либо процессу, либо «свободен».**

Составной ресурс находится в состоянии «свободен», если ни один из его составных элементов не распределен для использования. Если же все элементы такого ресурса выделены для использования, то он находится в состоянии «занят». Если часть элементов ресурса распределена, а остальные (известно какие) нет, то ресурс «частично занят».

7. **По восстанавливаемости:** воспроизводимый, потребляемый

При построении механизмов распределения ресурсов на основе использования той или иной дисциплины особенно важно учитывать характер использования распределяемых ресурсов. По этому признаку учитывается и сущность ресурса, возможность в этой связи восстанавливаемости ресурса в системе после его использования. **По возможности восстанавливаемости ресурсы подразделяются на воспроизводимые и потребляемые.** Предполагаются, что в отношении каждого ресурса процесс-пользователь выполняет три типа действий: ЗАПРОС, ИСПОЛЬЗОВАНИЕ, ОСВОБОЖДЕНИЕ. **Если при распределении системой ресурса допускается многократное выпол-**

нение действий в последовательности запрос-использование-освобождение, то такой ресурс называют воспроизводимым. После возвращения он доступен для использования его другим процессом. Поэтому, если не учитывать вид изменений ресурса при каждом разовом использовании, можно считать время жизни ресурса бесконечно большим или достаточно большим, пока он не потеряет своих функциональных свойств. В отношении определенной категории ресурсов правомочно использование действий в следующем порядке: **освобождение-запрос-использование,** после чего ресурс, который в данном случае называют потребительным, изымается из сферы потребления (например, отношение производитель-потребитель).

Срок жизни **потребляемого ресурса,** определяемый периодом между выполнением действий **освобождение и использование**, конечен. В отношении процесса производителя и процесса-потребителя потребляемые ресурсы ведут себя как **временные**.

8. **По характеру использования:** параллельно используемый, последовательно используемый;

Природа ресурса и (или) используемое правило распределения ресурса обусловлены **параллельной** или **последовательной** схемой использования распределяемого между несколькими процессами ресурса. **Последовательная схема предполагает, что в отношении некоторого ресурса, который называют последовательно-используемым, допустимо строго последовательное во времени выполнение цепочек действий «запрос-исполнение-освобождение» каждым процессом-потребителем этого ресурса; для** параллельных процессов такие цепочки действий являются критическими областями и должны выполняться так, чтобы удовлетворять правилу взаимного исключения, определенному ранее. Поэтому **последовательно используемый ресурс, разделяемый несколькими параллельными процессами, чаще называют критическим ресурсом.**

Параллельная схема предполагает параллельное, т.е. одновременное, использование одного ресурса, который поэтому называют параллельно используемым более чем одним процессом. Такое использование не должно вносить каких-либо ошибок в логику развития каждого процесса (массив в памяти для чтения).

9. **По форме реализации:** мягкий, твердый

По форме реализации различают «твердые» и «мягкие» ресурсы. **Под «твердыми» понимают аппаратные компоненты машины, а также человеческие ресурсы. Все остальные виды ресурсов относятся к разряду «мягких».** Существенно разным для твердых и мягких ресурсов помимо сложности и стоимости является их подверженность сбойным или отказываемым ситуациям и последующее восстановление работоспособности. В отличие от «твердых» «мягкие» ресурсы не могут стать неработоспособными из-за усталостного отказа. В классе «мягких» ресурсов выделяют два типа: **программные и информационные**. **Если «мягкий» ресурс допускает копирование и эффект от использования ресурса-оригинала и ресурса-копии идентичен, то такой ресурс называют программным мягким ресурсом. В противном случае его следует отнести к информационному типу (это программы, файлы, массивы и т.п.). «Мягкие» информационные ресурсы либо принципиально не допускают копирование, либо допускают копирование, но оно является функцией времени.** Это различного вида потребляемые ресурсы: сообщения, сигналы прерывания, запросы к ОС на различного вида услуги, сигналы синхронизации. Такие сообщения и сигналы информационно значимы (доступны и ценны, как ресурс) только в течение некоторого конечного интервала времени. Например, если в некоторую ячейку памяти записывается периодически некоторые сообщения, то возможно копирование конкретного поступившего сообщения от момента записи его в эту ячейку до момента поступления туда нового сообщения. Последующее копирование уже дает другой результат от использования выбранного сообщения.

10. **По функциональной избыточности (по стоимости):** дорогой, дешевый.

Разделение ресурсов на **дорогие** и **дешевые** связано с реализацией принципа функциональной избыточности при распределении ресурсов. Перед пользователем стоит задача выбора - получить быстро требуемый ресурс и дорого заплатить за такую услугу, либо подождать выделения требуемого ресурса и после его использования заплатить более дешево. При наличии в системе альтернативных ресурсов вводятся и различные цены за их использование.

В терминах ОС понятие ресурс обычно используется по отношению к повторно используемым, относительно стабильным и зачастую недостающим объектам, которые могут запрашиваться, использоваться и освобождаться. Ресурсы бывают разделяемые, когда несколько процессов использует их одновременно (в один и тот же момент времени) или параллельно (используя ресурс попеременно в течение некоторого интервала времени), и неделимыми, когда ресурс может использоваться только одним процессом.

При разработке первых ОС ресурсами считались **процессорное время, память, каналы ВВ и периферийные устройства.** Позже понятие ресурса стало более универсальным и общим. К ним стали относиться и разного рода **программные и информационные ресурсы**, которые с точки зрения системы, также могут являться объектами, которые возможно распределять и управлять доступом. Понятие ресурса превратилось в абстрактную структуру с рядом атрибутов, характеризующих способы доступа к ней и ее физическое представление в системе. Кроме системных ресурсов, в это понятие стали включаться и такие объекты межпроцессного обмена, как **сообщения и синхросигналы**.

Одним из основных видов ресурсов является процессор. При этом собственно процессор как ресурс выступает лишь для многопроцессорных систем, **в однопроцессорных же системах ресурсом является процессорное время.** Его разделение производится по параллельной схеме.

Следующий вид ресурсов - память. Она может быть разделена и одновременным способом (в памяти одновременно находятся несколько процессов) и параллельным (память предоставляется процессам поочередно). **Проблема эффективного разделения оперативной памяти между процессами является одной из самых актуальных.** В общем случае, **собственно память и доступ к ней являются разными ресурсами.** Каждый из них может быть предоставлен независимо друг от друга, но для полной работы с памятью необходимы оба из них. Так, например, внешняя память может разделяться одновременно, а доступ к ней - попеременно.

Внешние устройства являются еще одним видом ресурсов. При наличии механизмов прямого доступа они могут разделяться одновременно. Если же устройство имеет только последовательный доступ, то оно не является разделяемым ресурсом, например, принтер, накопитель на магнитной ленте.

Программные модули так же являются одним из ресурсов. Однократно используемые модули могут быть правильно выполнены только один раз, в процессе работы они могут либо испортить свой код, либо исходные данные. **Такие модули являются неделимым ресурсом. Повторно используемые модули могут быть непривилегированными, привилегированными, реентерабельными и повторно входимыми.**

Данные выступают в качестве информационных ресурсов. Это либо переменные в ОЗУ, либо файлы. В случае использования данных только для чтения, они легко разделяются. В случае же разрешения процессам изменения этого вида ресурса, то проблема его разделения значительно усложняется.

Ресурсы подразделяются на выгружаемые и невыгружаемые. Выгружаемый ресурс можно безболезненно забирать у владеющего им процесса, например, память. Невыгружаемый ресурс нельзя забрать от владельца, не уничтожив результаты вычислений. Например, нельзя прервать запись компакт-диска.

Контрольные вопросы:

1. Пояснить, что такое ресурс ВС
2. Перечислить основные ресурсы ВС
3. Привести классификацию ресурсов ВС
4. Охарактеризовать классификацию ресурсов по определенному признаку по выбору преподавателя

Лекция 14
Управление процессами (заданиями, задачами)

Основными понятиями управления прохождением задач в ЭВМ являются **процесс, задача, работа, программа, ресурс, дисциплина распределения ресурса.**

Процесс — минимальный программный объект, обладающий собственными системными ресурсами (запущенная программа).

Классификация процессов

По временным характеристикам различают:

- ✓ **интерактивные** - время существования **интерактивного процесса** определяется реакцией ЭВМ на запрос обслуживания и составляет секунды.
- ✓ **пакетные процессы** - запускаются один вслед за другим и здесь время реакции минуты и даже часы.
- ✓ **процессы реального времени** - имеют гарантированное время окончания работы и время реакции порядка миллисекунд (мс).

По генеалогическому признаку различают порождающие и порожденные процессы.

По результативности различают:

- ✓ **эквивалентные**
- ✓ **тождественные**
- ✓ **равные процессы.**

Все они имеют одинаковый конечный результат, *но **эквивалентные процессы** могут реализовываться как на одном, так и на многих процессорах по одному или разным алгоритмам, т. е. они имеют разные трассы, которые определяют порядок и длительность пребывания процесса в разных состояниях. **Тождественные процессы** реализуются по одной и той же программе, но имеют разные трассы. **Одинаковые процессы** реализуются по одной программе и имеют одинаковые трассы.*

По времени развития различают процессы:

- ✓ **последовательные,**
- ✓ **параллельные**
- ✓ **комбинированные** (для последних есть точки, в которых существуют оба процесса, и точки, в которых существует только один

процесс).

По месту развития процессы делятся на **внутренние** (реализуются на центральном процессоре) и **внешние** (реализуются на внешних процессорах).

По принадлежности к операционной системе процессы бывают:

- ✓ **системные** (исполняют программу из состава операционной системы)
- ✓ **пользовательские**.

По связности различают процессы:

а. **взаимосвязанные**, которые имеют какую-то связь (пространственно-временную, управляющую, информационную);
б. **изолированные** — слабо связанные;
в. **информационно-независимые,** которые используют совместные ресурсы, но имеют собственные информационные базы;
г. **взаимодействующие** — имеют информационные связи и разделяют общие структуры данных;
д. **взаимосвязанные по ресурсам;**
е. **конкурирующие.**

Порядок взаимосвязи процессов определяется **правилами синхронизации**. Процессы могут находиться в отношении:

а. **предшествования** — один всегда находится в активном состоянии раньше, чем другой;
б. **приоритетности** — когда процесс может быть переведен в активное состояние только в том случае, если в состоянии готовности нет процессов с более высоким приоритетом, или процессор свободен, или на нем реализуется процесс с меньшим приоритетом;
в. **взаимного исключения** — в процессе используется общий критический ресурс, и процессы не могут развиваться одновременно: если один из них использует критический ресурс, то другой находится в состоянии ожидания.

Ресурс *— любая потребляемая (расходуемая) сущность.*

По запасам *ресурсы подразделяются на* ***исчерпаемые*** *и* ***неисчерпае-***

***мые**. Потребители ресурсов — процессы.*

***Ресурс** — средство вычислительной системы, которое может быть выделено процессу на определенный интервал времени.*

Основными ресурсами вычислительной системы являются:

- ✓ *процессоры;*
- ✓ *области основной памяти;*
- ✓ *наборы данных;*
- ✓ *периферийные устройства;*
- ✓ *программы*

Процессор — любое устройство в составе ЭВМ, способное автоматически выполнять допустимые для него действия (процессоры, каналы и устройства, работающие с каналами). Реализация системы управления процессами в составе ОС предъявляет определенные требования к свойствам процессоров.

Дисциплина распределения ресурса определяет порядок использования многими процессами того или иного ресурса, который в каждый момент времени может обслуживать только один процесс.

Управление процессами

Процесс — это программный модуль, выполняемый в центральном процессоре (CPU). Операционная система контролирует следующую деятельность, связанную с процессами:

- ✓ создание и удаление процессов;
- ✓ планирование процессов;
- ✓ синхронизация процессов;
- ✓ коммуникация процессов;
- ✓ разрешение тупиковых ситуаций.

Не следует смешивать понятия «процесс» и «программа». **Программа** — это план действий, а **процесс** — это собственно действие, поэтому понятие процесса включает:

- ✓ программный код;
- ✓ данные;
- ✓ содержимое стека;
- ✓ содержимое адресного и других регистров процессора.

Таким образом, для одной программы могут быть созданы несколько процессов в том случае, если с помощью одной программы в CPU выполняется несколько несовпадающих последовательностей команд. За время су-

ществования процесс многократно изменяет свое состояние.

Различают следующие состояния процесса (рис. 16):

- ✓ **новый** (процесс только что создан);
- ✓ **выполняемый** (команды программы выполняются в CPU);
- ✓ **ожидающий** (процесс ожидает завершения некоторого события, чаще всего операции ввода-вывода);
- ✓ **готовый** (процесс ожидает освобождения CPU);
- ✓ **завершенный** (процесс завершил свою работу).

Переход из одного состояния в другое не может выполняться произвольным образом. На рис. 16 приведена типовая диаграмма переходов для состояний процессов.

Рис. 16. Состояния процесса

Каждый процесс представлен в операционной системе набором данных, называемых **таблица управления процессом** (**ТУП — PCB** — process control block). В PCB процесс описывается набором значений, параметров, характеризующих его текущее состояние и используемых операционной системой для управления прохождением процесса через компьютер.

На рис. 17 схематически показано, каким образом операционная система использует PCB для переключения процессора с одного процесса на другой.

Планирование процессов

Система управления процессами обеспечивает прохождение процесса через компьютер. В зависимости от состояния процесса ему должен быть предоставлен тот или иной ресурс. Например, новый процесс необходимо разместить в основной памяти, следовательно, ему необходимо выделить часть адресного пространства. Процессу в состоянии **готовый** должно быть предоставлено процессорное время. **Выполняемый** процесс может потребо-

вать оборудование ввода-вывода и доступ к файлу.

Рис. 17. Переходы между процессами

Очереди

Распределение процессов между имеющимися ресурсами носит название планирование процессов. Одним из методов планирования процессов, ориентированных на эффективную загрузку ресурсов, является **метод очередей ресурсов. Новые процессы находятся во входной очереди, часто называемой очередью работ — заданий.**

Входная очередь располагается во внешней памяти, во входной очереди процессы ожидают освобождения ресурса — адресного пространства основной памяти.

Готовые к выполнению процессы располагаются в основной памяти и связаны **очередью готовых процессов.** Процессы в этой очереди ожидают освобождения ресурса **процессорное время.**

Процесс в состоянии ожидания завершения операции ввода-вывода находится в одной из **очередей к оборудованию** ввода-вывода.

При прохождении через компьютер процесс мигрирует между различными очередями под управлением программы, которая называется планировщик (scheduler).

Операционная система, обеспечивающая режим мультипрограммирования, обычно включает два планировщика — **долгосрочный** и **краткосрочный**. Например, в OS/360 долговременный планировщик назывался **планировщиком заданий,** а краткосрочный — **супервизором задач.**

На уровень долгосрочного планирования выносятся редкие системные действия, требующие больших затрат системных ресурсов, на уровень краткосрочного планирования — частые и более короткие процессы. На каждом

уровне существует свой объект и собственные средства управления им.

Основное различие между долгосрочным и краткосрочным планировщиками заключается в частоте запуска, например, краткосрочный планировщик может запускаться каждые 100 мс, долгосрочный — 1 раз за несколько минут.

Долгосрочный планировщик решает, какой из процессов, находящихся во входной очереди, должен быть переведен в очередь готовых процессов в случае освобождения ресурсов памяти.

Долгосрочный планировщик выбирает процесс из входной очереди с целью создания неоднородной мультипрограммной смеси. Это означает, что в очереди готовых процессов должны находиться в разной пропорции как процессы, ориентированные на ввод-вывод, так и процессы, ориентированные на преимущественную работу с CPU.

На уровне долгосрочного планирования объектом является не отдельный процесс, а некоторое объединение процессов по функциональному назначению, которое называется работой (приложением). *Каждая работа рассматривается как независимая от других работ деятельность, связанная с использованием одной или многих программ и характеризующаяся конечностью и определенностью. По мере порождения новых работ создается собственная виртуальная машина для их выполнения. Например, в ОС Windows 95 для каждого 32-разрядного приложения реализуется своя виртуальная машина. Распределение машин производится однократно в отличие от краткосрочного планирования, где процессор процессу может выделяться многократно.*

Краткосрочный планировщик решает, какой из процессов, находящихся в очереди готовых процессов, должен быть передан на выполнение в CPU.

В некоторых операционных системах ***долгосрочный планировщик*** *может отсутствовать. Например,* ***в системах разделения времени*** *(time-sharing system) каждый новый процесс сразу же помещается в основную память.*

На уровне краткосрочного планирования объектом управления являются процессы, которые выступают как потребители центрального процессора для внутренних процессов или внешнего процессора для внешних процессов. Причинами порождения процесса могут быть процессы на том же уровне или сигналы, посылаемые от долгосрочного планировщика.

Выделение процессора процессу производится многократно, с целью достижения эффекта мультипрограммирования, и такой процесс называется **диспетчеризацией**.

Взаимодействие процессов

Совместно выполняемые процессы могут быть либо **независимыми**, либо **взаимодействующими**. Взаимодействие процессов часто понимается в смысле взаимного обмена данными через общий буфер данных.

Взаимодействие процессов удобно рассматривать в схеме производитель-потребитель. Например, программа вывода на печать производит последовательность символов, которые потребляются драйвером принтера, или компилятор производит ассемблерный текст, который затем потребляется ассемблером.

Для взаимодействия процесса-производителя и процесса-потребителя создается совместный буфер, заполняемый процессом-производителем и потребляемым процессом-потребителем.

Буфер имеет фиксированные размеры и, следовательно, процессы могут находиться в состоянии ожидания, когда:

- ✓ *буфер заполнен — ожидает процесс-производитель;*
- ✓ *буфер пуст — ожидает процесс-потребитель.*

Буфер может предоставляться и поддерживаться самой ОС, например, с помощью средств межпроцессной коммуникации, либо должен быть организован прикладным программистом. При этом оба процесса используют общий участок памяти.

Взаимодействие заключается в передаче данных между процессами или совместном использовании некоторых ресурсов и обычно реализуется с помощью таких механизмов, как **транспортеры, очереди, сигналы, семафоры.**

Транспортеры (каналы). Являются средством взаимодействия родственных процессов, представляют собой **область памяти, имеющую файловую организацию,** для которой обеспечивается запись и считывание данных в транспортере. Реализуется очередь обслуживания. Порядок записи данных на транспортер неизменен, не допускается повторное считывание данных. Обмен данными происходит не непосредственно, а через транспортер. Из вызвавшего процесса задается размер транспортера. Дочерние процессы могут использовать родительский транспортер.

Очереди. Эти механизмы могут обеспечивать передачу или использование общих данных без перемещения данных, а с передачей элемента очереди, содержащего указатель данных и объем массива данных. Очередь используется вместе с механизмом общей памяти. Элемент очереди может

быть считан с уничтожением или без уничтожения этого элемента. Чтение элемента может осуществляться в соответствии с механизмом очереди или стека. Чтение элементов очереди осуществляет только создающий очереди процесс, все другие процессы могут только записать элемент в очередь.

Создающий процесс может выполнять следующие действия над очередью:

- ✓ *создание очереди;*
- ✓ *просмотр очереди;*
- ✓ *чтение очереди;*
- ✓ *закрытие очереди.*

Записывающий процесс осуществляет действия:

- ✓ *открыть очередь;*
- ✓ *записать в очередь;*
- ✓ *закрыть очередь.*

Имя очереди, которое присваивается создающим процессом, имеет вид полной спецификации файла. ***Ожидание элементов в очереди организуется с помощью семафора,*** *сигнализирующего о записи элемента в очередь.*

Для работы с очередью определены такие дополнительные функции:

- ✓ *определение количества элементов в очереди в текущий момент;*
- ✓ *очистка очереди создавшим ее процессом.*

Преимущества очереди: *передача данных по указателю без копирования, гибкое изменение порядка передачи и доступа, возможность просмотра элементов очереди без их удаления.*

Сигналы. Сигнал является механизмом передачи требования от одного процесса к другому на немедленное выполнение действия. Обработчик сигнала создается процессом и помещается в начале первого потока процесса. Является аналогом обработки прерывания. При передаче управления обработчику передается адрес возврата и тип принятого сигнала. Процесс, посылающий сигнал типа флаг, может передать дополнительную информацию обработчику сигнала.

Характер выполняемых действий при возникновении сигнала:

- ✓ обработка системной ошибки при появлении сигнала,
- ✓ блокирование сигнала,
- ✓ передача управления подпрограмме.

Семафоры. Являются механизмами передачи сообщений от одного потока к другому о наступлении некоторого события. **Различают семафоры**

системные и оперативной памяти.

Семафоры оперативной памяти *— двойное слово в памяти системы, его описатель — адрес места в памяти. Такие семафоры не создаются и не открываются, а устанавливаются в определенное состояние. Процессы, использующие семафоры оперативной памяти, должны иметь доступ к соответствующему сегменту памяти. Операционная система такие семафоры не обслуживает и не сообщает об их освобождении или захвате. При создании семафора или его открытии возвращается описатель семафора, включающий его имя. Операционная система контролирует завершение каждого процесса, владеющего системным семафором, и освобождает его для процессов.*

Если семафор свободен, то он захватывается вызывающим его процессом, если семафор занят, то вызвавший его поток переходит в режим ожидания освобождения семафора или ожидает истечения времени. Если семафор освобождается всеми использующими его процессами, то он удаляется из системы.

Управление семафором реализуется с помощью функций:

- ✓ *установки семафора с целью сигнализации;*
- ✓ *ожидания вызывающим потоком, пока семафор не будет выключен;*
- ✓ *ожидания потоком выключения одного из нескольких семафоров.*

Операционные системы используют разные термины для определения способов межпроцессного взаимодействия.

В операционных системах OS/2 и Microsoft Windows существует ***специальный механизм для взаимодействия процессов в реальном масштабе времени. Этот механизм называется DDE*** *(Dynamic Data Exchange — динамический обмен данными). Он стандартизирует процесс обмена командами, сообщениями и объектами для обработки между задачами. Наиболее распространенным процессом, для которого используется DDE, является печать.*

Другим интерфейсом для обмена данными является OLE *(Object Linking and Embedding — связывание и встраивание объектов). Этот интерфейс позволяет хранить объекты, созданные одной программой, в объектах, созданных другой программой, а также редактировать (печатать) их без нарушения целостности информации и связей.*

Одним из наиболее простых, удобных и интуитивных интерфейсов межпрограммного взаимодействия является буфер обмена *— clipboard. Буфер обмена может содержать в себе один информационный объект — фрагмент текста, рисунок и т. д. С помощью системного вызова процесс*

может получить копию информации, содержащейся в буфере обмена, или сам поместить объект в буфер, при этом старое содержимое буфера теряется. Таким образом, программы получают простой, но эффективный способ обмена информацией в процессе своей работы.

Планирование работы процессора

Краткосрочный планировщик выбирает процессы из очереди готовых процессов и передает их на выполнение в CPU. Существуют различные алгоритмы или стратегии решения этой задачи, различающиеся отношением к критериям планирования.

Известны следующие критерии, позволяющие сравнивать алгоритмы краткосрочных планировщиков:

- ✓ **утилизация CPU** (использование процессора). Утилизация CPU теоретически может находиться пределах от 0 до 100%. В реальных системах утилизация CPU колеблется **в** пределах 40% для легко загруженного CPU, 90% для тяжело загруженного CPU;
- ✓ **пропускная способность CPU.** Пропускная способность CPU может измеряться количеством процессов, которые выполняются в единицу времени;
- ✓ **время оборота** — для некоторых процессов важным критерием является полное время выполнения, т. е. интервал от момента появления процесса во входной очереди до момента его завершения. Это время названо временем оборота и включает время ожидания во входной очереди, время ожидания в очереди готовых процессов, время ожидания в очередях к оборудованию, время выполнения в процессоре и время ввода-вывода;
- ✓ **время ожидания** — под этим понимается суммарное время нахождения процесса в очереди готовых процессов;
- ✓ **время отклика** — для интерактивных программ важным показателем является время отклика или время, прошедшее от момента попадания процесса во входную очередь до момента первого обращения к терминалу.

Очевидно, что **простейшая стратегия краткосрочного планировщика должна быть направлена на максимизацию средних значений загруженности и пропускной способности, времени ожидания и времени отклика.**

*В ряде случаев используются сложные критерии, например, так называемый **минимаксный критерий,** т. е. вместо простого критерия минимум среднего времени отклика используется **минимум максимального времени***

отклика.

Стратегии планирования процессора

Первый пришел — первый обслуживается, FIFO — first come — first served (FCFS)

FCFS является наиболее простой стратегией планирования процессов и заключается в том, что процессор передается тому процессу, который раньше всех других его запросил.

Когда процесс попадает в очередь готовых процессов, ТУП (PCB) присоединяется к хвосту очереди.

Среднее время ожидания для стратегии **FCFS** часто весьма велико и зависит от порядка поступления процессов в очередь готовых процессов.

Стратегии **FCFS** присущ так называемый **«эффект конвоя».** В том случае, когда в компьютере имеется один большой процесс и несколько малых, то все процессы собираются в начале очереди готовых процессов, а затем в очереди к оборудованию. Таким образом, **«эффект конвоя»** приводит к снижению загруженности как процессора, так и периферийного оборудования.

Стратегия «наиболее короткая работа выполняется первой», SJF — Shortest Job First

Одним из методов борьбы с «эффектом конвоя» является стратегия, позволяющая процессу из очереди выполняться первым. Стратегия **SJF** снижает время ожидания очереди. Наибольшая трудность в практической реализации **SJF** заключается в невозможности заранее определить величину времени последующего обслуживания.

Поэтому стратегия **SJF часто применяется в долгосрочных планировщиках, обслуживающих пакетный режим.** В этом случае вместо величины времени последующего обслуживания используется допустимое максимальное время выполнения задания, которое программист должен специфицировать перед отправкой задания в пакет.

Приоритетное планирование

Описанные ранее стратегии могут рассматриваться как частные случаи **стратегии приоритетного планирования.** Эта стратегия предполагает, что **каждому процессу приписывается приоритет, определяющий очередность предоставления ему CPU.** Например, стратегия **FCFS** предполагает, что все процессы имеют одинаковые приоритеты, а стратегия **SJF** предполагает, что приоритет есть величина, обратная времени последующего обслу-

живания.

Обычно **приоритет — это целое положительное число, находящееся в некотором диапазоне, например, от 0 до 7 или от 0 до 1024.** Будем считать, что **чем меньше значение числа, тем выше приоритет процесса.** *Приоритеты назначаются, исходя из совокупности внутренних и внешних по отношению к операционной системе факторов.*

Внутренние факторы:

- ✓ *требования к памяти;*
- ✓ *количество открытых файлов;*
- ✓ *отношение среднего времени ввода-вывода к среднему времени использования ресурсов CPU и т. д.*

Внешние факторы:

- ✓ *важность процесса;*
- ✓ *тип и величина файлов, используемых для оплаты;*
- ✓ *отделение, выполняющее работы, и т. д.*

Внутренние факторы могут использоваться для автоматического назначения приоритетов самой операционной системой, а внешние — для принудительного, с помощью оператора.

Главный недостаток приоритетного планирования заключается в возможности блокирования на неопределенно долгое время низкоприоритетных процессов.

Известен случай, когда в 1973 г. в Массачусетском технологическом институте при остановке компьютера IBM 7094 в очереди готовых процессов были обнаружены процессы, активизированные в 1967 г. но так и не выполненные.

Для устранения отмеченного недостатка используются следующие методы: процессы, время ожидания которых превышает фиксированную величину, например 15 минут, автоматически получают единичное приращение приоритета.

«Карусельная» стратегия планирования RR-Round Robin — применяется в системах разделения времени. Определяется **небольшой отрезок времени t_k,** названный **квантом времени (10...100 мс). Очередь готовых процессов рассматривается как кольцевая.** Процессы циклически перемещаются по очереди, получая CPU на время, равное одному кванту. Новый процесс добавляется в хвост очереди. Если процесс не завершился в пределах выделенного ему кванта времени, его работа принудительно прерывается, и он перемещается в хвост очереди.

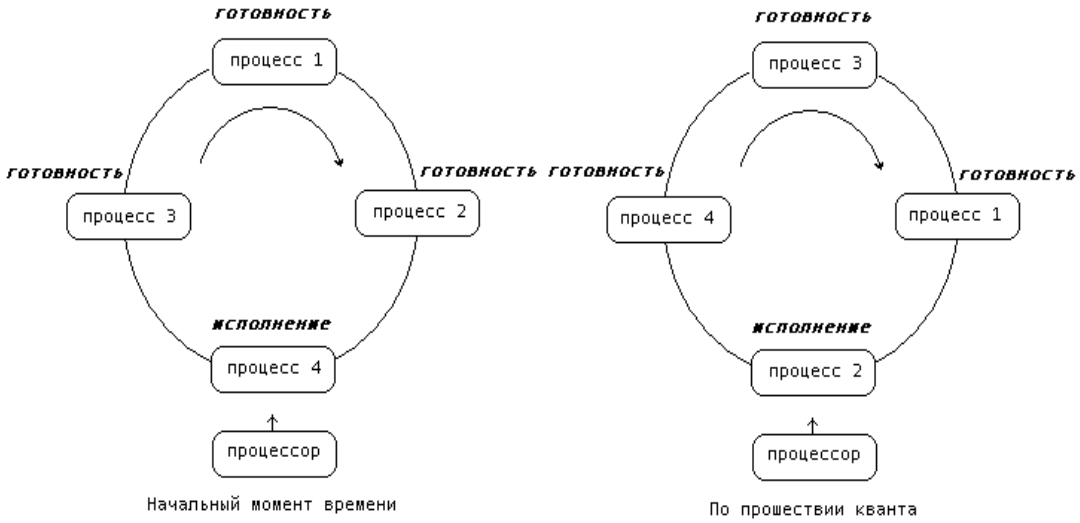

Рис 18. Процессы на карусели.

Свойства стратегии **Round Robin** сильно зависят от величины временного кванта t_k. Чем больше временной квант, тем ближе стратегия Round Robin приближается к FCFS-стратегии. При очень малых значениях временного кванта Round Robin стратегию называют **разделением процессора — processor sharing.** Теоретически это означает, что каждый из N процессов работает со своим собственным процессором, производительность процессора равна $1/N$ от производительности физического процессора.

Планирование с использованием многоуровневой очереди (Multilevel queue scheduling)

Эта стратегия разработана для ситуации, когда процессы могут быть легко классифицированы на несколько групп, например, часто процессы разделяют на две группы:

- ✓ интерактивные (процессы переднего плана)
- ✓ пакетные (фоновые).

Интерактивные и пакетные процессы имеют различные требования к краткосрочному планировщику, например по отношению ко времени отклика.

Стратегия многоуровневой очереди разделяет очередь готовых процессов на несколько очередей, в каждой из которых находятся процессы с одинаковыми свойствами, и каждый из которых может планироваться индивидуальной стратегией, например, **Round Robin стратегия** для интерактивных процессов и **FCFS** для пакетных процессов.

Взаимодействие очередей осуществляется по следующим правилам: ни один процесс с более низким приоритетом не может быть запущен, пока не выполнятся процессы во всех очередях с более высоким при-

оритетом.

Работа процесса из очереди с более низким приоритетом может быть приостановлена, если в одной из очередей с более высоким приоритетом появился процесс.

Использование многоуровневой очереди с обратными связями (multilevel feedback queue sheduling) (рис. 19)

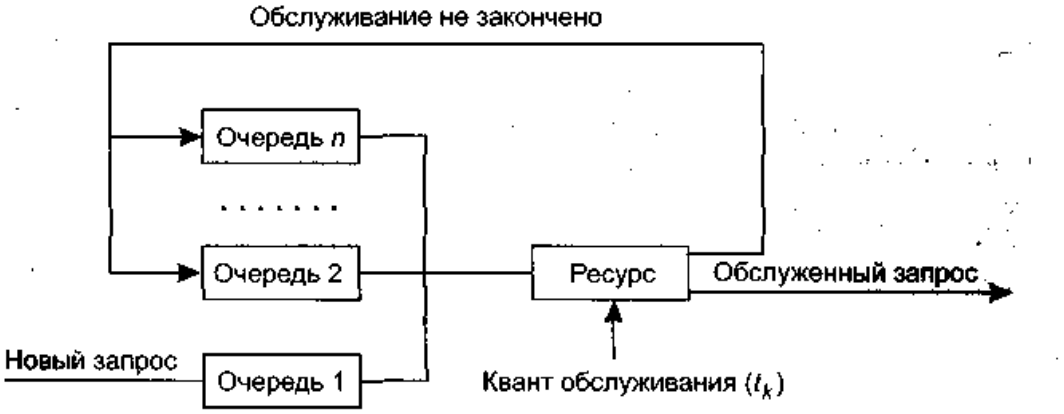

Рис. 19. Многоуровневая очередь с обратными связями

Обычная многоуровневая очередь не допускает перемещения процессов между очередями. Многоуровневая очередь с обратными связями предполагает, что процессы при определенных условиях могут перемещаться между очередями. Здесь организуется N очередей. Все новые запросы поступают в конец первой очереди. Первый запрос из i-й очереди поступает на обслуживание лишь тогда, когда все очереди от 1-й до (i – 1)-й пустые. На обслуживание выделяется квант времени $\mathbf{t_k}$**.** Если за это время обслуживание запроса завершается полностью, то он покидает систему. В противном случае недообслуженный запрос поступает в конец (i + 1)-й очереди.

После обслуживания запроса из i-й очереди система выбирает для обслуживания запрос из непустой очереди с самым младшим номером. Таким запросом может быть следующий запрос из очереди i или из очереди i + 1 (при условии, что после обслуживания запроса из очереди i последняя оказалась пустой). Новый запрос поступает в 1-ю очередь (i = 1). В такой ситуации после окончания времени $\mathbf{t_k}$**,** выделенного для обслуживания запроса из очереди i, будет начато обслуживание запроса первой очереди. Если система выходит на обслуживание заявок из N-й очереди, то они обслуживаются либо по дисциплине **FIFO** (каждая заявка обслуживается до конца), либо по циклическому алгоритму. Данная система наиболее быстро обслуживает все короткие по времени обслуживания запросы. Недостаток системы заключается

в затратах времени на перемещение запросов из одной очереди в другую.

Данная стратегия является универсальной и сочетает в себе свойства всех рассмотренных раньше стратегий — **FCFS, SJF, приоритетная, Round Robin, многоуровневая очередь.**

Рис. 20 Приоритетная, многоочередная дисциплина обслуживания

Вновь поступающие в систему запросы устанавливаются не обязательно в 1-ю очередь, а в очередь в соответствии с имеющимися **приоритетами**, которые определяются параметрами обслуживания процессов. Приоритетные многоочередные дисциплины обслуживания могут использовать обслуживание с абсолютным и относительным приоритетом. **При обслуживании с абсолютным приоритетом** приоритет определяется номером очереди, и **первыми обслуживаются запросы, обладающие наивысшим приоритетом** (из очереди с меньшим номером запрос из очереди i - 1 будет прерывать обработку запроса из очереди i).

В данной дисциплине еще более увеличивается степень дискриминации по среднему времени ожидания в очереди между высоко- и низкоприоритетными запросами. Время ожидания высокоприоритетных заявок сокращается, но за счет большей задержки в обслуживании низкоприоритетных заявок. Достигается это за счет усложнения логики системы, дополнительной обработки запросов и выбора правила дообслуживания прерываемых процессов. **Обслуживание с относительным приоритетом** не вызывает прерывания обслуживаемой заявки до ее завершения, даже если она менее приоритетная.

Контрольные вопросы:

1. Дать определение понятию «процесс». Привести классификацию процессов по временным характеристикам

2. Перечислить состояния процесса. Нарисовать диаграмму переходов для состояний процессов
3. Привести классификацию процессов по связности
4. Перечислить состояния процесса. Нарисовать диаграмму переходов между процессами
5. Дать определение понятию «очередь». Перечислить виды очередей процессов.
6. Перечислить функции ОС при работе с процессами
7. Перечислить и охарактеризовать критерии, позволяющие сравнивать алгоритмы краткосрочных планировщиков
8. Перечислить и охарактеризовать стратегии планирования процессора

Лекция 15
Управление оперативной памятью

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы. Распределению подлежит вся оперативная память, не занятая операционной системой (обычно ОС располагается в самых младших адресах). От выбранных механизмов распределения памяти между выполняющимися процессами в значительной степени зависит эффективность использования ресурсов системы, ее производительность.

Функциями ОС по управлению памятью являются:

- ✓ отслеживание свободной и занятой памяти,
- ✓ выделение памяти процессам и освобождение памяти при завершении процессов,
- ✓ вытеснение процессов из оперативной памяти на диск, когда размеры основной памяти не достаточны для размещения в ней всех процессов,
- ✓ возвращение процессов в оперативную память, когда в ней освобождается место,
- ✓ настройка адресов программы на конкретную область физической памяти.

Память и отображения, виртуальное адресное пространство

Типы адресов

Для идентификации переменных и команд используются:

- ✓ символьные имена (метки),
- ✓ виртуальные адреса,
- ✓ физические адреса.

Символьные имена присваивает пользователь при написании программы на алгоритмическом языке или ассемблере.

Имена переменных и входных точек программных модулей составляют пространство символьных имен. Иногда это адресное пространство называют логическим.

Виртуальные адреса вырабатывает транслятор, переводящий программу на машинный язык. Так как во время трансляции в общем случае не известно, в какое место оперативной памяти будет загружена программа, то транслятор присваивает переменным и командам **виртуальные (условные) адреса,** обычно считая по умолчанию, что программа будет размещена, начиная с нулевого адреса.

Совокупность виртуальных адресов процесса называется виртуальным адресным пространством.

Каждый процесс имеет собственное виртуальное адресное пространство. Максимальный размер виртуального адресного пространства ограничивается разрядностью адреса, присущей данной архитектуре компьютера, и, как правило, не совпадает с объемом физической памяти, имеющимся в компьютере.

Физическая память представляет собой упорядоченное множество ячеек реально существующей оперативной памяти, и все они пронумерованы, то есть к каждой из них можно обратиться, указав ее порядковый номер (адрес). Количество ячеек физической памяти ограниченно и фиксировано.

Физические адреса соответствуют номерам ячеек оперативной памяти, где в действительности расположены или будут расположены переменные и команды.

Системное программное обеспечение должно **связать** каждое указанное пользователем **символьное имя с физической ячейкой памяти, то есть осуществить отображение пространства имен на физическую память компьютера.**

В общем случае это отображение осуществляется в два этапа:

1. **системой программирования** (программа - транслятор),
2. **операционной системой** (это второе отображение осуществляется с помощью соответствующих аппаратных средств процессора — **подсистемы управления памятью – диспетчером памяти,** которая использует дополнительную информацию, подготавливаемую и обрабатываемую операционной системой).

Между этими этапами **обращения к памяти имеют форму виртуального адреса.** При этом можно сказать, что множество всех допустимых значений виртуального адреса для некоторой программы определяет ее **виртуальное адресное пространство** или **виртуальную память.**

Виртуальное адресное пространство программы зависит, прежде всего, от архитектуры процессора и от системы программирования и практически не зависит от объема реальной физической памяти компьютера. Можно еще сказать, что адреса команд и переменных в машинной программе, подготовленной к выполнению системой программирования, как раз и являются виртуальными адресами.

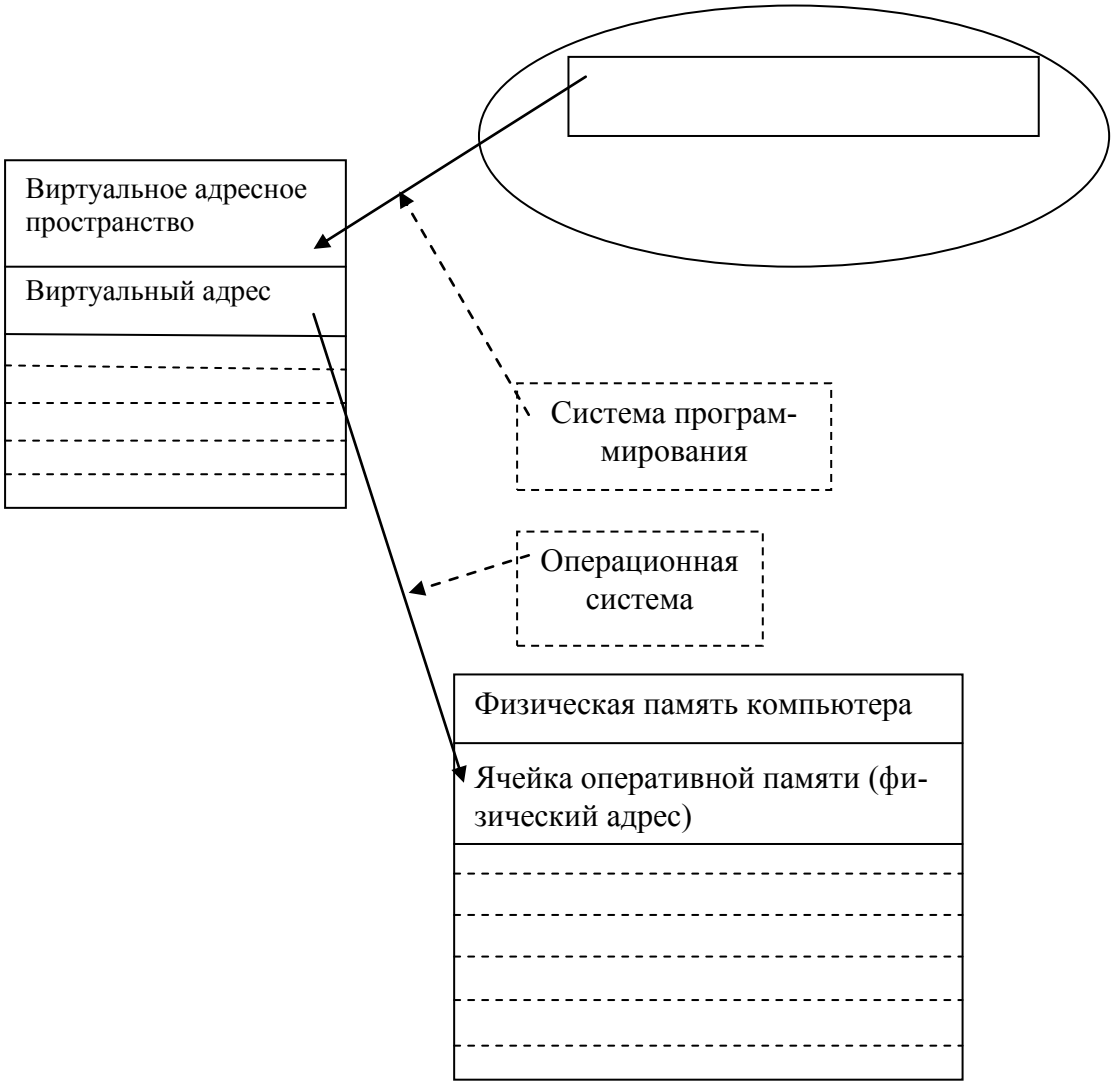

Рис. 21. Память и отображения

Возможны три ситуации:

- ✓ объем виртуального адресного пространства программы меньше объема физической памяти (сейчас почти не встречается);

- ✓ объем виртуального адресного пространства программы равен объему физической памяти (встречается очень часто, особенно характерна была для недорогих вычислительных комплексов);
- ✓ объем виртуального адресного пространства программы больше объема физической памяти (теперь это самая обычная ситуация).

Простое непрерывное распределение

Простое непрерывное распределение (смежное размещение процессов) — это самая простая схема, согласно которой вся память условно может быть разделена на три области:

- ✓ область, занимаемая операционной системой;
- ✓ область, в которой размещается исполняемая задача;
- ✓ незанятая ничем (свободная) область памяти.

Чтобы для задач отвести как можно больший объем памяти, операционная система строится таким образом, чтобы постоянно в оперативной памяти располагалась только **самая нужная ее часть**. Эту часть операционной системы стали называть **ядром.**

Остальные модули операционной системы, не относящиеся к ее ядру, могут быть обычными **транзитными**, то есть загружаться в оперативную память только по необходимости и после своего выполнения вновь освобождать память.

Рис. 22. Однопрограммный режим

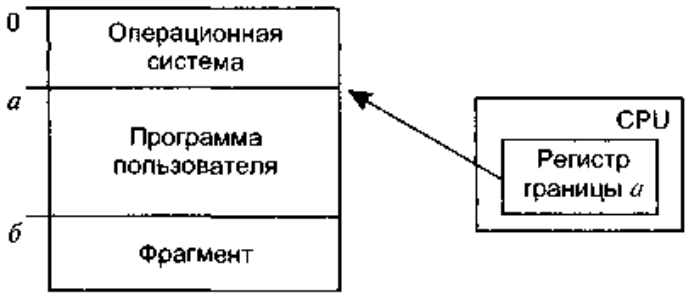

Рис. 23. Регистр границы

В процессе выполнения программы все ее адреса не должны быть меньше числа *а.* В противном случае возможна запись какого-либо результата работы программы поверх операционной системы и уничтожение некоторых ее частей. Защиту операционной системы в случае смежного размещения

при однопрограммном режиме можно осуществить с помощью регистра границы (рис. 23).

Во время работы прикладной программы все адреса, генерируемые CPU, сравниваются с содержимым регистра границы. Если генерируется адрес меньше числа *a,* работа программы прерывается.

Если нужно создать программу, логическое адресное пространство которой должно быть больше, чем свободная область памяти, или даже чем весь возможный объем оперативной памяти, то используется **распределение с перекрытием — так называемые оверлейные структуры** (от overlay — перекрытие, расположение поверх чего-то). Этот метод распределения предполагает, что вся программа может быть разбита на части — **сегменты**. Пока в оперативной памяти располагаются выполняющиеся сегменты, остальные находятся во внешней памяти.

Система MS DOS поддерживает распределения памяти с перекрытием (оверлейные структуры).

Распределение памяти статическими и динамическими разделами

Память задаче может выделяться одним сплошным участком (в этом случае говорят о методах неразрывного распределения памяти) **или несколькими порциями,** которые могут быть размещены в разных областях памяти (тогда говорят о методах разрывного распределения).

Память, не занятая ядром операционной системы, может быть разбита на несколько частей — разделов.

Разделы с фиксированными границами

Метод предполагает разделение адресного пространства на ряд разделов фиксированной величины. В каждом разделе размещается один процесс. Каждый раздел может иметь свою очередь или может существовать одна общая (глобальная) очередь для всех разделов (рис. 24.).

Эта схема была реализована в IBM OS/360 (MFT) и в DEC RSX-11.

Подсистема управления памятью (диспетчер памяти) сравнивает размер программы, поступившей на выполнение, выбирает подходящий раздел, осуществляет загрузку программы и настройку адресов.

В какой раздел помещать программу? Распространены **три стратегии:**

- ✓ **Стратегия первого подходящего** (First fit). Задание помещается в первый подходящий по размеру раздел.

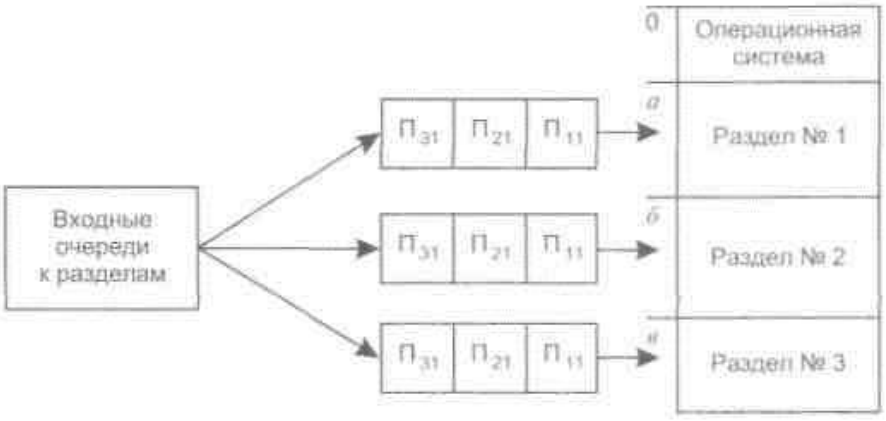

Рис. 24.Режим мультипрограммирования с фиксированным количеством разделов

- ✓ **Стратегия наиболее подходящего** (Best fit). Задание помещается в тот раздел, где ему наиболее тесно.
- ✓ **Стратегия наименее подходящего** (Worst fit). При помещении в самый большой раздел в нем остается достаточно места для возможного размещения еще одного процесса.

Недостатком рассматриваемого способа распределения памяти является **фрагментация памяти**

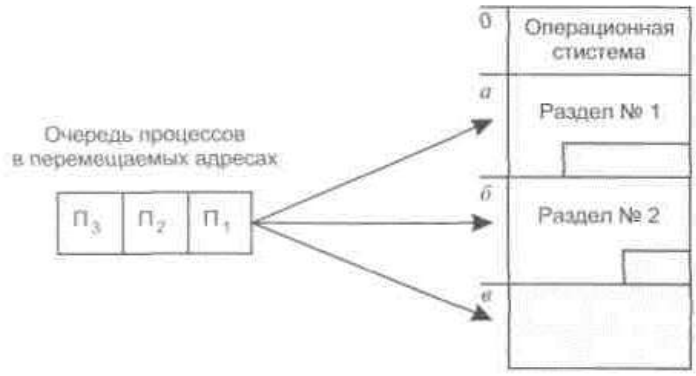

Рис. 25. Режим мультипрограммирования с фиксированным количеством разделов (фрагментация памяти)

При мультипрограммировании имеются две причины фрагментации:

1. размер загруженного процесса меньше размера, занимаемого разделом (внутренняя фрагментация),

размер процесса в очереди больше размера свободного раздела, и этот раздел остается свободным (внешняя фрагментация).

Для защиты памяти при мультипрограммировании с фиксированным количеством разделов необходимы два регистра:

- ✓ регистр верхней границы (наименьший адрес),
- ✓ регистр нижней границы (наибольший адрес).

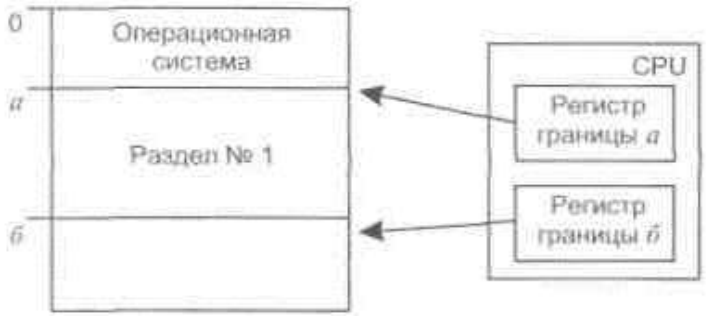

Рис. 26. Регистры границ

Прежде чем программа в разделе *N* начнет выполняться, ее граничные адреса загружаются в соответствующие регистры. В процессе работы программы все формируемые ею адреса контролируются на удовлетворение неравенства $a < Адр. < б$.

При выходе любого адреса программы за отведенные ей границы работа программы прерывается.

Разделы с подвижными границами (мультипрограммирование с переменными разделами)

Метод предполагает разделение памяти на разделы, но границы разделов не фиксируются.

В начальной фазе отсутствует фрагментация, связанная с тем, что размер очередного процесса меньше размера, занимаемого этим процессом раздела. На этой фазе причиной фрагментации является несоответствие размера очередного процесса и оставшегося участка памяти. По мере завершения работы программы освобождаются отдельные разделы. В том случае, **когда освобождаются смежные разделы, границы между ними удаляются и разделы объединяются.**

За счет объединения или слияния смежных разделов образуются большие фрагменты, в которых можно разместить большие программы из очереди. Таким образом, на фазе повторного размещения действуют те же причины фрагментации, что и для метода MFT.

Мультипрограммирование с переменными разделами и уплотнением памяти

Ясно, что этот метод может создать ситуацию, когда в памяти образуется множество малых фрагментов, каждый из которых может быть недостаточен для размещения очередного процесса, однако суммарный размер фрагментов превышает размер этого процесса.

Уплотнением памяти называется перемещение всех занятых разделов по адресному пространству памяти таким образом, чтобы свободный фрагмент занимал одну связную область *(в данном случае ОС должна время от времени копировать содержимое разделов из одного места памяти в другое, корректируя таблицы свободных и занятых областей. Эта процедура называется «сжатием». Сжатие может выполняться либо при каждом завершении задачи, либо только тогда, когда для вновь поступившей задачи нет свободного раздела достаточного размера)*

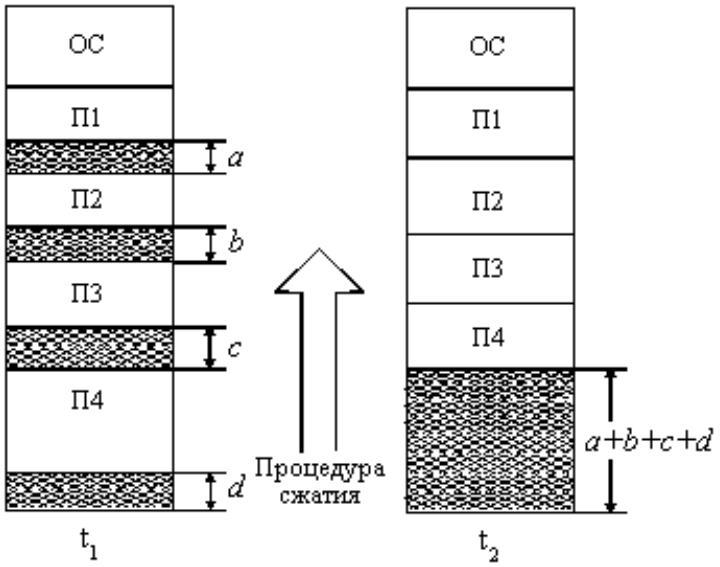

Рис. 27. Режим уплотнения памяти

Контрольные вопросы:

1. Перечислить функции ОС по управлению памятью
2. Перечислить типы адресов программы, описать алгоритм преобразования адресов
3. Пояснить понятие «простое непрерывное распределение». На какие области согласно этому принципу делится оперативная память? Нарисовать схему разделения памяти при однопрограммном режиме.
4. Описать метод разделения памяти на разделы с фиксированными границами. Нарисовать схему.

5. Перечислить типы фрагментации памяти при использовании метода разделения памяти на разделы с фиксированными границами. Нарисовать схему.
6. Описать метод разделения памяти на разделы с подвижными границами. Дать определение понятию «уплотнение памяти». Нарисовать схему.

Лекция 16
Управление виртуальной памятью

Проблема размещения больших программ. Понятие виртуальной памяти

Уже давно существует проблема размещения в памяти программ, размер которых превышает объем доступной памяти. Один из вариантов ее решения организация структур с перекрытием рассмотрен в предыдущей главе. При этом предполагалось активное участие программиста в процессе сегментации и загрузки программы. Было предложено переложить проблему на компьютер. Развитие архитектуры компьютеров привело к значительному усложнению организации памяти, соответственно, усложнились и расширились задачи операционной системы по управлению памятью. Одним из главных усовершенствований архитектуры стало появление виртуальной памяти (virtual memory). Она впервые была реализована в 1959 г. на компьютере Атлас, разработанном в Манчестерском университете, и стала популярной только спустя десятилетие.

При помощи виртуальной памяти обычно решают две задачи:

1. **виртуальная память позволяет адресовать пространство, гораздо большее, чем емкость физической памяти конкретной вычислительной машины.** В соответствии с принципом локальности для реальных программ обычно нет необходимости в помещении их в физическую память целиком.

Возможность выполнения программы, находящейся в памяти лишь частично, имеет ряд вполне очевидных преимуществ:

- ✓ Программа не ограничена величиной физической памяти. Упрощается разработка программ, поскольку можно задействовать большие виртуальные пространства, не заботясь о размере используемой памяти.
- ✓ Поскольку появляется возможность частичного помещения программы (процесса) в память и гибкого перераспределения памяти между программами, можно разместить в памяти больше программ, что увеличивает загрузку процессора и пропускную способность системы.
- ✓ Объем ввода-вывода для выгрузки части программы на диск может быть меньше, чем в варианте классического свопинга, в итоге, каждая программа будет работать быстрее.

Таким образом, возможность обеспечения (при поддержке операционной системы) для программы видимости практически неограниченной (32- или 64-разрядной) адресуемой пользовательской памяти при наличии основной памяти существенно меньших размеров - очень важный аспект.

2. **обеспечение контроля доступа к отдельным сегментам памяти и в частности защиту пользовательских программ друг от друга и защиту ОС от пользовательских программ.**

С целью защиты виртуальная память поддерживалась и на компьютерах с 16-разрядной адресацией, в которых объем основной памяти зачастую существенно превышал 64 Кбайта (размер виртуальной памяти). Например, 16-разрядный компьютер PDP-11/70 мог иметь до 2 Мбайт оперативной памяти. Операционная система этого компьютера, тем не менее, поддерживала виртуальную память, основным смыслом которой являлось **обеспечение защиты и перераспределения основной памяти между пользовательскими процессами.**

Напомним, что в системах с виртуальной памятью те адреса, которые генерирует программа, - (логические адреса) - называются **виртуальными**, и они формируют **виртуальное адресное пространство. В отсутствие механизма виртуальной памяти виртуальное адресное пространство непосредственно отображается в физическое пространство**.

Хотя известны и чисто программные реализации виртуальной памяти, это направление получило наиболее широкое развитие после получения соответствующей аппаратной поддержки. Идея аппаратной части механизма виртуальной памяти состоит в том, что **адрес памяти, вырабатываемый командой, интерпретируется аппаратурой не как реальный адрес некоторого элемента основной памяти, а как некоторая структура, где адрес является лишь одним из компонентов наряду с атрибутами, характеризующими способ обращения по данному адресу.**

Традиционно считается, что существует три модели виртуальной памяти:

1. страничная,
2. сегментная
3. и их комбинация - сегментно-страничная модель.

По-видимому, более правильно считать, что существует (и поддерживается аппаратно большинством платформ) **страничная модель виртуальной памяти.** Причем для тех архитектур, в которых сегменты не поддерживаются аппаратно, их реализация – задача архитектурно-независимой компоненты менеджера памяти. Сегментная организация в чистом виде практически не встречается.

Архитектурные средства поддержки виртуальной памяти

Очевидно, что невозможно создать полностью машинно-независимый компонент управления виртуальной памятью. С другой стороны, имеются существенные части программного обеспечения, связанного с управлением виртуальной памятью, для которых детали аппаратной реализации совершенно не важны. Одним из достижений современных ОС является грамотное и эффективное разделение средств управления виртуальной памятью на аппаратно-независимую и аппаратно-зависимую части. Коротко рассмотрим, что и каким образом входит в аппаратно-зависимую часть подсистемы управления виртуальной памятью.

Итак, мы имеем большое (для 32-разрядных архитектур это обычно 2^{32} = 4 Гб) виртуальное адресное пространство и физическое пространство существенно меньшего размера. Пользовательский процесс или ОС должны иметь возможность осуществить запись по виртуальному адресу, а задача ОС сделать так, чтобы записанная информация оказалась в физической памяти (впоследствии при нехватке оперативной памяти она может быть вытеснена во внешнюю память).

Таким образом, важный компонент менеджера виртуальной памяти - **система или функция отображения (трансляции) адресов.** Механизм преобразования виртуальных адресов в физические должен предусматривать **ведение таблиц**, показывающих, **какие области виртуальной памяти в текущий момент находятся в физической памяти и где именно размещаются.** Если бы такое отображение осуществлять побайтно, то информация об отображении была бы велика, и для ее хранения потребовалось бы больше реальной памяти, чем для процессов. Необходим способ, позволяющий существенно сократить объем информации отображения. Поэтому обычно отображаемая информация группируется в блоки (программа занимает целое количество блоков памяти).

Страничная память

В наиболее простом и наиболее часто используемом случае страничной виртуальной памяти виртуальная память и физическая представляются состоящими из наборов блоков или страниц одинакового размера. Виртуальные адреса делятся на страницы (page), соответствующие единицы в физической памяти образуют страничные кадры (page frames), а в целом **система поддержки страничной виртуальной памяти называется пейджингом** (paging). Передача информации между памятью и диском всегда осуществляется целыми страницами. Страницы, в отличие от сегментов, имеют фиксированную длину, обычно являющуюся степенью числа 2, и не могут перекрываться.

Виртуальный адрес в страничной системе - упорядоченная пара (p,d), где p - номер страницы в виртуальной памяти, а d - смещение в рамках страницы p, где размещается адресуемый элемент. **Процесс может выполняться, если его текущая страница находится в оперативной памяти. Если текущей страницы в главной памяти нет, она должна быть переписана (подкачана) из внешней памяти. Поступившая страница может быть размещена в любой свободный страничный кадр. Система отображения виртуальных адресов в физические сводится к системе отображения виртуальных страниц в физические и представляет собой таблицу страниц.**

Для преобразования адресного пространства каждого процесса используется одна или несколько таблиц страниц, которые обычно хранятся в оперативной памяти. Для ссылки на таблицу страниц используется специальный регистр процессора. Особенности хранения таблицы страниц описаны ниже. Интерпретация виртуального (логического) адреса показана на рис. 28.

Динамическое преобразование адресов в системе осуществляется следующим образом. Выполняемый процесс обращается по виртуальному адресу v = (p,d). Механизм отображения ищет номер страницы p в таблице отображения и определяет, что эта страница находится в страничном кадре p', формируя реальный адрес из p' и d.

Номер страницы, как индекс хранится в **таблице страниц** (page table), которая адресуется при помощи специального регистра процессора, и позволяет определить номер кадра по виртуальному адресу. Помимо этой основной задачи **при помощи атрибутов, записанных в строке таблицы страниц, можно организовать контроль доступа к конкретной странице и ее защиту.**

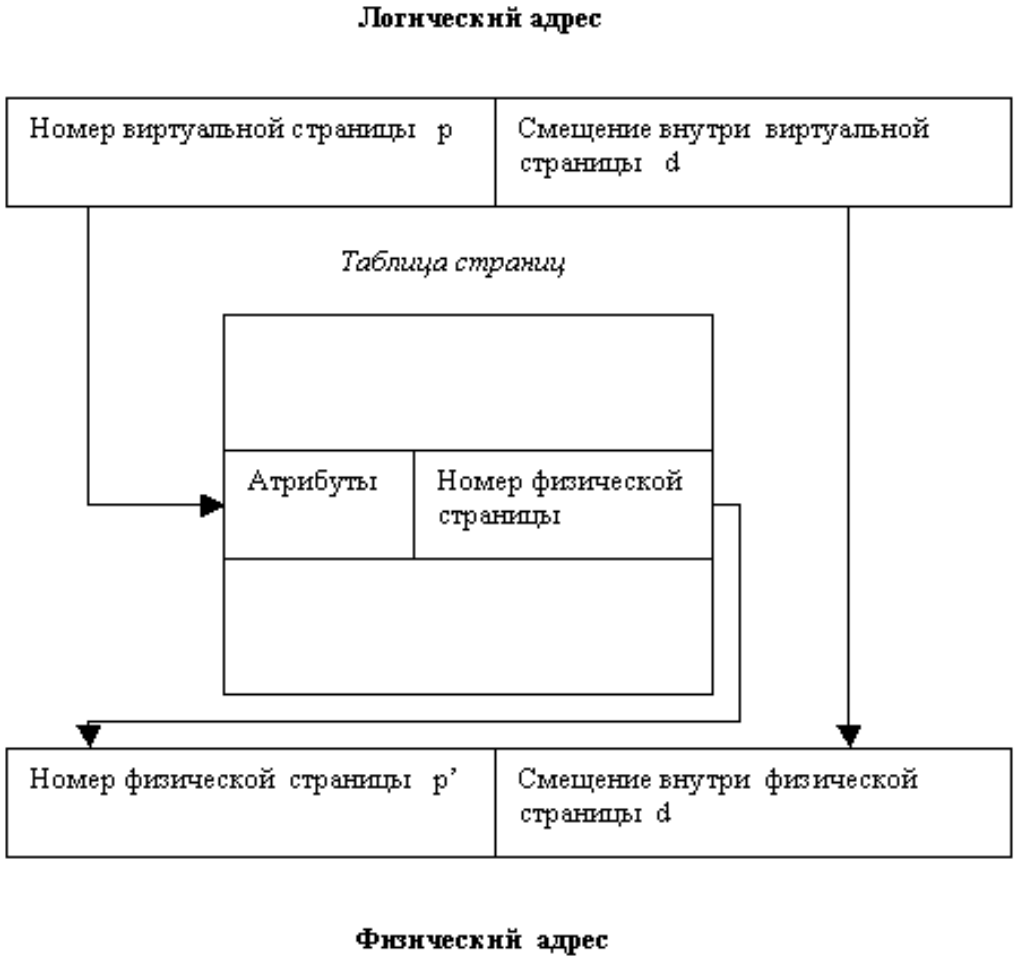

Рис. 28. Связь логического и физического адресов при страничной организации памяти.

При использовании схемы пэйджинга внешняя фрагментация отсутствует. Однако существует внутренняя фрагментация: адресное пространство процесса занимает целое число страниц, и, в среднем, половина страницы на процесс пропадает.

Важный аспект - различие точек зрения пользователя и системы на используемую память. С точки зрения пользователя его память - единое непрерывное пространство, содержащее, только одну программу. Реальное отображение скрыто от пользователя и контролируется ОС. Заметим, что процессу пользователя недоступна чужая память. Он не имеет возможности адресовать память за пределами своей таблицы страниц, которая включает только его собственные страницы.

Для управления физической памятью ОС поддерживает структуру таблицы кадров. Она имеет одну запись на каждый физический кадр, показывающий его состояние.

Отображение должно происходить корректно даже в сложных случаях. ОС поддерживает одну или несколько (при наличии нескольких сегментов памяти) таблиц страниц для каждого процесса, для ссылки, на которую, как уже говорилось, обычно используется специальный регистр. При переключении процессов диспетчер должен найти его таблицу страниц, указатель на которую, таким образом, входит в контекст процесса.

В большинстве современных компьютеров со страничной организацией виртуальной памяти все таблицы страниц хранятся в основной памяти, а быстрота доступа к элементам таблицы текущей виртуальной памяти достигается за счет наличия сверхбыстродействующей буферной памяти (кэша).

Сегментная и сегментно-страничная организации памяти

Существуют две другие схемы организации виртуальной памяти: сегментная и сегментно-страничная. **При сегментной организации виртуальный адрес по-прежнему является двумерным и состоит из двух полей - номера сегмента и смещения внутри сегмента.** Заметим, что с точки зрения ОС сегменты являются логическими сущностями и их главное назначение - хранение и защита однородной информации (кода, данных и т.д.).

С точки зрения пользователя процесс представляется обычно не как линейный массив байтов, а как набор сегментов переменного размера (данные, код, стек). **Сегментация - схема управления памятью, поддерживающая этот взгляд пользователя. Сегменты содержат процедуры, массивы, стек или скалярные величины, но обычно не содержат информацию смешанного типа.**

Программисты, пишущие на языках низкого уровня, должны иметь представление о сегментной организации, явным образом меняя значения сегментных регистров (это хорошо видно по текстам программ, написанных на Ассемблере). **Логическое адресное пространство - набор сегментов. Каждый сегмент имеет имя, размер и другие параметры** (уровень привилегий, разрешенные виды обращений, флаги присутствия). **Пользователь**

специфицирует каждый адрес двумя величинами: именем сегмента и смещением. (В отличие от схемы пэйджинга, где пользователь задает только один адрес, который разбивается hardware на номер страницы и смещение, прозрачным для программиста образом.)

Каждый сегмент - линейная последовательность адресов от 0 до максимума. Различные сегменты могут иметь различные длины, которые могут меняться динамически (например, сегмент стека). В элементе таблицы сегментов помимо физического адреса начала сегмента (если виртуальный сегмент содержится в основной памяти) содержится длина сегмента. **Если размер смещения в виртуальном адресе выходит за пределы размера сегмента, возникает прерывание.**

Логический адрес - упорядоченная пара v = (s,d) - номер сегмента и смещение внутри сегмента.

В системах, где сегменты поддерживаются аппаратно, эти параметры обычно хранятся **в таблице дескрипторов сегментов,** а программа обращается к этим дескрипторам по **номерам-селекторам**. При этом в контекст каждого процесса входит набор **сегментных регистров**, содержащих селекторы текущих сегментов кода, стека, данных и др. и определяющих, какие сегменты будут использоваться при разных видах обращений к памяти. Это позволяет процессору уже на аппаратном уровне определять допустимость обращений к памяти, упрощая реализацию защиты информации от повреждения и несанкционированного доступа.

Аппаратная поддержка сегментов относительно слабо распространена (главным образом на процессорах архитектуры Intel) и характеризуется довольно медленной загрузкой селекторов в сегментные регистры, выполняемая при каждом переключении контекста и при каждом переходе между разными сегментами. В системах с чисто страничной организацией памяти для описания типового адресного пространства процесса, представляющего собой набор сегментов, сегментация реализуется на уровне, независимом от аппаратуры.

Хранение в памяти сегментов большого размера может оказаться неудобным. Возникает идея их пейджинга. **При сегментно-страничной организации виртуальной памяти происходит двухуровневая трансляция виртуального адреса в физический.**

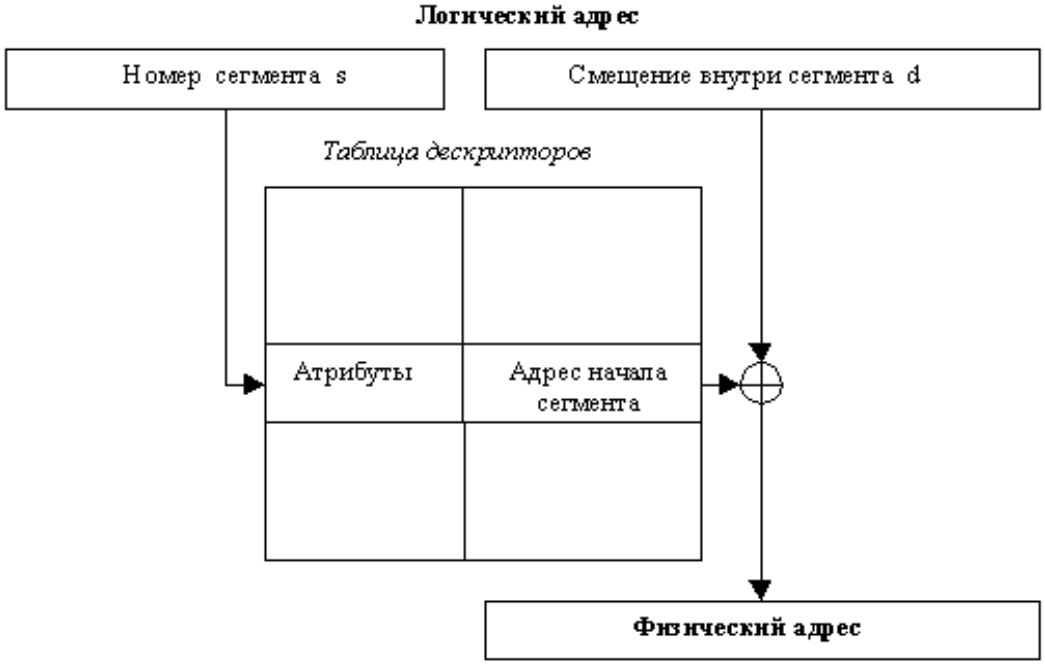

Рис. 29. Преобразование логического адреса при сегментной организации памяти

В этом случае виртуальный адрес v состоит из трех полей:

- ✓ **s - номера сегмента виртуальной памяти,**
- ✓ **p - номера страницы внутри сегмента**
- ✓ **d - смещения внутри страницы.**

Соответственно, используются две таблицы отображения - таблица сегментов, связывающая номер сегмента с таблицей страниц, и отдельная таблица страниц для каждого сегмента.

Сегментно-страничная организация виртуальной памяти позволяет совместно использовать одни и те же сегменты данных и программного кода в виртуальной памяти разных задач (для каждой виртуальной памяти существовала отдельная таблица сегментов, но для совместно используемых сегментов поддерживались общие таблицы страниц).

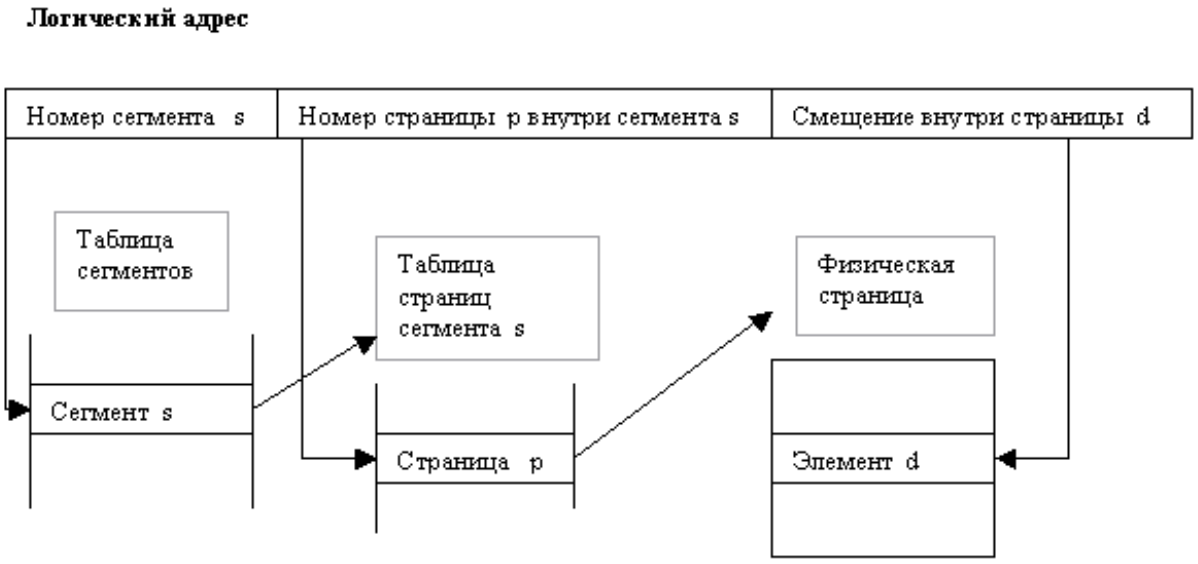

Рис. 30. Формирование физического адреса при сегментно-страничной организации памяти.

Контрольные вопросы:

1. Пояснить необходимость использования механизма виртуальной памяти
2. Перечислить модели виртуальной памяти
3. Охарактеризовать метод страничной организации памяти или пэйджинг
4. Охарактеризовать метод сегментной организации памяти
5. Охарактеризовать метод сегментно-страничной организации памяти

Лекция 17
Модели памяти в ОС Windows

Краткие сведения об архитектуре Windows 3.x

В соответствии с архитектурой Windows все прикладные программы и системный код размещаются в едином адресном пространстве. Это означает, что недоработанная прикладная программа, содержащая ошибки, может испортить области памяти, которые используются операционной средой или другой прикладной программой. Результатом будет весьма неприятная ошибка общего нарушения защиты (General Protection Fault). Иногда Windows с честью выходит из положения, восстанавливая свою работоспособность, но чаще всего это ей не удается.

В своей основе Windows 3.x — 16-разрядная операционная система, поэтому для программ память представляется состоящей из 64-Кбайт сегментов, а все данные в своей основе 16-разрядные. Такая система может оказаться менее эффективной по сравнению с 32-разрядной адресацией при работе с большими массивами данных. Еще одно следствие 16-разрядной базы этой ОС — ограниченность системных ресурсов. В Windows 3.x для хранения таких структур, как дескрипторы файлов прикладных программ, выделяется лишь небольшой блок памяти в других адресах. После того как эти области памяти заполнятся, Windows не может загрузить новые прикладные программы, даже если в ее распоряжении остается вполне достаточно памяти в других адресах (рис. 31).

В основе организации Windows 3.x лежит 16-разрядная архитектура. Ее ядро, большинство важнейших компонентов и собственные прикладные программы представляют собой 16-разрядные коды. (Ее редко используемый интерфейс Win32 API дает возможность выполнять 32-разрядные прикладные программы, но не позволяет работать с несколькими потоками.)

Все собственные прикладные программы Windows 3.x и все ее системные библиотеки DLL отображаются в общее сегментированное виртуальное адресное пространство размером 4 Гбайт. Все эти компоненты видимы (и часто доступны на уровне записи) друг для друга. В нижней части этого адресного пространства, обычно ниже метки 1 Мбайт, размещаются драйверы устройств реального режима, обеспечивающие взаимодействие с периферийными подсистемами, такими, как видеоплаты или принтеры.

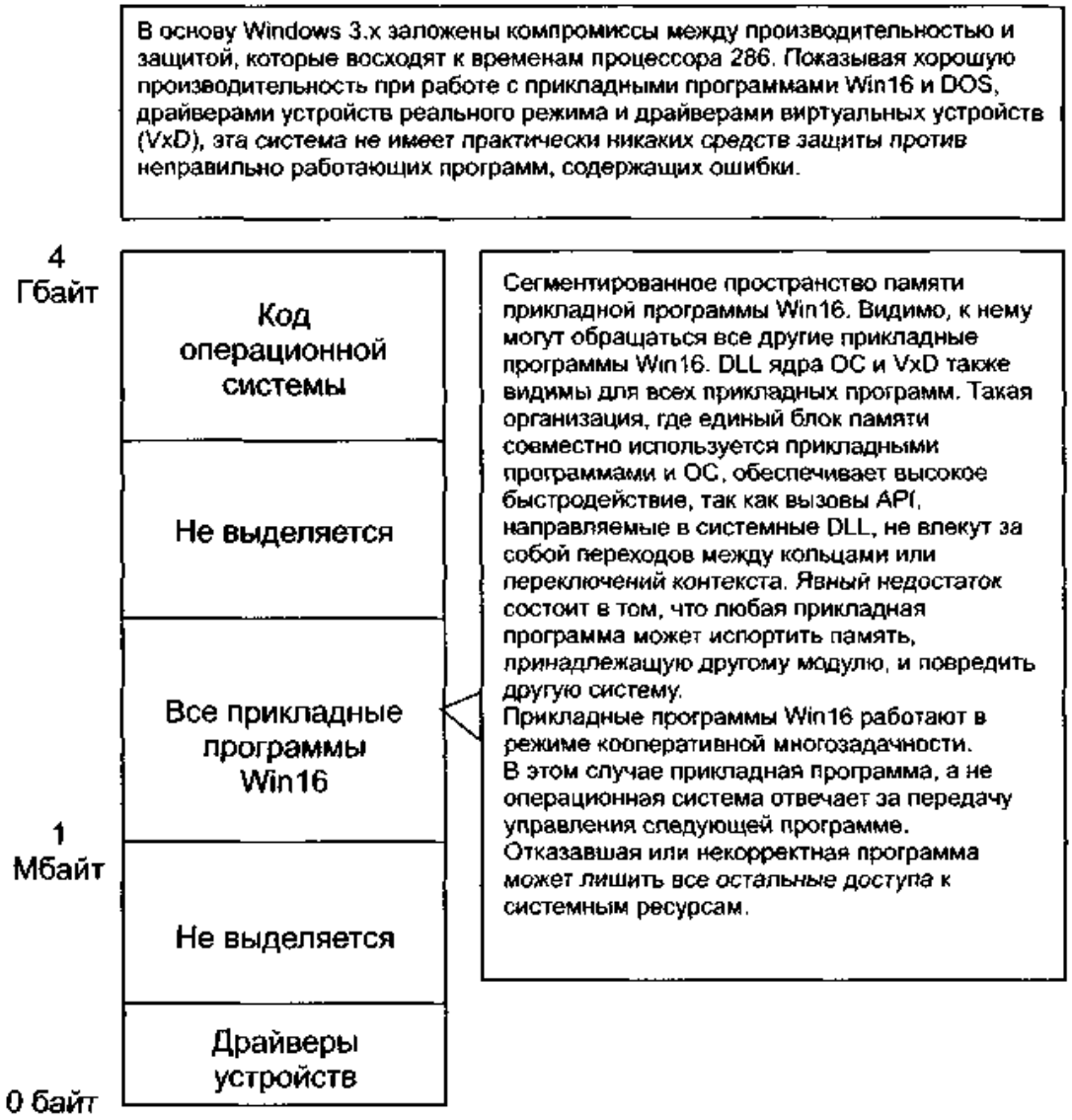

Рис. 31. Модель памяти Windows 3.x

В Windows 3.11 драйверы VxD файловой системы используются для отыскания маршрута доступа к диску в защищенном режиме. Упрощенная организация системы позволяет получить очень малое рабочее множество (working set — прикладной и системный код, который необходимо загрузить в память для любой данной задачи), поэтому Windows 3.1x может успешно выполняться на компьютерах с ОП ограниченного размера. Такая архитектура также способствует повышению эффективности исполнения кода, так как программы могут вызывать функции API из собственного пространства памяти. **Недостаток архитектуры состоит в слабой защите от сбоев при неправильной работе программ. Программы и системные компоненты видимы друг для друга, модуль, содержащий ошибки, может легко испортить содержимое памяти, принадлежащей другому процессу. Хотя Windows 3.1x способна восстанавливать свою работо-**

способность после некоторых нарушений зашиты общего характера (General Protection Fault), зачастую результатом становится крах всей системы.

Windows 3.1x одновременно выполняет несколько прикладных программ с помощью простого механизма планирования, называемого кооперативной многозадачностью. В этой системе каждая прикладная программа должна добровольно уступить управление, когда, проверив свою очередь сообщений, она обнаруживает, что та пуста. Но если прикладная программа не проверит свою очередь сообщений либо по причине занятости, либо вследствие зависания, то другие прикладные программы лишатся доступа к совместно используемым ресурсам.

Краткие сведения по архитектуре Windows 95/98

Windows 95/98 представляет собой продукт эволюционного развития систем Windows 3.1x. Хотя она и несет в себе много важных изменений по сравнению с 16-разрядной архитектурой Windows, в ней сохранены некоторые важнейшие свойства ее предшественницы. Результатом стало появление гибридной ОС, способной работать с 16-разрядными прикладными программами Windows, программами, унаследованными от DOS, и старыми драйверами устройств реального режима и в то же время совместимой с 32-разрядными прикладными программами и 32-разрядными драйверами виртуальных устройств.

Среди наиболее важных усовершенствований, явившихся в Windows 95/98, — заложенная в ней способность:

- ✓ работать с 32-разрядными многопотоковыми прикладными программами,
- ✓ защищенные адресные пространства,
- ✓ вытесняющая многозадачность,
- ✓ намного более широкое и эффективное использование драйверов виртуальных устройств.

Ее наиболее существенный недостаток состоит в относительно слабой защищенности от плохо работающих программ, содержащих ошибки.

Каждая собственная прикладная программа Windows 95/98 «видит» неструктурированное адресное пространство размером 4 Гбайт, в котором размещается она сама плюс системный код и драйверы Windows 95/98 (рис. 32).

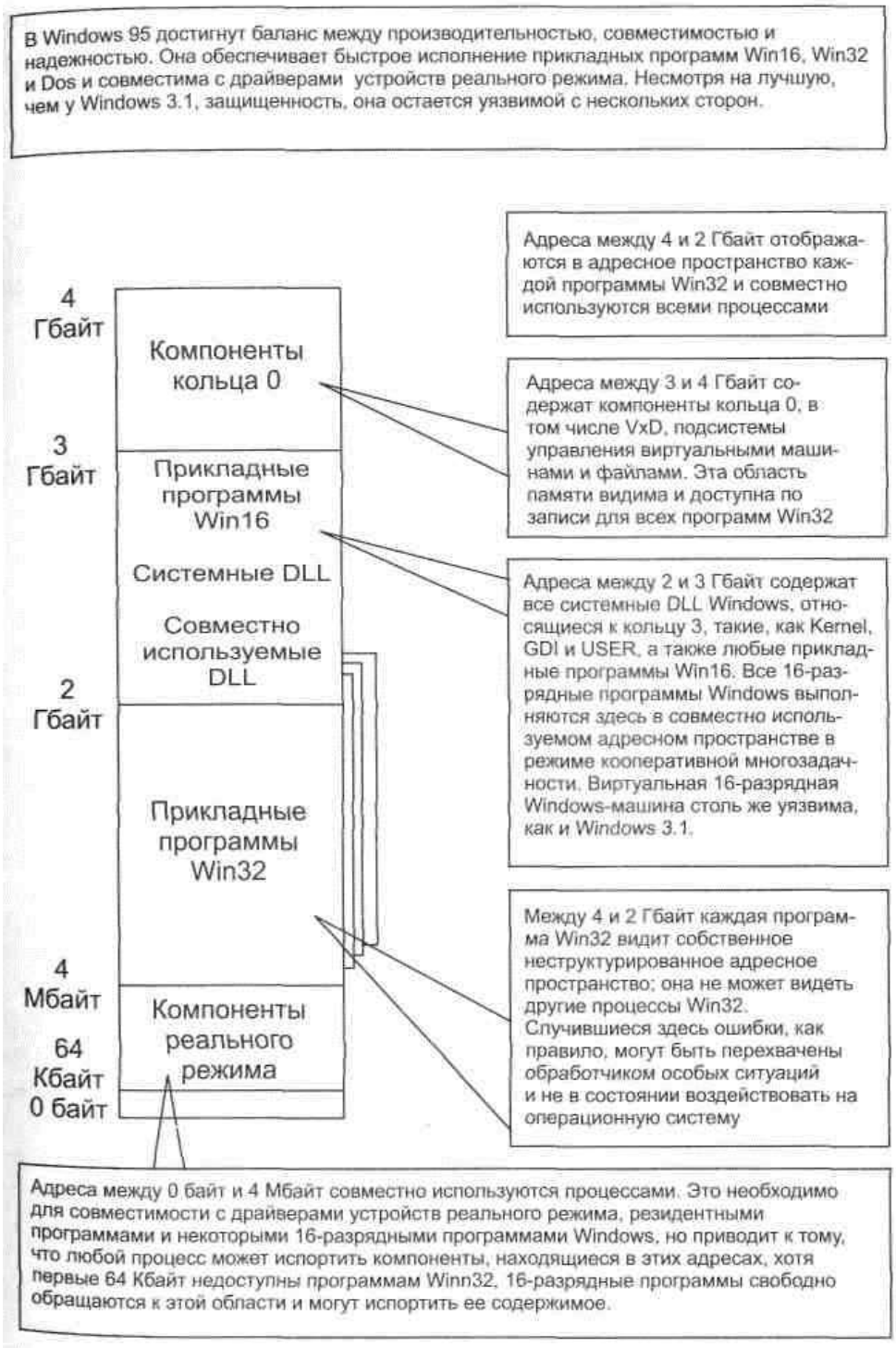

Рис. 32. Модель памяти Windows 95/98

Каждая 32-разрядная прикладная программа выполняется так, как будто она монопольно использует ПК. Код прикладной программы загружается в это адресное пространство между отметками 4 Мбайт и 2 Гбайт. Хотя 32-разрядные прикладные программы «не видят» друг друга, они могут обмениваться данными через буфер обмена (Clipboard), механизмы DDE и OLE. **Все 32-разрядные прикладные программы выполняются**

в соответствии с моделью вытесняющей многозадачности, основанной на управлении отдельными потоками. Планировщик потоков, представляющий собой составную часть системы управления виртуальной памятью (VMM), распределяет время среди группы одновременно выполняемых потоков на основе оценки текущего приоритета каждого потока и его готовности к выполнению. Вытесняющее планирование позволяет реализовать намного более плавный и надежный механизм многозадачности, чем кооперативный метод, используемый в Windows 3.1x.

Системный код Windows 95/98 размещается выше границы 2 Гбайт. В пространстве между отметками 2 и 3 Гбайт находятся системные библиотеки DLL кольца 3 и любые DLL, используемые несколькими программами. (В 32-разрядных процессорах фирмы Intel предоставляются **четыре уровня аппаратной защиты, поименованные, начиная с кольца 0 до кольца 3, причем кольцо 0 является наиболее привилегированным**.) **Компоненты кольца 0 в системе Windows 95/98 отображаются в пространство между 3 и 4 Гбайт. Эти важные участки кода с максимальным уровнем привилегий содержат подсистему управления виртуальными машинами (VMM), файловую систему и драйверы VxD.**

Область памяти между 2 и 4 Гбайт отображается в адресное пространство каждой 32-разрядной прикладной программы, т. е. оно совместно используется всеми 32-разрядными прикладными программами в вашем ПК. Такая организация позволяет обслуживать вызовы API непосредственно в адресном пространстве прикладной программы и ограничивает размер рабочего множества. Однако за это приходится расплачиваться снижением надежности. **Ничто не может помешать программе, содержащей ошибку, произвести запись в адреса, принадлежащие системным DLL, и вызвать крах всей системы.**

В области между 2 и 3 Гбайт также находятся все активные 16-разрядные прикладные программы Windows. С целью обеспечения совместимости эти программы выполняются в совместно используемом адресном пространстве, где они могут испортить друг друга так же, как и в Windows 3.1x.

Адреса памяти ниже 4 Мбайт также отображаются в адресное пространство каждой прикладной программы и совместно используются всеми процессами. Благодаря этому становится возможной совместимость с существующими драйверами реального режима, которым необходим доступ к этим адресам**. Это делает еще одну область памяти незащищенной от случайной записи. К самым нижним 64 Кбайт этого адресного пространства 32-разрядные прикладные программы обращаться не могут, что**

дает возможность перехватывать неверные указатели, но 16-разрядные программы, которые, возможно, содержат ошибки, могут записывать туда данные.

Управление памятью Windows NT

Windows NT Workstation 3.51 по существу представляет собой операционную систему сервера, приспособленную для использования на рабочей станции. Этим обусловлена архитектура, в которой абсолютная защита прикладных программ и данных берет верх над соображениями скорости и совместимости. Чрезвычайная надежность Windows NT обеспечивается ценой высоких системных затрат, поэтому для получения приемлемой производительности необходимы быстродействующий ЦП и по меньшей мере 16-Мбайт ОП. В системе Windows NT безопасность нижней памяти достигается за счет отказа от совместимости с драйверами устройств реального режима. В среде Windows NT работают собственные 32-разрядные NT-прикладные программы, а также большинство прикладных программ Windows 95/98. Так же, как и Windows 95/98, система Windows NT позволяет выполнять в своей среде 16-разрядные Windows- и DOS-программы.

Схема распределения памяти Windows NT отличается от распределения памяти системы Windows 95/98 (см. рис. 2.31). **Собственным прикладным программам выделяется 2 Гбайт особого адресного пространства, от границы 64 Кбайт до 2 Гбайт** (первые 64 Кбайт полностью недоступны). **Прикладные программы изолированы друг от друга, хотя могут общаться через буфер обмена Clipboard, механизмы DDE и OLE.**

В верхней части каждого 2-Гбайт блока прикладной программы размещен код, воспринимаемый прикладной программой как системные библиотеки DLL кольца 3. На самом деле это просто заглушки, выполняющие перенаправление вызовов, называемые **DLL клиентской стороны** (client-side DLLs). **При вызове большинства функций API из прикладной программы библиотеки DLL клиентской стороны обращаются к локальным процедурам (Local Process Communication — LPC), которые передают вызов и связанные с ним параметры в совершенно изолированное адресное пространство, где содержится собственно системный код. Этот сервер-процесс (server process) проверяет значение параметров, исполняет запрошенную функцию и пересылает результаты назад в адресное пространство прикладной программы. Хотя сервер-процесс сам по себе остается прикладного уровня, он полностью защищен от вызывающей его программы и изолирован от нее.**

Между отметками **2 и 4 Гбайт** расположены низкоуровневые системные компоненты Windows NT кольца 0, в том числе **ядро, планировщик потоков и диспетчер виртуальной памяти.** Системные страницы в этой области наделены привилегиями супервизора, которые задаются физическими схемами кольцевой защиты процессора. Это делает низкоуровневый системный код невидимым и недоступным по записи для программ прикладного уровня, но приводит к падению производительности во время переходов между кольцами.

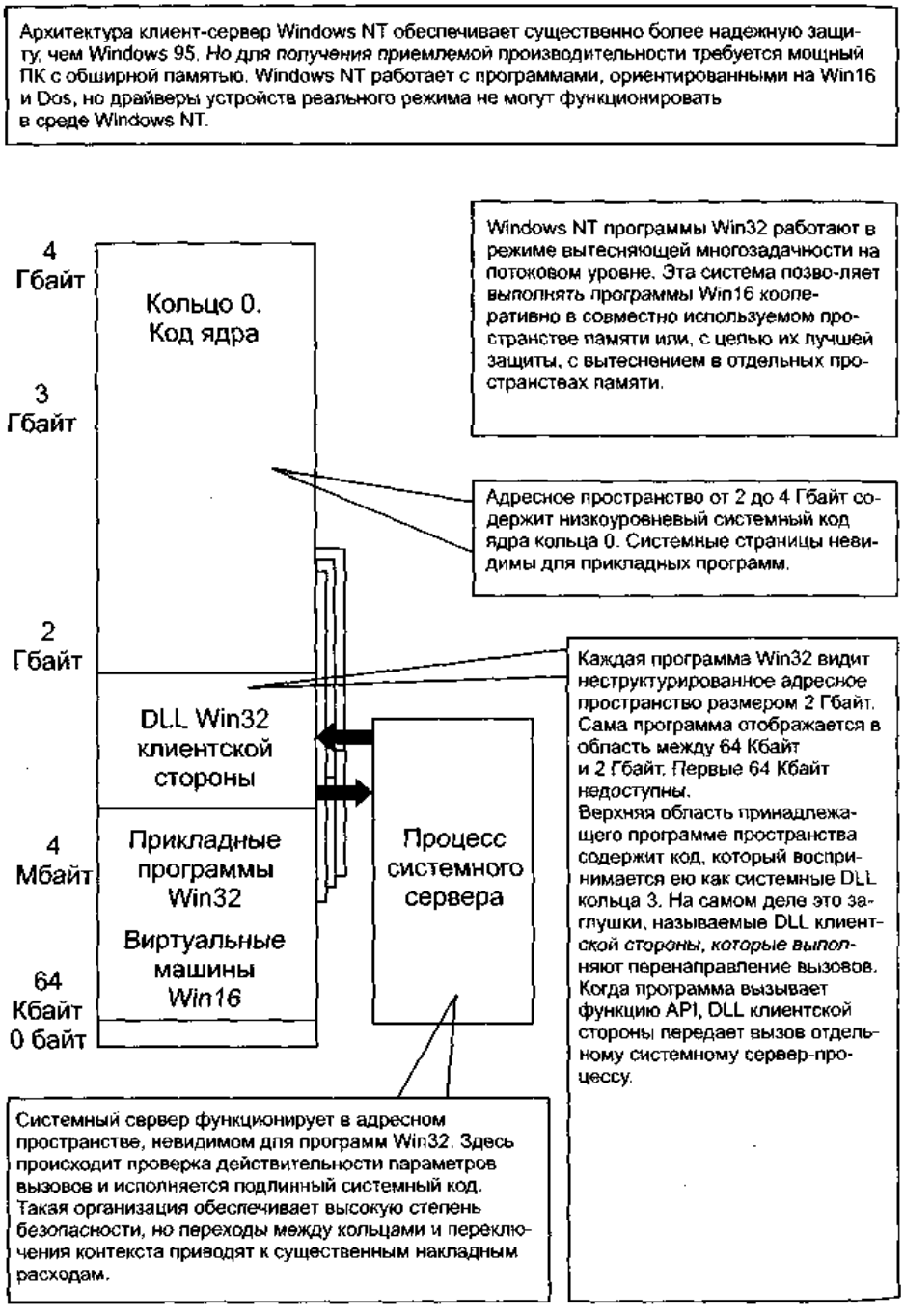

Рис. 33. Модель памяти Windows NT

Для 16-разрядных прикладных Windows-программ Windows NT реализует сеансы Windows on Windows (WOW). **Windows NT дает возможность выполнять 16-разрядные программы Windows индивидуально в собственных пространствах памяти или совместно в разделяемом адресном пространстве.** Почти во всех случаях 16- и 32-разрядные прикладные программы Windows могут свободно взаимодействовать, используя OLE (при необходимости через особые процедуры thunk) независимо оттого, выполняются они в отдельной или общей памяти. **Собственные прикладные программы и сеансы WOW выполняются в режиме вытесняющей многозадачности, основанной на управлении отдельными потоками. Множественные 16-разрядные прикладные программы Windows в одном сеансе WOW выполняются в соответствии с кооперативной моделью многозадачности. Windows NT может также выполнять в многозадачном режиме несколько сеансов DOS.** Поскольку Windows NT имеет полностью 32-разрядную архитектуру, не существует теоретических ограничений на ресурсы GDI и USER.

API (application programming interface) - это интерфейс прикладного программирования. Если говорить более простым языком, то это набор различных функций, констант, классов, форматов запросов, которые можно использовать в других программах.

Контрольные вопросы:

1. Охарактеризовать модель памяти Windows 3.x
2. Охарактеризовать модель памяти Windows 95/98
3. Охарактеризовать модель памяти Windows NT

Лекция 18
Обработка прерываний

Прерывания - механизм, позволяющий координировать параллельное функционирование отдельных устройств вычислительной системы, реагировать на особые состояния, возникающие при работе процессора.

Прерывание — это принудительная передача управления от выполняемой программы системе (а через нее — к соответствующей программе обработки прерывания), происходящая при возникновении определенного события.

Идея прерывания была предложена также очень давно — в середине 50-х годов, — можно без преувеличения сказать, что она внесла наиболее весомый вклад в развитие вычислительной техники. Основная цель введения прерываний — реализация асинхронного режима функционирования и распараллеливание работы отдельных устройств вычислительного комплекса.

Механизм прерываний реализуется аппаратно-программными средствами. Структуры систем прерывания (в зависимости от аппаратной архитектуры) могут быть самыми разными, но все они имеют одну общую особенность — **прерывание непременно влечет за собой изменение порядка выполнения команд процессором.**

Механизм обработки прерываний независимо от архитектуры вычислительной системы подразумевает выполнение некоторой последовательности шагов.

1. Установление факта прерывания (прием сигнала запроса на прерывание) и идентификация прерывания (в операционных системах идентификация прерывания иногда осуществляется повторно, на шаге 4).

2. Запоминание состояния прерванного процесса вычислений. *Состояние процесса выполнения программы определяется, прежде всего, значением счетчика команд (адресом следующей команды, который, например, в i80x86 определяется регистрами CS и IP — указателем команды), содержимым регистров процессора, и может включать также спецификацию режима (например, режим пользовательский или привилегированный) и другую информацию.*

3. Управление аппаратно передается на подпрограмму обработки прерывания. *В простейшем случае в счетчик команд заносится начальный адрес подпрограммы обработки прерываний, а в соответствующие регистры — информация из слова состояния. В более развитых процессорах, например в 32-разрядных микропроцессорах фирмы Intel (начиная с i80386 и включая последние процессоры Pentium IV) и им подобных, осуществляются достаточно сложная процедура определения начального адреса соответствующей подпрограммы обработки прерывания и не менее сложная процедура инициализации рабочих регистров процессора.*

4. Сохранение информации о прерванной программе, которую не удалось спасти на шаге 2 с помощью аппаратуры. *В некоторых процессорах предусматривается запоминание довольно большого объема информации о состоянии прерванных вычислений.*

5. Собственно выполнение программы, связанной с обработкой прерывания. *Эта работа может быть выполнена той же подпрограммой, на которую было передано управление на шаге 3, но в операционных системах достаточно часто она реализуется путем последующего вызова соответствующей подпрограммы.*

6. Восстановление информации, относящейся к прерванному процессу (этап, обратный шагу 4).

7. Возврат на прерванную программу.

Шаги 1-3 реализуются аппаратно, шаги 4-7 — программно.

При возникновении запроса на прерывание естественный ход вычислений нарушается и управление передается на программу обработки возникшего прерывания (рис. 34). При этом средствами аппаратуры сохраняется (как правило, с помощью механизмов стековой памяти) адрес той команды, с которой следует продолжить выполнение прерванной программы. После выполнения программы обработки прерывания управление возвращается на прерванную ранее программу посредством занесения в указатель команд сохраненного адреса команды, которую нужно было бы выполнить, если бы не возникло прерывание. Однако такая схема используется только в самых простых программных средах. В мультипрограммных операционных системах обработка прерываний происходит по более сложным схемам.

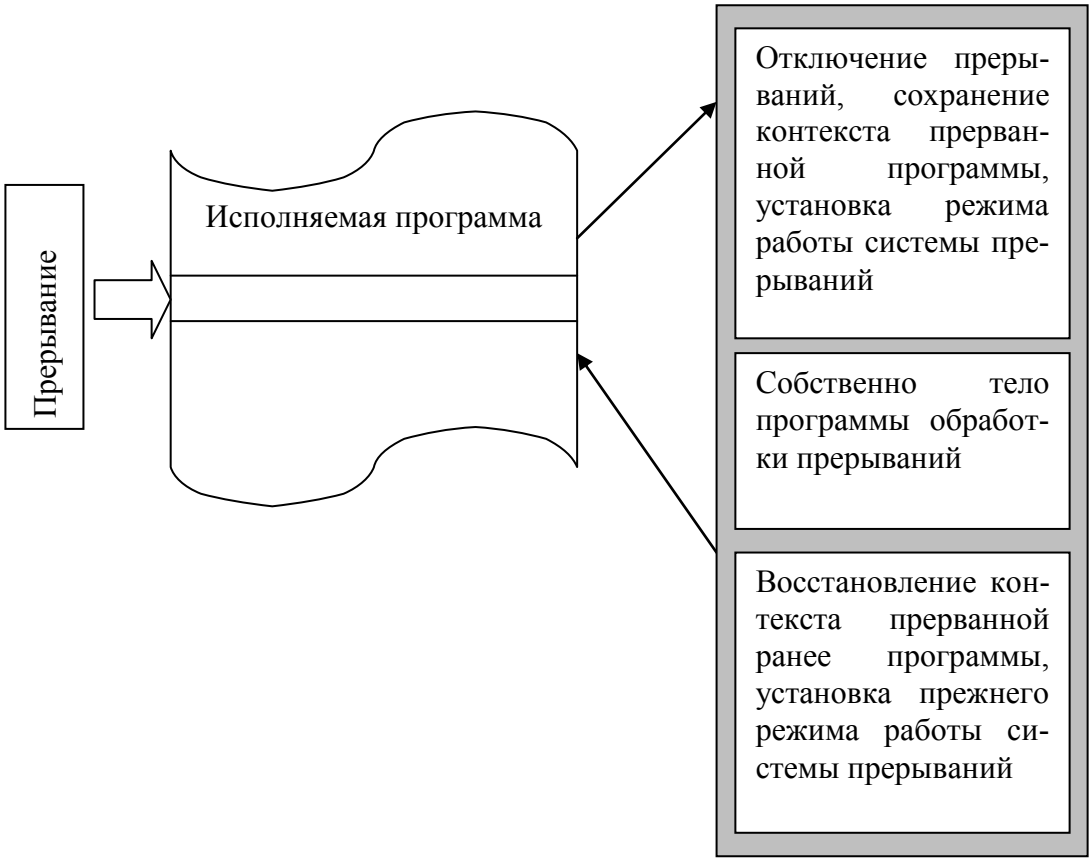

Рис. 34. Обработка прерывания

Итак, **главные функции механизма прерываний:**

1. распознавание или классификация прерываний;
2. передача управления соответствующему обработчику прерываний;
3. корректное возвращение к прерванной программе.

Переход от прерываемой программы к обработчику и обратно должен выполняться как можно быстрей. Одним из самых простых и быстрых методов является использование таблицы, содержащей перечень всех допустимых для компьютера прерываний и адреса соответствующих обработчиков. **Для корректного возвращения к прерванной программе перед передачей управления обработчику прерываний содержимое регистров процессора запоминается либо в памяти с прямым доступом, либо в системном стеке (system stack).**

Прерывания, возникающие при работе вычислительной системы, можно разделить на два основных класса:

1. **внешние** (их иногда называют асинхронными)

2. **внутренние** (синхронные).

Внешние прерывания вызываются асинхронными событиями, которые происходят вне прерываемого процесса, например:

1. прерывания от таймера;
2. прерывания от внешних устройств (прерывания по вводу-выводу);
3. прерывания по нарушению питания;
4. прерывания с пульта оператора вычислительной системы;
5. прерывания от другого процессора или другой вычислительной системы.

Внутренние прерывания вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями. Примерами являются следующие запросы на прерывания:

1. при нарушении адресации (в адресной части выполняемой команды указан запрещенный или несуществующий адрес, обращение к отсутствующему сегменту или странице при организации механизмов виртуальной памяти);
2. при наличии в поле кода операции незадействованной двоичной комбинации;
3. при делении на ноль;
4. вследствие переполнения или исчезновения порядка;
5. от средств контроля (например, вследствие обнаружения ошибки четности, ошибок в работе различных устройств).

Могут еще существовать ***прерывания в связи с попыткой выполнить команду, которая сейчас запрещена.*** *Во многих компьютерах часть команд должна выполняться только кодом самой операционной системы, но не прикладными программами. Это делается с целью повышения защищенности выполняемых на компьютере вычислений. Соответственно в аппаратуре предусмотрены различные режимы работы,* ***пользовательские программы выполняются в режиме, в котором некоторое подмножество команд, называемых привилегированными, не исполняется. К привилегированным командам помимо команд ввода-вывода относятся:***

- ✓ *команды переключения режима работы центрального процессора,*

- ✓ *команды инициализации некоторых системных регистров процессора.*

При попытке использовать команду, запрещенную в данном режиме, происходит внутреннее прерывание, и управление передается самой операционной системе.

Наконец, существуют собственно **программные прерывания.** *Эти прерывания происходят по соответствующей команде прерывания, то есть по этой команде процессор осуществляет практически те же действия, что и при обычных внутренних прерываниях. Этот механизм был специально введен для того, чтобы переключение на системные программные модули происходило не просто как переход на подпрограмму, а точно таким же образом, как и обычное прерывание. Этим, прежде всего, обеспечивается автоматическое переключение процессора в привилегированный режим с возможностью исполнения любых команд. Сигналы, вызывающие прерывания, формируются вне процессора или в самом процессоре, они могут возникать одновременно. Выбор одного из них для обработки осуществляется на основе приоритетов, приписанных каждому типу прерывания.* **Так, со всей очевидностью, прерывания от схем контроля процессора должны обладать наивысшим приоритетом (действительно, если аппаратура работает неправильно, то не имеет смысла продолжать обработку информации).**

На рис. 35 изображен **обычный порядок (приоритеты) обработки прерываний в зависимости от типа прерываний.** Учет приоритета может быть встроен в технические средства, а также определяться операционной системой, то есть кроме аппаратно реализованных приоритетов прерывания большинство вычислительных машин и комплексов допускают программно-аппаратное управление порядком обработки сигналов прерывания. Второй способ, дополняя первый, позволяет применять различные дисциплины обслуживания прерываний.

Наличие сигнала прерывания не обязательно должно вызывать прерывание исполняющейся программы.

Процессор может обладать средствами защиты от прерываний:

1. отключение системы прерываний,
2. маскирование (запрет) отдельных сигналов прерывания.

Рис. 35. Распределение прерываний по уровням приоритета

Программное управление этими средствами (существуют специальные команды для управления работой системы прерываний) **позволяет операционной системе:**

- ✓ регулировать обработку сигналов прерывания, заставляя процессор обрабатывать их сразу по приходу;
- ✓ откладывать обработку на некоторое время;
- ✓ полностью игнорировать прерывания.

Обычно операция прерывания выполняется только после завершения выполнения текущей команды. Поскольку сигналы прерывания возникают в произвольные моменты времени, то на момент прерывания может существовать несколько сигналов прерывания, которые могут быть обработаны только последовательно. **Чтобы обработать сигналы прерывания в разумном порядке, им присваиваются приоритеты. Сигнал с более высоким приоритетом обрабатывается в первую очередь, обработка остальных сигналов прерывания откладывается.**

Программное управление специальными регистрами маски (маскирование сигналов прерывания) позволяет реализовать различные дисциплины обслуживания:

С относительными приоритетами, то есть обслуживание не прерывается даже при наличии запросов с более высокими приоритетами. После окончания обслуживания данного запроса обслуживается запрос с наивысшим приоритетом. Для организации такой дисциплины необходимо в программе обслуживания данного запроса наложить маски на все остальные сигналы прерывания или просто отключить систему прерываний.

С абсолютными приоритетами, то есть всегда обслуживается прерывание с наивысшим приоритетом. Для реализации этого режима необходимо на время обработки прерывания замаскировать все запросы с более низким приоритетом. При этом возможно многоуровневое прерывание, то есть прерывание программ обработки прерываний. Число уровней прерывания в этом режиме изменяется и зависит от приоритета запроса.

По принципу стека, или, как иногда говорят, **по дисциплине LCFS** (Last Come First Served - последним пришел, первым обслужен), то есть запросы с более низким приоритетом могут прерывать обработку прерывания с более высоким приоритетом. Для этого необходимо не накладывать маску ни на один из сигналов прерывания и не выключать систему прерываний.

Следует особо отметить, что для правильной реализации последних двух дисциплин нужно обеспечить полное маскирование системы прерываний при выполнении шагов 1-4 и 6-7. Это необходимо для того, чтобы не потерять запрос и правильно его обслужить. Многоуровневое прерывание должно происходить на этапе собственно обработки прерывания, а не на этапе перехода с одного процесса вычислений на другой.

Управление ходом выполнения задач со стороны операционной системы заключается в организации реакций на прерывания, в организации обмена информацией (данными и программами), в предоставлении необходимых ресурсов, в динамике выполнения задачи и в организации сервиса. **Причины прерываний определяет операционная система (модуль, который называют супервизором прерываний**), она же и выполняет действия, необходимые при данном прерывании и в данной ситуации. Поэтому **в состав любой операционной системы реального времени прежде всего входят программы:**

- ✓ управления системой прерываний,
- ✓ контроля состояний задач и событий,
- ✓ синхронизации задач,

- ✓ средства распределения памяти и управления ею,
- ✓ средства организации данных (с помощью файловых систем) и т.д.

Следует, однако, заметить, что современная операционная система реального времени должна вносить в аппаратно-программный комплекс нечто большее, нежели просто обеспечение быстрой реакции на прерывания.

Как мы уже знаем, при появлении запроса на прерывание система прерываний идентифицирует сигнал и, если прерывания разрешены, то управление передается на соответствующую подпрограмму обработки. Из рис. 34. видно, что ***в подпрограмме обработки прерывания имеется две служебные секции:***

- ✓ *первая секция, в которой осуществляется сохранение контекста прерываемых вычислений, который не смог быть сохранен на шаге 2,*
- ✓ *последняя, заключительная секция, в которой, наоборот, осуществляется восстановление контекста.*

Для того чтобы система прерываний не среагировала повторно на сигнал запроса на прерывание, она обычно автоматически «закрывает» (отключает) прерывания, поэтому необходимо потом в подпрограмме обработки прерываний вновь включать систему прерываний. В соответствии с рассмотренными режимами обработки прерываний (с относительными и абсолютными приоритетами и по правилу LCFS) установка этих режимов осуществляется в конце первой секции подпрограммы обработки. Таким образом, на время выполнения центральной секции (в случае работы в режимах с абсолютными приоритетами и по дисциплине LCFS) прерывания разрешены. На время работы заключительной секции подпрограммы обработки система прерываний вновь должна быть отключена и после восстановления контекста опять включена. Поскольку эти действия необходимо выполнять практически в каждой подпрограмме обработки прерываний, ***во многих операционных системах первые секции подпрограмм обработки прерываний выделяются в уже упоминавшийся специальный системный программный модуль, называемый супервизором прерываний****.*

Супервизор прерываний:

1. прежде всего сохраняет в дескрипторе текущей задачи рабочие регистры процессора, определяющие контекст прерываемого вычислительного процесса.

2. Далее он определяет ту подпрограмму, которая должна выполнить действия, связанные с обслуживанием настоящего (текущего) запроса на прерывание.
3. Наконец, перед тем, как передать управление на эту подпрограмму, супервизор прерываний устанавливает необходимый режим обработки прерывания.

После выполнения подпрограммы обработки прерывания управление вновь передается ядру операционной системы. На этот раз уже на тот модуль, который занимается диспетчеризацией задач. И уже диспетчер задач, в свою очередь, в соответствии с принятой дисциплиной распределения процессорного времени (между выполняющимися вычислительными процессами) восстановит контекст той задачи, которой будет решено выделить процессор (рис. 36).

Как мы видим из рисунка, здесь отсутствует возврат в прерванную ранее программу непосредственно из самой подпрограммы обработки прерывания. Для прямого возврата достаточно адрес возврата сохранить в стеке, что и делает аппаратура процессора. При этом стек легко обеспечивает возможность возврата в случае вложенных прерываний, поскольку он всегда реализует дисциплину LCFS.

Однако если бы контекст вычислительных процессов сохранялся просто в стеке, как это обычно реализуется аппаратурой, а не в специальных структурах данных, называемых **дескрипторами**, то у нас не было бы возможности гибко подходить к выбору той задачи, которой нужно передать процессор после завершения работы подпрограммы обработки прерывания.

Естественно, что это только общий принцип. В конкретных процессорах и в конкретных операционных системах могут существовать некоторые отступления от рассмотренной схемы и/или дополнения. Например, в современных процессорах часто имеются специальные аппаратные возможности для сохранения контекста прерываемого вычислительного процесса непосредственно в его дескрипторе, то есть дескриптор процесса (по крайней мере, его часть) становится структурой, которую поддерживает аппаратура.

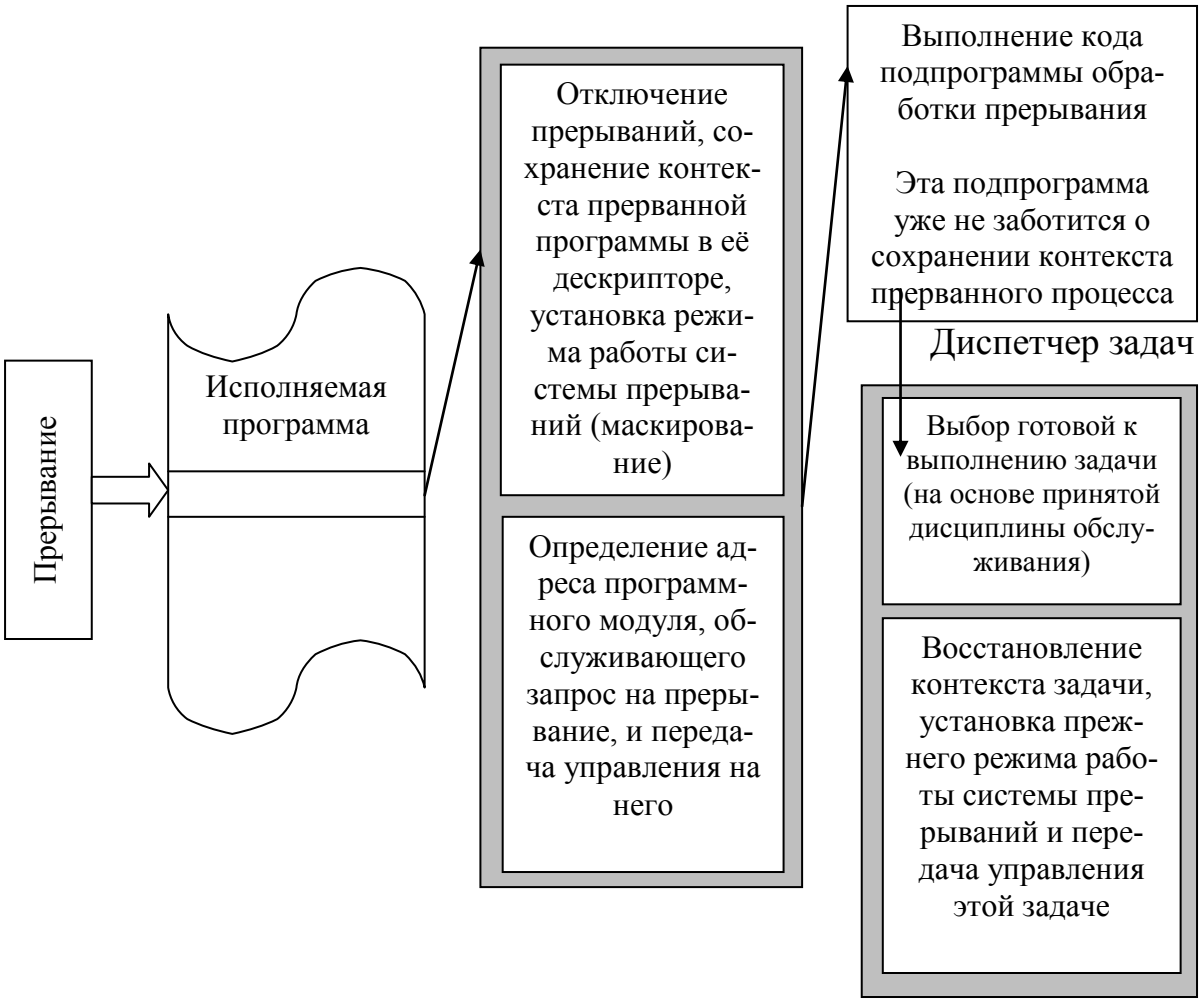

Рис. 36. Обработка прерывания при участии супервизоров ОС

Контрольные вопросы:

1. Дать определение понятию «прерывание».
2. Описать пошагово механизм обработки прерываний.
3. Перечислить виды прерываний. Привести примеры каждого вида прерываний.
4. Перечислить и охарактеризовать дисциплины обслуживания прерываний.
5. Пояснить, для чего используются приоритеты прерываний. Перечислить средства защиты от прерываний.
6. Описать алгоритм работы супервизора прерываний.

Лекция 19
Управление вводом-выводом

В состав любой операционной системы входят программные модули, обеспечивающие управление устройствами ввода-вывода ЭВМ. Эти программные модули называют драйверами устройств, а совокупность драйверов ввода-вывода образует систему ввода-вывода, входящую в состав операционной системы.

Драйвер устройства (Device driver) — программа, обеспечивающая взаимодействие операционной системы с физическим устройством.

Система ввода-вывода (Input-Output System) — часть операционной системы, обеспечивающая управление внешними устройствами, подключенными к ЭВМ.

Основной задачей системы ввода-вывода является обеспечение непрерывной организации (планирования, управления) и двусторонней передачи данных между основной памятью и внешними устройствами с целью достижения максимального перекрытия во времени работы этой аппаратуры и процессора.

Состав систем ввода-вывода и, следовательно, перечень драйверов устройств в различных операционных системах не совпадают, что объясняется имеющимися отличиями в аппаратуре ввода-вывода, а также множеством методов, используемых для управления этой аппаратурой. Вместе с тем в большинстве операционных систем существует некоторое ядро системы ввода-вывода, получившее название базовой системы ввода-вывода.

Базовая система ввода-вывода (BIOS — Basic Input Output System) — часть программного обеспечения ЭВМ, поддерживающая управление адаптерами внешних устройств и представляющая стандартный интерфейс для обеспечения переносимости операционных систем между ЭВМ с одинаковым процессором. Базовая система ввода-вывода, как правило, разрабатывается изготовителем ЭВМ, хранится в постоянном запоминающем устройстве и рассматривается как часть ЭВМ.

При построении систем ввода-вывода аппаратура ввода-вывода рассматривается как совокупность аппаратурных процессоров, которые способны работать параллельно и независимо друг от друга, а также относительно центрального процессора. На таких процессорах развиваются так называемые внешние процессы.

Внешние процессы, используя аппаратуру ввода-вывода, могут взаимодействовать как между собой, так и с внутренними процессами, которые развиваются на центральном процессоре. Важным фактом является то, что

скорости развития внешних и внутренних процессов существенно различаются, причем эти различия могут достигать нескольких порядков.

Система управления вводом-выводом представляет собой один или несколько системных процессов (т.е. процессов, принадлежащих операционной системе), обеспечивающих информационное и управляющее взаимодействие внутренних и внешних процессов. Через эту систему происходит инициализация, управление развитием и уничтожение внешних процессов.

С точки зрения внутренних (программных) процессов-пользователей система управления вводом-выводом представляет собой программный интерфейс с необходимыми для этих процессов внешними устройствами. В составе этого интерфейса пользователь имеет возможность выражать запросы на выполнение действий в отношении внешних устройств. При этом различают три типа действий:

- ✓ операции чтения и записи данных,
- ✓ операции управления устройством,
- ✓ операции по проверке состояния устройств.

При построении систем управления вводом-выводом руководствуются стремлением сделать большинство ее компонентов «невидимыми» для пользователей. что достигается созданием развитых драйверов внешних устройств с понятным интерфейсом и доступными из любой системы программирования.

Для сглаживания эффекта несоответствия скоростей между внутренними и внешними процессами в системах управления вводом-выводом применяют три основных метода:

- ✓ синхронизация по прерываниям ввода-вывода,
- ✓ буферизация ввода-вывода,
- ✓ блокирование данных.

Для синхронизации параллельной работы могут применяться различные методы, среди которых наиболее совершенными являются средства, основанные на использовании системы прерывания. Канал ввода-вывода через систему прерываний прерывает работу центрального процессора всякий раз при завершении операции ввода-вывода или при возникновении ошибки. Такие сигналы прерывания являются по своему смыслу синхронизирующими, так как они используются для оповещения определенного внутреннего процесса о событии, которое произошло при работе канала ввода-вывода или внешнего устройства.

Одной из главных функций ОС является управление всеми устройствами ввода-вывода компьютера. ОС должна передавать устройствам команды, перехватывать прерывания и обрабатывать ошибки; она также долж-

на обеспечивать интерфейс между устройствами и остальной частью системы. В целях развития интерфейс должен быть одинаковым для всех типов устройств (независимость от устройств).

Физическая организация устройств ввода-вывода

Устройства ввода-вывода делятся на два типа:

- ✓ блок-ориентированные устройства,
- ✓ байт-ориентированные устройства.

Блок-ориентированные устройства хранят информацию в блоках фиксированного размера, каждый из которых имеет свой собственный адрес. **Самое распространенное блок-ориентированное устройство — диск. Байт-ориентированные устройства** не адресуемы и не позволяют производить операцию поиска, они генерируют или потребляют последовательность байтов. **Примерами являются терминалы, строчные принтеры, сетевые адаптеры.** Однако некоторые внешние устройства не относятся ни к одному классу, например, часы, которые, с одной стороны, не адресуемы, а с другой стороны, не порождают потока байтов. Это устройство только выдает сигнал прерывания в некоторые моменты времени.

Внешнее устройство обычно состоит из механического и электронного компонента. **Электронный компонент называется контроллером устройства или адаптером. Механический компонент представляет собственно устройство.** Некоторые контроллеры могут управлять несколькими устройствами. Если интерфейс между контроллером и устройством стандартизован, то независимые производители могут выпускать совместимые как контроллеры, так и устройства.

Операционная система обычно имеет дело не с устройством, а с контроллером. Контроллер, как правило, выполняет простые функции, например преобразует поток бит в блоки, состоящие из байт, и осуществляют контроль и исправление ошибок. Каждый контроллер имеет несколько регистров, которые используются для взаимодействия с центральным процессором. В некоторых компьютерах эти регистры являются частью физического адресного пространства. В таких компьютерах нет специальных операций ввода-вывода. В других компьютерах адреса регистров ввода-вывода, называемых часто портами, образуют собственное адресное пространство за счет введения специальных операций ввода-вывода (например, команд IN и OUT в процессорах 186).

ОС выполняет ввод-вывод, записывая команды в регистры контроллера. Например, контроллер гибкого диска IBM PC принимает 15 команд, таких

как **READ, WRITE, SEEK, FORMAT** и т.д. Когда команда принята, процессор оставляет контроллер и занимается другой работой. При завершении команды контроллер организует прерывание для того, чтобы передать управление процессором операционной системе, которая должна проверить результаты операции. Процессор получает результаты и статус устройства, читая информацию из регистров контроллера.

Организация программного обеспечения ввода-вывода

Уровни организации программного обеспечения ввода-вывода

Основная идея организации программного обеспечения ввода-вывода состоит в разбиении его на несколько уровней, причем нижние уровни обеспечивают экранирование особенностей аппаратуры от верхних, а те, в свою очередь, обеспечивают удобный интерфейс для пользователей.

Ключевым принципом является независимость от устройств. Вид программы не должен зависеть от того, читает ли она данные с гибкого диска или с жесткого диска.

Очень близкой к идее независимости от устройств является идея единообразного именования, т.е. для именования устройств должны быть приняты единые правила.

Другим важным вопросом для программного обеспечения ввода - вывода является обработка ошибок. Вообще говоря, ошибки следует обрабатывать как можно ближе к аппаратуре. Если контроллер обнаруживает ошибку чтения, то он должен попытаться ее скорректировать. Если же это ему не удается, то исправлением ошибок должен заняться драйвер устройства. Многие ошибки могут исчезать при повторных попытках выполнения операций ввода-вывода, например, ошибки, вызванные наличием пылинок на головках чтения или на диске. И только если нижний уровень не может справиться с ошибкой, он сообщает об ошибке верхнему уровню.

Еще один ключевой вопрос — это использование блокирующих (синхронных) и неблокирующих (асинхронных) передач. Большинство операций физического ввода-вывода выполняется асинхронно — процессор начинает передачу и переходит на другую работу, пока не наступает прерывание. Пользовательские программы намного легче писать, если операции ввода-вывода блокирующие — после команды READ программа автоматически приостанавливается до тех пор, пока данные не попадут в буфер программы. ОС выполняет операции ввода-вывода асинхронно, но представляет их для пользовательских программ в синхронной форме.

Последняя проблема состоит в том, что одни устройства являются раз-

деляемыми, а другие — выделенными. Диски — это разделяемые устройства, так как одновременный доступ нескольких пользователей к диску не представляет собой проблему. Принтеры — это выделенные устройства, потому что нельзя смешивать строчки, печатаемые различными пользователями. Наличие выделенных устройств создает для операционной системы некоторые проблемы.

Для решения поставленных проблем целесообразно разделить программное обеспечение ввода-вывода на четыре слоя (рис. 37):

- ✓ обработка прерываний,
- ✓ драйверы устройств,
- ✓ независимый от устройств слой операционной системы,
- ✓ пользовательский слой программного обеспечения.

Рис. 37. Многоуровневая организация программного обеспечения системы ввода-вывода

Обработка прерываний. Прерывания должны быть скрыты как можно глубже в недрах операционной системы, чтобы как можно меньшая часть ОС имела с ними дело. Наилучший способ состоит в разрешении процессу, инициировавшему операцию ввода-вывода, блокировать себя до завершения операции и наступления прерывания. Процесс может блокировать себя, используя, например, вызов DOWN для семафора, или вызов WAIT для пере-

менной условия, или вызов RECEIVE для ожидания сообщения. При наступлении прерывания процедура обработки прерывания выполняет разблокирование процесса, инициировавшего операцию ввода-вывода, используя вызовы UP, SIGNAL или посылая процессу сообщение. В любом случае эффект от прерывания будет состоять в том, что ранее заблокированный процесс теперь продолжит свое выполнение.

Драйверы устройств. Весь зависимый от устройства код помещается в драйвер устройства. Каждый драйвер управляет устройствами одного типа или, может быть, одного класса.

В операционной системе только драйвер устройства знает о конкретных особенностях какого-либо устройства. Например, только драйвер диска имеет дело с дорожками, секторами, цилиндрами, временем установления головки и другими факторами, обеспечивающими правильную работу диска.

Драйвер устройства принимает запрос от устройств программного слоя и решает, как его выполнить. Типичным запросом является чтение ***n*** блоков данных. Если драйвер был свободен во время поступления запроса, то он начинает выполнять запрос немедленно. Если же он был занят обслуживанием другого запроса, то вновь поступивший запрос присоединяется к очереди уже имеющихся запросов, и он будет выполнен, когда наступит его очередь.

Первый шаг в реализации запроса ввода-вывода, например, для диска, состоит в преобразовании его из абстрактной формы в конкретную. Для дискового драйвера это означает преобразование номеров блоков в номера цилиндров, головок, секторов, проверку, работает ли мотор, находится ли головка над нужным цилиндром. Короче говоря, он должен решить, какие операции контроллера нужно выполнить и в какой последовательности.

После передачи команды контроллеру драйвер должен решить, блокировать ли себя до окончания заданной операции или нет. Если операция занимает значительное время, как при печати некоторого блока данных, то драйвер блокируется до тех пор, пока операция не завершится, и обработчик прерывания не разблокирует его. Если команда ввода-вывода выполняется быстро (например, прокрутка экрана), то драйвер ожидает ее завершения без блокирования.

Независимый от устройств слой операционной системы. Большая часть программного обеспечения ввода-вывода является независимой от устройств. Точная граница между драйверами и независимыми от устройств программами определяется системой, так как некоторые функции, которые

могли бы быть реализованы независимым способом, в действительности выполнены в виде драйверов для повышения эффективности или по другим причинам.

Типичными функциями для независимого от устройств слоя являются:

- ✓ обеспечение общего интерфейса к драйверам устройств;
- ✓ именование устройств;
- ✓ защита устройств;
- ✓ обеспечение независимого размера блока;
- ✓ буферизация;
- ✓ распределение памяти на блок-ориентированных устройствах;
- ✓ распределение и освобождение выделенных устройств;
- ✓ уведомление об ошибках.

Остановимся на некоторых функциях данного перечня. Верхним слоям программного обеспечения неудобно работать с блоками разной величины, поэтому данный слой обеспечивает единый размер блока, например, за счет объединения нескольких различных блоков в единый логический блок. В связи с этим верхние уровни имеют дело с абстрактными устройствами, которые используют единый размер логического блока независимо от размера физического сектора.

При создании файла или заполнении его новыми данными необходимо выделить ему новые блоки. Для этого ОС должна вести список или битовую карту свободных блоков диска. На основании информации о наличии свободного места на диске может быть разработан алгоритм поиска свободного блока, независимый от устройства реализуемый программным слоем, находящимся выше слоя драйверов.

Пользовательский слой программного обеспечения. Хотя большая часть программного обеспечения ввода-вывода находится внутри ОС, некоторая его часть содержится в библиотеках, связываемых с пользовательскими программами. Системные вызовы, включающие вызовы ввода-вывода, обычно делаются библиотечными процедурами. Если программа, написанная на языке С, содержит вызов

count = write (fd, buffer, nbytes),

то библиотечная процедура write будет связана с программой. Набор подобных процедур является частью системы ввода-вывода. В частности, форматирование ввода или вывода выполняется библиотечными процедурами. Примером может служить функция printf языка С, которая принимает строку формата и, возможно, некоторые переменные в качестве входной ин-

формации, затем строит строку символов и делает вызов write для вывода этой строки. Стандартная библиотека ввода-вывода содержит большое число процедур, которые выполняют ввод-вывод и работают как часть пользовательской программы.

Другой категорией программного обеспечения ввода-вывода является **подсистема спулинга** (spooling). **Спулинг — это способ работы с выделенными устройствами в мультипрограммной системе.** Рассмотрим типичное устройство, требующее спулинга — принтер. Хотя технически легко позволить каждому пользовательскому процессу открыть специальный файл, связанный с принтером, такой способ опасен из-за того, что пользовательский процесс может монополизировать принтер на произвольное время. Вместо этого создается специальный процесс — монитор, который получает исключительные права на использование этого устройства. Также создается специальный каталог, называемый каталогом спулинга. Для того чтобы напечатать информацию, пользовательский процесс размещает ее в файле, а файл помещает в каталог спулинга. Процесс-монитор по очереди распечатывает все файлы, содержащиеся в каталоге спулинга.

Контрольные вопросы:

1. Пояснить назначение системы ввода-вывода
2. Охарактеризовать функции базовой системы ввода-вывода
3. Охарактеризовать функции блок-ориентированных устройств ввода-вывода
4. Охарактеризовать функции байт-ориентированных устройств ввода-вывода
5. Охарактеризовать уровни организации программного обеспечения ввода-вывода
6. Пояснить функции драйверов устройств
7. Пояснить функции пользовательского слоя программного обеспечения
8. Пояснить, что такое спулинг

Лекция 20
Основные понятия и положения защиты информации в КС

В октябре 1988 г. в США произошло событие, названное специалистами крупнейшим нарушением безопасности американских компьютерных систем из когда-либо случавшихся. 23-летний студент выпускного курса Корнельского университета Роберт Т. Моррис запустил в компьютерной сети ARPANET программу, представлявшую собой редко встречающуюся разновидность компьютерных вирусов – сетевых "червей". В результате атаки был полностью или частично заблокирован ряд общенациональных компьютерных сетей, в частности Internet, CSnet, NSFnet, BITnet, ARPANET и несекретная военная сеть Milnet. В итоге вирус поразил более 6200 компьютерных систем по всей Америке, включая системы многих крупнейших университетов, институтов, правительственных лабораторий, частных фирм, военных баз, клиник, агентства NASA. Общий ущерб от этой атаки оценивается специалистами минимум в 100 млн. долл. Р. Моррис был исключен из университета с правом повторного поступления через год и приговорен судом к штрафу в 270 тыс. долл. и трем месяцам тюремного заключения.

Важность решения проблемы информационной безопасности в настоящее время общепризнана, подтверждением чему служат громкие процессы о нарушении целостности систем. Убытки ведущих компаний в связи с нарушениями безопасности информации составляют триллионы долларов, причем только треть опрошенных компаний смогли определить количественно размер потерь. Проблема обеспечения безопасности носит комплексный характер, для ее решения необходимо сочетание законодательных, организационных и программно-технических мер.

Таким образом, обеспечение информационной безопасности требует системного подхода и нужно использовать разные средства и приемы – морально-этические, законодательные, административные и технические. Нас будут интересовать последние. Технические средства реализуются программным и аппаратным обеспечением и решают разные задачи по защите, они могут быть встроены в операционные системы либо могут быть реализованы в виде отдельных продуктов. Во многих случаях центр тяжести смещается в сторону защищенности операционных систем.

Есть несколько причин для реализации дополнительных средств защиты. Наиболее очевидная – помешать внешним попыткам нарушить доступ к

конфиденциальной информации. Не менее важно, однако, гарантировать, что каждый программный компонент в системе использует системные ресурсы только способом, совместимым с установленной политикой применения этих ресурсов. Такие требования абсолютно необходимы для надежной системы. Кроме того, наличие защитных механизмов может увеличить надежность системы в целом за счет обнаружения скрытых ошибок интерфейса между компонентами системы. Раннее обнаружение ошибок может предотвратить "заражение" неисправной подсистемой остальных.

Политика в отношении ресурсов может меняться в зависимости от приложения и с течением времени. Операционная система должна обеспечивать прикладные программы инструментами для создания и поддержки защищенных ресурсов. Здесь реализуется важный для гибкости системы принцип – отделение политики от механизмов. Механизмы определяют, как может быть сделано что-либо, тогда как политика решает, что должно быть сделано. Политика может меняться в зависимости от места и времени. Желательно, чтобы были реализованы по возможности общие механизмы, тогда как изменение политики требует лишь модификации системных параметров или таблиц.

К сожалению, построение защищенной системы предполагает необходимость склонить пользователя к отказу от некоторых интересных возможностей.

Угрозы безопасности

Знание возможных угроз, а также уязвимых мест защиты, которые эти угрозы обычно эксплуатируют, необходимо для того, чтобы выбирать наиболее экономичные средства обеспечения безопасности.

Считается, что **безопасная система должна обладать свойствами конфиденциальности, доступности и целостности. Любое потенциальное действие, которое направлено на нарушение конфиденциальности, целостности и доступности информации, называется угрозой. Реализованная угроза называется атакой.**

Конфиденциальная (confidentiality) система обеспечивает уверенность в том, что секретные данные будут доступны только тем пользователям, которым этот доступ разрешен (такие пользователи называются **автори-**

зованными). **Под доступностью (availability) понимают гарантию того, что авторизованным пользователям всегда будет доступна информация, которая им необходима.** И наконец, **целостность (integrity) системы подразумевает, что неавторизованные пользователи не могут каким-либо образом модифицировать данные.**

Защита информации ориентирована на борьбу с так называемыми умышленными угрозами, то есть с теми, которые, в отличие от случайных угроз (ошибок пользователя, сбоев оборудования и др.), преследуют цель нанести ущерб пользователям ОС.

Умышленные угрозы подразделяются на активные и пассивные. Пассивная угроза – несанкционированный доступ к информации без изменения состояния системы, **активная** – несанкционированное изменение системы. Пассивные атаки труднее выявить, так как они не влекут за собой никаких изменений данных. Защита против пассивных атак базируется на средствах их предотвращения.

Можно выделить **несколько типов угроз.** Наиболее распространенная угроза – **попытка проникновения в систему под видом легального пользователя,** например, попытки угадывания и подбора паролей. Более сложный вариант – **внедрение в систему программы, которая выводит на экран слово login.** Многие легальные пользователи при этом начинают пытаться входить в систему, и их попытки могут протоколироваться. Такие безобидные с виду программы, выполняющие нежелательные функции, называются "троянскими конями". Иногда удается торпедировать работу программы проверки пароля путем многократного нажатия клавиш del, break, cancel и т. д. Для защиты от подобных атак ОС запускает процесс, называемый **аутентификацией пользователя**.

Угрозы другого рода связаны с нежелательными действиями легальных пользователей, которые могут, например, предпринимать попытки чтения страниц памяти, дисков и лент, которые сохранили информацию, связанную с предыдущим использованием. Защита в таких случаях базируется **на надежной системе авторизации**. В эту категорию также попадают атаки типа отказ в обслуживании, когда сервер затоплен мощным потоком запросов и становится фактически недоступным для отдельных авторизованных пользователей.

Наконец, функционирование системы может быть нарушено с помощью программ-вирусов или программ – «червей», которые специально предназначены для того, чтобы причинить вред или недолжным образом использовать ресурсы компьютера. Общее название угроз такого рода – вредоносные программы (malicious software). Обычно они распространяются сами по себе, переходя на другие компьютеры через зараженные файлы, дискеты или по электронной почте. Наиболее эффективный способ борьбы с подобными программами – соблюдение правил «компьютерной гигиены». Многопользовательские компьютеры меньше страдают от вирусов по сравнению с персональными, поскольку там имеются системные средства защиты.

Таковы основные угрозы, на долю которых приходится львиная доля ущерба, наносимого информационным системам.

Формализация подхода к обеспечению информационной безопасности

Проблема информационной безопасности оказалась настолько важной, что в ряде стран были выпущены основополагающие документы, в которых регламентированы основные подходы к проблеме информационной безопасности. В результате оказалось возможным ранжировать информационные системы по степени надежности.

Наиболее известна оранжевая (по цвету обложки) книга Министерства обороны США [DoD, 1993]. В этом документе определяется четыре уровня безопасности – D, C, B и A. По мере перехода от уровня D до A к надежности систем предъявляются все более жесткие требования. Уровни C и B подразделяются на классы (C1, C2, B1, B2, B3). Чтобы система в результате процедуры сертификации могла быть отнесена к некоторому классу, ее защита должна удовлетворять оговоренным требованиям.

В качестве примера рассмотрим требования класса C2, которому удовлетворяют ОС Windows NT, отдельные реализации Unix и ряд других.

Каждый пользователь должен быть идентифицирован уникальным входным именем и паролем для входа в систему. Доступ к компьютеру предоставляется лишь после аутентификации.

Система должна быть в состоянии использовать эти уникальные идентификаторы, чтобы следить за действиями пользователя (управление избира-

тельным доступом). Владелец ресурса (например, файла) должен иметь возможность контролировать доступ к этому ресурсу.

Операционная система должна защищать объекты от повторного использования. Перед выделением новому пользователю все объекты, включая память и файлы, должны инициализироваться.

Системный администратор должен иметь возможность вести учет всех событий, относящихся к безопасности.

Система должна защищать себя от внешнего влияния или навязывания, такого как модификация загруженной системы или системных файлов, хранящихся на диске.

Сегодня на смену оранжевой книге пришел стандарт Common Criteria, а набор критериев Controlled Access Protection Profile сменил критерии класса C2.

По существу, проектирование системы безопасности подразумевает ответы на следующие вопросы: какую информацию защищать, какого рода атаки на безопасность системы могут быть предприняты, какие средства использовать для защиты каждого вида информации? Поиск ответов на данные вопросы называется **формированием политики безопасности**, которая помимо чисто технических аспектов включает также и решение организационных проблем.

Формируя политику безопасности, необходимо учитывать несколько базовых принципов. Так, Зальтцер (Saltzer) и Шредер (Schroeder) (1975) на основе своего опыта работы с MULTICS сформулировали следующие рекомендации для проектирования системы безопасности ОС:

1. Проектирование системы должно быть открытым. Нарушитель и так все знает (криптографические алгоритмы открыты).
2. Не должно быть доступа по умолчанию. Ошибки с отклонением легитимного доступа будут обнаружены скорее, чем ошибки там, где разрешен неавторизованный доступ.
3. Нужно тщательно проверять текущее авторство. Так, многие системы проверяют привилегии доступа при открытии файла и не делают этого после. В результате пользователь может открыть файл и держать его открытым в течение недели и иметь к нему доступ, хотя владелец уже сменил защиту.
4. Давать каждому процессу минимум возможных привилегий.

5. Защитные механизмы должны быть просты, постоянны и встроены в нижний слой системы, это не аддитивные добавки (известно много неудачных попыток "улучшения" защиты слабо приспособленной для этого ОС MS-DOS).
6. Важна физиологическая приемлемость. Если пользователь видит, что защита требует слишком больших усилий, он от нее откажется. Ущерб от атаки и затраты на ее предотвращение должны быть сбалансированы.

Приведенные соображения показывают необходимость продумывания и встраивания защитных механизмов на самых ранних стадиях проектирования системы.

Информационная безопасность относится к числу дисциплин, развивающихся чрезвычайно быстрыми темпами. Только комплексный, систематический, современный подход способен успешно противостоять нарастающим угрозам.

Ключевые понятия информационной безопасности: конфиденциальность, целостность и доступность информации, а любое действие, направленное на их нарушение, называется угрозой.

Основные понятия информационной безопасности регламентированы в основополагающих документах.

Угроза безопасности компьютерной системы - это потенциально возможное происшествие, которое может оказать нежелательное воздействие на саму систему, а также на информацию, хранящуюся в ней.

Уязвимость компьютерной системы - это некая ее неудачная характеристика, которая делает возможным возникновение угрозы.

Атака на компьютерную систему - это действие, предпринимаемое злоумышленником, которое заключается в поиске и использовании той или иной уязвимости.

Исследователи обычно выделяют **три основных вида угроз безопасности - это угрозы раскрытия, целостности и отказа в обслуживании.**

Угроза раскрытия заключается том, что информация становится известной тому, кому не следовало бы ее знать. Иногда вместо слова "раскрытие" используются термины "кража" или "утечка".

Угроза целостности включает в себя любое умышленное изменение данных, хранящихся в вычислительной системе или передаваемых из одной системы в другую. Обычно считается, что угрозе раскрытия подвержены в большей степени государственные структуры, а угрозе целостности - деловые или коммерческие.

Угроза отказа в обслуживании возникает всякий раз, когда в результате некоторых действий блокируется доступ к некоторому ресурсу вычислительной системы. Реально блокирование может быть постоянным, так чтобы запрашиваемый ресурс никогда не был получен, или оно может вызвать только задержку запрашиваемого ресурса, достаточно долгую для того, чтобы он стал бесполезным. В таких случаях говорят, что ресурс исчерпан.

В локальных вычислительных системах (ВС) наиболее частыми являются угрозы раскрытия и целостности, а в глобальных на первое место выходит угроза отказа в обслуживании.

Особенности безопасности компьютерных сетей

Основной особенностью любой сетевой системы является то, что её компоненты распределены в пространстве и связь между ними физически осуществляется при помощи сетевых соединений (коаксиальный кабель, витая пара, оптоволокно и т. п.) и программно при помощи механизма сообщений. При этом все управляющие сообщения и данные, пересылаемые между объектами распределенной вычислительной системы, передаются по сетевым соединениям в виде пакетов обмена.

Сетевые системы характерны тем, что, наряду с обычными (локальными) атаками, осуществляемыми в пределах одной компьютерной системы, к ним применим специфический вид атак, обусловленный распределенностью ресурсов и информации в пространстве. Это так называемые сетевые (или удалённые) атаки (remote или network attacks). Они характеризуются, во-первых, тем, что злоумышленник может находиться за тысячи километров от атакуемого объекта, и, во-вторых, тем, что нападению может подвергаться не конкретный компьютер, а информация, передающаяся по сетевым соединениям. С развитием локальных и глобальных сетей именно удалённые атаки становятся лидирующими как по количеству попыток, так и по успешности их применения и, соответственно, обеспечение безопасности ВС с точки зрения противостояния удалённым атакам приобретает первостепенное значение.

Классификация компьютерных атак

Формы организации атак весьма разнообразны, но в целом все они принадлежат к одной из следующих категорий:

- ✓ **Удаленное проникновение в компьютер**: программы, которые получают неавторизованный доступ к другому компьютеру через Интернет (или локальную сеть);
- ✓ **Локальное проникновение в компьютер**: программы, которые получают неавторизованный доступ к компьютеру, на котором они работают;
- ✓ **Удаленное блокирование компьютера**: программы, которые через Интернет (или сеть) блокируют работу всего удаленного компьютера или отдельной программы на нем;
- ✓ **Локальное блокирование компьютера**: программы, которые блокируют работу компьютера, на котором они работают;
- ✓ **Сетевые сканеры**: программы, которые осуществляют сбор информации о сети, чтобы определить, какие из компьютеров и программ, работающих на них, потенциально уязвимы к атакам;
- ✓ **Сканеры уязвимых мест программ**: программы, проверяют большие группы компьютеров в Интернет в поисках компьютеров, уязвимых к тому или иному конкретному виду атаки;
- ✓ **Вскрыватели паролей**: программы, которые обнаруживают легко угадываемые пароли в зашифрованных файлах паролей;
- ✓ **Сетевые анализаторы**: программы, которые слушают сетевой трафик. Часто в них имеются возможности автоматического выделения имен пользователей, паролей и номеров кредитных карт из трафика;
- ✓ **Модификация передаваемых данных или подмена информации;**
- ✓ **Подмена доверенного объекта распределённой ВС** (работа от его имени) или ложный объект распределённой ВС (РВС).
- ✓ **Социальная инженерия - несанкционированный доступ к информации иначе, чем взлом программного обеспечения.** Цель - обхитрить людей для получения паролей к системе или иной информации, которая поможет нарушить безопасность системы;

Методы защиты от удалённых атак в сети Internet

Наиболее простыми и дешёвыми являются **административные методы защиты**, как то использование в сети стойкой криптографии, использование или неиспользование определённых операционных систем и другие методы.

Следующая группа методов защиты от удалённых атак - **программно-аппаратные.** К ним относятся:

- программно-аппаратные шифраторы сетевого трафика;
- защищённые сетевые криптопротоколы;
- программные средства обнаружения атак;
- программные средства анализа защищённости;
- защищённые сетевые ОС.

Здесь можно выделить подгруппу методов защиты - **программные методы.** К ним относятся прежде всего защищённые криптопротоколы, используя которые можно повысить надёжность защиты соединения.

Контрольные вопросы:

1. Пояснить, что такое безопасная система, угроза, атака
2. Перечислить и охарактеризовать основные типы угроз КС
3. Перечислить ключевые понятия информационной безопасности
4. Перечислить три основных вида угроз безопасности
5. Привести классификацию компьютерных атак
6. Перечислить и охарактеризовать методы защиты от удалённых атак в сети Internet

Лекция 21
Механизмы защиты операционных систем

Решение вопросов безопасности операционных систем обусловлено их архитектурными особенностями.

Основными задачами защиты операционных систем являются:

- ✓ идентификация,
- ✓ аутентификация,
- ✓ разграничение доступа пользователей к ресурсам,
- ✓ протоколирование и аудит самой системы.

Идентификация и аутентификация

Для начала рассмотрим проблему контроля доступа в систему. Наиболее распространенным способом контроля доступа является процедура регистрации. Обычно каждый пользователь в системе имеет уникальный идентификатор. Идентификаторы пользователей применяются с той же целью, что и идентификаторы любых других объектов, файлов, процессов. **Идентификация** заключается в сообщении пользователем своего идентификатора. Для того чтобы установить, что пользователь именно тот, за кого себя выдает, то есть что именно ему принадлежит введенный идентификатор, в информационных системах предусмотрена **процедура аутентификации** (authentication - опознавание, в переводе с латинского означает «установление подлинности»), задача которой - предотвращение доступа к системе нежелательных лиц.

Обычно аутентификация базируется на одном или более из трех пунктов:

- ✓ то, чем пользователь владеет (ключ или магнитная карта);
- ✓ то, что пользователь знает (пароль);
- ✓ атрибуты пользователя (отпечатки пальцев, подпись, голос).

Пароли, уязвимость паролей

Наиболее простой подход к аутентификации - **применение пользовательского пароля**.

Когда пользователь идентифицирует себя при помощи уникального идентификатора или имени, у него запрашивается пароль. Если пароль, сообщенный пользователем, совпадает с паролем, хранящимся в системе, система предполагает, что пользователь легитимен. Пароли часто используются для защиты объектов в компьютерной системе в отсутствие более сложных схем защиты.

Недостатки паролей связаны с тем, что трудно сохранить баланс между удобством пароля для пользователя и его надежностью. Пароли могут быть угаданы, случайно показаны или нелегально переданы авторизованным пользователем неавторизованному.

Есть два общих способа угадать пароль.

Один связан со сбором информации о пользователе. Люди обычно используют в качестве паролей очевидную информацию (например, имена животных или номерные знаки автомобилей). Для иллюстрации важности разумной политики назначения идентификаторов и паролей можно привести данные исследований, проведенных в AT&T, показывающие, что из 500 попыток несанкционированного доступа около 300 составляют попытки угадывания паролей или беспарольного входа по пользовательским именам guest, demo и т.д.

Другой способ - попытаться перебрать все наиболее вероятные комбинации букв, чисел и знаков пунктуации (атака по словарю). Например, четыре десятичные цифры дают только 10 000 вариантов, более длинные пароли, введенные с учетом регистра символов и пунктуации, не столь уязвимы, но тем не менее таким способом удается разгадать до 25% паролей. Чтобы заставить пользователя выбрать трудноугадываемый пароль, во многих системах внедрена реактивная проверка паролей, которая при помощи собственной программы-взломщика паролей может оценить качество пароля, введенного пользователем.

Несмотря на все это, пароли распространены, поскольку они удобны и легко реализуемы.

Шифрование пароля

Для хранения секретного списка паролей на диске во многих ОС используется **криптография**. Система задействует одностороннюю функцию,

которую просто вычислить, но для которой чрезвычайно трудно (разработчики надеются, что невозможно) подобрать обратную функцию.

Хранятся только кодированные пароли. В процессе аутентификации представленный пользователем пароль кодируется и сравнивается с хранящимися на диске. Таким образом, файл паролей нет необходимости держать в секрете.

При удаленном доступе к ОС нежелательна передача пароля по сети в открытом виде. Одним из типовых решений является использование криптографических протоколов. В качестве примера можно рассмотреть протокол опознавания с подтверждением установления связи путем вызова - CHAP (Challenge Handshake Authentication Protocol).

Опознавание достигается за счет проверки того, что у пользователя, осуществляющего доступ к серверу, имеется секретный пароль, который уже известен серверу.

В системах, работающих с большим количеством пользователей, когда хранение всех паролей затруднительно, **применяются для опознавания сертификаты, выданные доверенной стороной.**

Авторизация. Разграничение доступа к объектам ОС

После успешной регистрации система должна осуществлять **авторизацию** (authorization) - **предоставление субъекту прав на доступ к объекту.** Средства авторизации контролируют доступ легальных пользователей к ресурсам системы, предоставляя каждому из них именно те права, которые были определены администратором, а также осуществляют контроль возможности выполнения пользователем различных системных функций. Система контроля базируется на общей модели, называемой **матрицей доступа**.

Компьютерная система может быть смоделирована как набор субъектов (процессы, пользователи) и объектов.

Под объектами понимаются как ресурсы оборудования (процессор, сегменты памяти, принтер, диски), **так и программные ресурсы** (файлы, программы, семафоры), **то есть все то, доступ к чему контролируется. Каждый объект имеет уникальное имя, отличающее его от других объ-**

ектов в системе, и каждый из них может быть доступен через хорошо определенные и значимые операции.

Различают дискреционный (избирательный) способ управления доступом и полномочный (мандатный).

При дискреционном доступе определенные операции над конкретным ресурсом запрещаются или разрешаются субъектам или группам субъектов. Текущее состояние прав доступа при дискреционном управлении описывается матрицей, в строках которой перечислены субъекты, в столбцах - объекты, а в ячейках - операции, которые субъект может выполнить над объектом.

Полномочный подход заключается в том, что все объекты могут иметь уровни секретности, а все субъекты делятся на группы, образующие иерархию в соответствии с уровнем допуска к информации. Иногда такой подход называют **моделью многоуровневой безопасности,** которая должна обеспечивать выполнение следующих правил:

- ✓ **Простое свойство секретности.** Субъект может читать информацию только из объекта, уровень секретности которого не выше уровня секретности субъекта (например, генерал читает документы лейтенанта, но не наоборот).
- ✓ ***-свойство.** Субъект может записывать информацию в объекты только своего уровня или более высоких уровней секретности (например, генерал не может случайно разгласить нижним чинам секретную информацию)

Большинство операционных систем реализуют именно дискреционное управление доступом. Главное его достоинство - гибкость, основные недостатки - рассредоточенность управления и сложность централизованного контроля.

Домены безопасности

При реализации схемы дискреционного доступа используется концепция **домена безопасности** (protection domain).

Каждый домен определяет набор объектов и типов операций, которые могут производиться над каждым объектом. Возможность выпол-

нять операции над объектом есть права доступа, каждое из которых есть упорядоченная пара <object-name, rights-set>.

Домен – это набор прав доступа, например, если домен D имеет права доступа <file F, {read, write}>, это означает, что процесс, выполняемый в домене D, может читать или писать в файл F, но не может выполнять других операций над этим объектом

Пример доменов приведен на рис. 38.

Объект / Домен	F1	F2	F3	Printer
D1	read			
D2				print
D3		read	execute	
D4	read write		read write	

Рис. 38. Специфицирование прав доступа к ресурсам

Связь конкретных субъектов, функционирующих в операционных системах, может быть организована следующим образом:

- ✓ **Каждый пользователь может быть доменом.** В этом случае набор объектов, к которым может быть организован доступ, зависит от идентификации пользователя.
- ✓ **Каждый процесс может быть доменом.** В этом случае набор доступных объектов определяется идентификацией процесса.
- ✓ **Каждая процедура может быть доменом.** В этом случае набор доступных объектов соответствует локальным переменным, определенным внутри процедуры. Заметим, что, когда процедура выполнена, происходит смена домена.

Матрица доступа

Модель безопасности (рис. 1) имеет вид матрицы, которая называется **матрицей доступа**. Эту матрицу можно разложить по столбцам, в результате чего получаются **списки прав доступа (access control list - ACL).** В результате разложения по строкам получаются **мандаты возможностей** (capability list или capability tickets).

Список прав доступа

Каждая колонка в матрице может быть реализована как **список доступа для одного объекта.** Очевидно, что пустые клетки могут не учитываться. В результате для каждого объекта определяется список упорядоченных пар <domain, rights-set>, который определяет все домены с непустыми наборами прав для данного объекта.

Элементами списка могут быть процессы, пользователи или группы пользователей. При реализации широко применяется предоставление доступа по умолчанию для пользователей, права которых не указаны.

Мандаты возможностей

Если матрицу доступа хранить по строкам, то есть если каждый субъект хранит список объектов и для каждого объекта - список допустимых операций, то такой способ хранения называется **«мандаты»** или **«перечни возможностей»** (capability list). Каждый пользователь обладает несколькими мандатами и может иметь право передавать их другим. Мандаты могут быть рассеяны по системе и вследствие этого представлять большую угрозу для безопасности, чем списки контроля доступа. Их хранение должно быть тщательно продумано.

Выявление вторжений. Аудит системы защиты

Даже самая лучшая система защиты рано или поздно будет взломана. **Обнаружение попыток вторжения** является важнейшей задачей системы защиты, поскольку ее решение позволяет минимизировать ущерб от взлома и собирать информацию о методах вторжения. Как правило, поведение взломщика отличается от поведения легального пользователя. Иногда эти различия можно выразить количественно, например, подсчитывая число некорректных вводов пароля во время регистрации.

Основным инструментом выявления вторжений является запись данных аудита. Отдельные действия пользователей протоколируются, а полученный протокол используется для выявления вторжений.

Аудит, таким образом, заключается в регистрации специальных данных о различных типах событий, происходящих в системе и так или иначе влия-

ющих на состояние безопасности компьютерной системы. К числу таких событий относятся:

- ✓ вход или выход из системы;
- ✓ операции с файлами (открыть, закрыть, переименовать, удалить);
- ✓ обращение к удаленной системе;
- ✓ смена привилегий или иных атрибутов безопасности (режима доступа, уровня благонадежности пользователя и т. п.).

Если фиксировать все события, объем регистрационной информации, скорее всего, будет расти слишком быстро, а ее эффективный анализ станет невозможным. Следует предусматривать наличие средств выборочного протоколирования как в отношении пользователей, когда слежение осуществляется только за подозрительными личностями, так и в отношении событий. Слежка важна в первую очередь как профилактическое средство. Можно надеяться, что многие воздержатся от нарушений безопасности, зная, что их действия фиксируются.

Помимо протоколирования можно периодически сканировать систему на наличие слабых мест в системе безопасности. Такое сканирование может проверить разнообразные аспекты системы:

- ✓ короткие или легкие пароли;
- ✓ неавторизованные программы, если система поддерживает этот механизм;
- ✓ неавторизованные программы в системных директориях;
- ✓ долго выполняющиеся программы;
- ✓ нелогичная защита как пользовательских, так и системных директорий и файлов. Примером нелогичной защиты может быть файл, который запрещено читать его автору, но в который разрешено записывать информацию постороннему пользователю;
- ✓ потенциально опасные списки поиска файлов, которые могут привести к запуску «троянского коня»;
- ✓ изменения в системных программах, обнаруженные при помощи контрольных сумм.

Любая проблема, обнаруженная сканером безопасности, может быть как ликвидирована автоматически, так и передана для решения менеджеру системы.

Анализ некоторых ОС с точки зрения их защищенности

Итак, ОС должна способствовать реализации мер безопасности или непосредственно поддерживать их. Примерами подобных решений в рамках аппаратуры и операционной системы могут быть:

- ✓ разделение команд по уровням привилегированности;
- ✓ сегментация адресного пространства процессов и организация защиты сегментов;
- ✓ защита различных процессов от взаимного влияния за счет выделения каждому своего виртуального пространства;
- ✓ особая защита ядра ОС;
- ✓ контроль повторного использования объекта;
- ✓ наличие средств управления доступом;
- ✓ структурированность системы, явное выделение надежной вычислительной базы (совокупности защищенных компонентов), обеспечение компактности этой базы;
- ✓ следование принципу минимизации привилегий - каждому компоненту дается ровно столько привилегий, сколько необходимо для выполнения им своих функций.

Большое значение имеет структура файловой системы. Например, в ОС с дискреционным контролем доступа каждый файл должен храниться вместе с дискреционным списком прав доступа к нему, а при копировании файла все атрибуты, в том числе и ACL, должны быть автоматически скопированы вместе с телом файла.

Меры безопасности не обязательно должны быть заранее встроены в ОС - достаточно принципиальной возможности дополнительной установки защитных продуктов. Так, сугубо ненадежная система MS-DOS может быть усовершенствована за счет средств проверки паролей доступа к компьютеру и/или жесткому диску, за счет борьбы с вирусами путем отслеживания попыток записи в загрузочный сектор CMOS-средствами и т.п. Тем не менее по-настоящему надежная система должна изначально проектироваться с акцентом на механизмы безопасности.

MS-DOS

ОС MS-DOS функционирует в реальном режиме (real-mode) процессора i80x86. В ней невозможно выполнение требования, касающегося изоляции

программных модулей (отсутствует аппаратная защита памяти). Уязвимым местом для защиты является также файловая система FAT, не предполагающая у файлов наличия атрибутов, связанных с разграничением доступа к ним. Таким образом, MS-DOS находится на самом нижнем уровне в иерархии защищенных ОС.

Windows NT/2000/XP

Компоненты защиты NT частично встроены в ядро, а частично реализуются подсистемой защиты. Подсистема защиты контролирует доступ и учетную информацию. Кроме того, Windows NT имеет встроенные средства, такие как поддержка резервных копий данных и управление источниками бесперебойного питания, которые в целом повышают общий уровень безопасности.

В дальнейшем ОС Windows NT/2000/XP, изготовленных по технологии NT, будем называть просто Windows NT.

Ключевая цель системы защиты Windows NT - следить за тем, кто и к каким объектам осуществляет доступ. Система защиты хранит информацию, относящуюся к безопасности для каждого пользователя, группы пользователей и объекта. Единообразие контроля доступа к различным объектам (процессам, файлам, семафорам и др.) обеспечивается тем, что с каждым процессом связан маркер доступа, а с каждым объектом - дескриптор защиты. Маркер доступа в качестве параметра имеет идентификатор пользователя, а дескриптор защиты - списки прав доступа. ОС может контролировать попытки доступа, которые производятся процессами прямо или косвенно инициированными пользователем.

Windows NT отслеживает и контролирует доступ как к объектам, которые пользователь может видеть посредством интерфейса (такие, как файлы и принтеры), так и к объектам, которые пользователь не может видеть (например, процессы и именованные каналы). Помимо разрешающих записей, списки прав доступа содержат и запрещающие записи, чтобы пользователь, которому доступ к какому-либо объекту запрещен, не смог получить его как член какой-либо группы, которой этот доступ предоставлен.

Система защиты ОС Windows NT состоит из следующих компонентов:

1. **Процедуры регистрации (Logon Processes),** которые обрабатывают запросы пользователей на вход в систему. Они включают в себя начальную интерактивную процедуру, отображающую начальный диалог с пользователем на экране и удаленные процедуры входа, которые позволяют удаленным пользователям получить доступ с рабочей станции сети к серверным процессам Windows NT.
2. **Подсистемы локальной авторизации (Local Security Authority, LSA),** которая гарантирует, что пользователь имеет разрешение на доступ в систему. Этот компонент - центральный для системы защиты Windows NT. Он порождает маркеры доступа, управляет локальной политикой безопасности и предоставляет интерактивным пользователям аутентификационные услуги. LSA также контролирует политику аудита и ведет журнал, в котором сохраняются сообщения, порождаемые диспетчером доступа.
3. **Менеджера учета (Security Account Manager, SAM),** который управляет базой данных учета пользователей. Эта база данных содержит информацию обо всех пользователях и группах пользователей. SAM предоставляет услуги по легализации пользователей, применяющиеся в LSA.
4. **Диспетчера доступа (Security Reference Monitor, SRM),** который проверяет, имеет ли пользователь право на доступ к объекту и на выполнение тех действий, которые он пытается совершить. Этот компонент обеспечивает легализацию доступа и политику аудита, определяемые LSA. Он предоставляет услуги для программ супервизорного и пользовательского режимов, для того чтобы гарантировать, что пользователи и процессы, осуществляющие попытки доступа к объекту, имеют необходимые права. Данный компонент также порождает сообщения службы аудита, когда это необходимо.

Контрольные вопросы:

1. Перечислить основные задачи защиты операционных систем
2. Пояснить механизмы идентификации и аутентификации пользователей
3. Пояснить механизмы применения пользовательского пароля
4. Охарактеризовать методы разграничения доступа к объектам ОС
5. Перечислить и охарактеризовать способы управления доступом в ОС
6. Провести анализ некоторых ОС с точки зрения их защищенности

Лекция 22
Системы защиты программного обеспечения

Основные угрозы целостности

Самыми частыми и самыми опасными (с точки зрения **размера ущерба**) являются непреднамеренные ошибки штатных пользователей, операторов, системных администраторов и других лиц, обслуживающих информационные системы.

На втором месте по **размерам ущерба** (после непреднамеренных ошибок и упущений) стоят **кражи и подлоги**. По данным газеты USA Today, еще в 1992 году в результате подобных противоправных действий с использованием персональных компьютеров американским организациям был нанесен общий ущерб в размере 882 миллионов долларов. Можно предположить, что реальный ущерб был намного больше, поскольку многие организации по понятным причинам скрывают такие инциденты; не вызывает сомнений, что в наши дни ущерб от такого рода действий вырос многократно.

В большинстве случаев виновниками оказывались штатные сотрудники организаций, хорошо знакомые с режимом работы и мерами защиты. Это еще раз подтверждает опасность внутренних угроз.

С целью нарушения **статистической целостности информации** злоумышленник (как правило, штатный сотрудник) может:

- ✓ ввести неверные данные;
- ✓ изменить данные.

Иногда изменяются содержательные данные, иногда - служебная информация. Заголовки электронного письма могут быть подделаны; письмо в целом может быть фальсифицировано лицом, знающим пароль отправителя. Отметим, что последнее возможно даже тогда, когда целостность контролируется криптографическими средствами. Здесь имеет место взаимодействие разных аспектов информационной безопасности: если нарушена конфиденциальность, может пострадать целостность.

Угрозой целостности является не только фальсификация или изменение данных, но и отказ от совершенных действий. Если нет средств обеспечить «неотказуемость», компьютерные данные не могут рассматриваться в качестве доказательства.

Потенциально уязвимы с точки зрения нарушения **целостности** не только **данные**, но и **программы**. Угрозами **динамической целостности** являются переупорядочение, кража, дублирование данных или внесение до-

полнительных сообщений (сетевых пакетов и т.п.). Соответствующие действия в сетевой среде называются **активным прослушиванием.**

Основные угрозы конфиденциальности

Конфиденциальную информацию можно разделить на **предметную** и **служебную. Служебная информация** (например, пароли пользователей) не относится к определенной предметной области, в информационной системе она играет техническую роль, но ее раскрытие особенно опасно, поскольку оно чревато получением несанкционированного доступа ко всей информации, в том числе предметной.

Даже если информация хранится в компьютере или предназначена для компьютерного использования, угрозы ее конфиденциальности могут носить некомпьютерный и вообще нетехнический характер.

Многим людям приходится выступать в качестве пользователей не одной, а целого ряда систем (информационных сервисов). Если для доступа к таким системам используются **многоразовые пароли** или иная конфиденциальная информация, то наверняка эти данные будут храниться не только в голове, но и в записной книжке или на листках бумаги, которые пользователь часто оставляет на рабочем столе или теряет. И дело здесь не в неорганизованности людей, а в изначальной непригодности парольной схемы. Невозможно помнить много разных паролей; рекомендации по их регулярной (по возможности - частой) смене только усугубляют положение, заставляя применять несложные схемы чередования или вообще стараться свести дело к двум-трем легко запоминаемым (и столь же легко угадываемым) паролям.

Описанный класс уязвимых мест можно назвать размещением конфиденциальных данных в среде, где им не обеспечена (и часто не может быть обеспечена) необходимая защита. Помимо паролей, хранящихся в записных книжках пользователей, в этот класс попадает **передача конфиденциальных данных в открытом виде** (в разговоре, в письме, по сети), которая делает возможным их перехват. Для атаки могут использоваться разные технические средства (подслушивание или прослушивание разговоров, **пассивное прослушивание сети** и т. п.), но идея одна - осуществить доступ к данным в тот момент, когда они наименее защищены.

Угрозу перехвата данных следует принимать во внимание не только при начальном конфигурировании ИС, но и, что очень важно, при всех изменениях. Весьма опасной угрозой являются выставки, на которые многие организации отправляют оборудование из производственной сети со всеми хранящимися на них данными. Остаются прежними пароли, при удаленном доступе они продолжают передаваться в открытом виде.

Еще один пример изменения: хранение данных на резервных носителях. Для защиты данных на основных носителях применяются развитые системы управления доступом; копии же нередко просто лежат в шкафах, и получить доступ к ним могут многие.

Перехват данных - серьезная угроза, и если конфиденциальность действительно является критичной, а данные передаются по многим каналам, их защита может оказаться весьма сложной и дорогостоящей. Технические средства перехвата хорошо проработаны, доступны, просты в эксплуатации, а установить их, например, на кабельную сеть, может кто угодно, так что эта угроза существует не только для внешних, но и для внутренних коммуникаций.

Кражи оборудования являются угрозой не только для резервных носителей, но и для компьютеров, особенно портативных. Часто ноутбуки оставляют без присмотра на работе или в автомобиле, иногда просто теряют.

Опасной нетехнической угрозой конфиденциальности являются методы морально-психологического воздействия, такие как **маскарад** - выполнение действий под видом лица, обладающего полномочиями для доступа к данным.

К неприятным угрозам, от которых трудно защищаться, можно отнести **злоупотребление полномочиями**. На многих типах систем привилегированный пользователь (например, системный администратор) способен прочитать любой (незашифрованный) файл, получить доступ к почте любого пользователя и т. д. Другой пример - нанесение ущерба при сервисном обслуживании. Обычно сервисный инженер получает неограниченный доступ к оборудованию и имеет возможность действовать в обход программных защитных механизмов.

Методы защиты

Существующие **методы и средства защиты информации** компьютерных систем (КС) можно подразделить на четыре основные группы:

- ✓ методы и средства организационно-правовой защиты информации;
- ✓ методы и средства инженерно-технической защиты информации;
- ✓ криптографические методы и средства защиты информации;
- ✓ программно-аппаратные методы и средства защиты информации.

Методы и средства организационно-правовой защиты информации

К методам и средствам организационной защиты информации относятся организационно-технические и организационно-правовые мероприятия,

проводимые в процессе создания и эксплуатации КС для обеспечения защиты информации. Эти мероприятия должны проводиться при строительстве или ремонте помещений, в которых будет размещаться КС; проектировании системы, монтаже и наладке ее технических и программных средств; испытаниях и проверке работоспособности КС.

На этом уровне защиты информации рассматриваются международные договоры, подзаконные акты государства, государственные стандарты и локальные нормативные акты конкретной организации.

Методы и средства инженерно-технической защиты

Под инженерно-техническими средствами защиты информации понимают физические объекты, механические, электрические и электронные устройства, элементы конструкции зданий, средства пожаротушения и другие средства, обеспечивающие:

- ✓ защиту территории и помещений КС от проникновения нарушителей;
- ✓ защиту аппаратных средств КС и носителей информации от хищения;
- ✓ предотвращение возможности удаленного (из-за пределов охраняемой территории) видеонаблюдения (подслушивания) за работой персонала и функционированием технических средств КС;
- ✓ предотвращение возможности перехвата ПЭМИН (побочных электромагнитных излучений и наводок), вызванных работающими техническими средствами КС и линиями передачи данных;
- ✓ организацию доступа в помещения КС сотрудников;
- ✓ контроль над режимом работы персонала КС;
- ✓ контроль над перемещением сотрудников КС в различных производственных зонах;
- ✓ противопожарную защиту помещений КС;
- ✓ минимизацию материального ущерба от потерь информации, возникших в результате стихийных бедствий и техногенных аварий.

Важнейшей составной частью инженерно-технических средств защиты информации являются **технические средства охраны**, которые образуют первый рубеж защиты КС и являются необходимым, но недостаточным условием сохранения конфиденциальности и целостности информации в КС.

Криптографические методы защиты и шифрование

Шифрование является основным средством обеспечения конфиденциальности. Так, в случае обеспечения конфиденциальности данных на локальном компьютере применяют шифрование этих данных, а в случае сетевого взаимодействия - шифрованные каналы передачи данных.

Науку о защите информации с помощью шифрования называют **криптографией** (криптография в переводе означает загадочное письмо или тайнопись).

Криптография применяется:

- ✓ для защиты конфиденциальности информации, передаваемой по открытым каналам связи;
- ✓ для аутентификации (подтверждении подлинности) передаваемой информации;
- ✓ для защиты конфиденциальной информации при ее хранении на открытых носителях;
- ✓ для обеспечения целостности информации (защите информации от внесения несанкционированных изменений) при ее передаче по открытым каналам связи или хранении на открытых носителях;
- ✓ для обеспечения неоспоримости передаваемой по сети информации (предотвращения возможного отрицания факта отправки сообщения);
- ✓ для защиты программного обеспечения и других информационных ресурсов от несанкционированного использования и копирования.

Программные и программно-аппаратные методы и средства обеспечения информационной безопасности

К аппаратным средствам защиты информации относятся электронные и электронно-механические устройства, включаемые в состав технических средств КС и выполняющие (самостоятельно или в едином комплексе с программными средствами) некоторые функции обеспечения информационной безопасности. Критерием отнесения устройства к аппаратным, а не к инженерно-техническим средствам защиты является обязательное включение в состав технических средств КС.

К основным аппаратным средствам защиты информации относятся:

- ✓ устройства для ввода идентифицирующей пользователя информации (магнитных и пластиковых карт, отпечатков пальцев и т. п.);
- ✓ устройства для шифрования информации;
- ✓ устройства для воспрепятствования несанкционированному включению рабочих станций и серверов (электронные замки и блокираторы).

Примеры вспомогательных аппаратных средств защиты информации:

- ✓ устройства уничтожения информации на магнитных носителях;
- ✓ устройства сигнализации о попытках несанкционированных действий пользователей КС и др.

Под программными средствами защиты информации понимают специальные программы, включаемые в состав программного обеспечения КС исключительно для выполнения защитных функций.

К основным программным средствам защиты информации относятся:

- ✓ программы идентификации и аутентификации пользователей КС;
- ✓ программы разграничения доступа пользователей к ресурсам КС;
- ✓ программы шифрования информации;
- ✓ программы защиты информационных ресурсов (системного и прикладного программного обеспечения, баз данных, компьютерных средств обучения и т. п.) от несанкционированного изменения, использования и копирования.

Заметим, что под идентификацией, применительно к обеспечению информационной безопасности КС, понимают однозначное распознавание уникального имени субъекта КС. Аутентификация означает подтверждение того, что предъявленное имя соответствует данному субъекту (подтверждение подлинности субъекта).

Примеры **вспомогательных программных средств** защиты информации:

- ✓ программы уничтожения остаточной информации (в блоках оперативной памяти, временных файлах и т. п.);
- ✓ программы аудита (ведения регистрационных журналов) событий, связанных с безопасностью КС, для обеспечения возможности восстановления и доказательства факта происшествия этих событий;

- ✓ программы имитации работы с нарушителем (отвлечения его на получение якобы конфиденциальной информации);
- ✓ программы тестового контроля защищенности КС и др.

Контрольные вопросы:

1. Перечислить и охарактеризовать основные угрозы целостности
2. Перечислить и охарактеризовать основные угрозы конфиденциальности
3. Перечислить группы методов и средств защиты информации
4. Перечислить и охарактеризовать методы и средства организационно-правовой защиты информации
5. Перечислить и охарактеризовать методы и средства инженерно-технической защиты информации
6. Перечислить и охарактеризовать криптографические методы защиты информации
7. Перечислить основные аппаратные средства защиты информации
8. Перечислить основные программные средства защиты информации

Лекция 23
Администрирование сети

Понятие администрирования. Задачи администратора сети

Основной целью администрирования является приведение сети в соответствие с целями и задачами, для которых она предназначена. Администрирование заключается в контроле за работой сетевого оборудования и управление функционированием сети в целом. Администрирование выполняет администратор сети – специалист, отвечающий за нормальное функционирование и использование ресурсов сети. Если более детально, то **администрирование информационных систем включает следующие цели:**

- ✓ Установка и настройка сети. Поддержка её дальнейшей работоспособности.
- ✓ Мониторинг. Планирование системы.
- ✓ Установка и конфигурация аппаратных устройств.
- ✓ Установка программного обеспечения.
- ✓ Архивирование (резервное копирование) информации.
- ✓ Создание и управление пользователями.
- ✓ Установка и контроль защиты.

Вот выписка должностных обязанностей администратора сети:

- ✓ Устанавливает на серверы и рабочие станции сетевое программное обеспечение.
- ✓ Конфигурирует систему на сервере.
- ✓ Обеспечивает интегрирование программного обеспечения на файл-серверах, серверах систем управления базами данных и на рабочих станциях.
- ✓ Поддерживает рабочее состояние программного обеспечения сервера.
- ✓ Регистрирует пользователей, назначает идентификаторы и пароли.
- ✓ Обучает пользователей работе в сети, ведению архивов; отвечает на вопросы пользователей, связанные с работой в сети; составляет инструкции по работе с сетевым программным обеспечением и доводит их до сведения пользователей.
- ✓ Контролирует использование сетевых ресурсов.
- ✓ Организует доступ к локальной и глобальной сетям.
- ✓ Устанавливает ограничения для пользователей по:

- использованию рабочей станции или сервера;
- времени;
- степени использования ресурсов.

✓ Обеспечивает своевременное копирование и резервирование данных.

✓ Обращается к техническому персоналу при выявлении неисправностей сетевого оборудования.

✓ Участвует в восстановлении работоспособности системы при сбоях и выходе из строя сетевого оборудования.

✓ Выявляет ошибки пользователей и сетевого программного обеспечения и восстанавливает работоспособность системы.

✓ Проводит мониторинг сети, разрабатывает предложения по развитию инфраструктуры сети.

✓ Обеспечивает:

- сетевую безопасность (защиту от несанкционированного доступа к информации, просмотра или изменения системных файлов и данных);
- безопасность межсетевого взаимодействия.

✓ Готовит предложения по модернизации и приобретению сетевого оборудования.

✓ Осуществляет контроль за монтажом оборудования специалистами сторонних организаций.

✓ Сообщает своему непосредственному руководителю о случаях злоупотребления сетью и принятых мерах.

✓ Ведет журнал системной информации, иную техническую документацию.

Группы пользователей - что это и зачем?

Все пользователи сети разделяются по своим полномочиям на группы. Каждая группа может отвечать за выполнение тех или иных задач. Возможно такое определение прав групп пользователей, при котором пользователи имеют все права, необходимые им для выполнения своих функций, но не более того. **Полностью все права должны быть только у одного пользователя - администратора (супервизора) сети.** Он имеет все права, в том числе может создавать группы пользователей и определять права, которыми они обладают.

Пользователи могут быть членами одновременно нескольких групп. Можно, например, создать новый каталог и разрешить к нему доступ сразу для всех пользователей сети. При этом вам придется изменять права доступа не для всех пользователей (их может быть несколько десятков), а только для одной группы, что существенно легче. Для каждой лаборатории или отдела имеет смысл создать свою группу пользователей. Если у вас есть пользователи, которым требуются дополнительные права (например, права доступа к каким-либо каталогам или сетевым принтерам), создайте соответствующие группы пользователей и предоставьте им эти права.

Если в сети много рабочих станций, которые находятся в разных помещениях и принадлежат разным отделам или лабораториям, имеет смысл создать группу администраторов сети. Права нескольких администраторов сети определяются системным администратором. Не следует предоставлять администраторам сети всех прав системного администратора. Вполне достаточно, если в каждом отделе или лаборатории будет Один-два администратора, которые обладают правами управления, только работающими в данном отделе или лаборатории пользователями. Если в отделе или лаборатории имеется сетевой принтер или какие-либо другие сетевые ресурсы, у администратора должны быть права на управление этими устройствами. Однако вовсе ни к чему, чтобы администратор одной лаборатории мог управлять сетевым принтером, принадлежащим другой лаборатории. При этом пользователи должны иметь минимально необходимые им для нормальной работы права доступа к дискам сервера.

Таким образом, очевидно, что создание групп пользователей актуально только в больших компьютерных сетях. Если сеть небольшая, то и один человек справится с такими вопросами, как добавление новых пользователей, разграничение доступа к дискам сервера, сетевым принтерам и другим сетевым ресурсам и в создании групп администраторов и обычных пользователей смысла нет.

Создание группы пользователей

Запускаем на виртуальной машине сервер. Наберем команду **mmc** и добавим в консоль оснастки, с которыми будем работать - **DNS, DHCP, AD-пользователи и компьютеры.** Для этого потребуется команда **Консоль - Добавить или удалить оснастку - Добавить** (рис. 39)

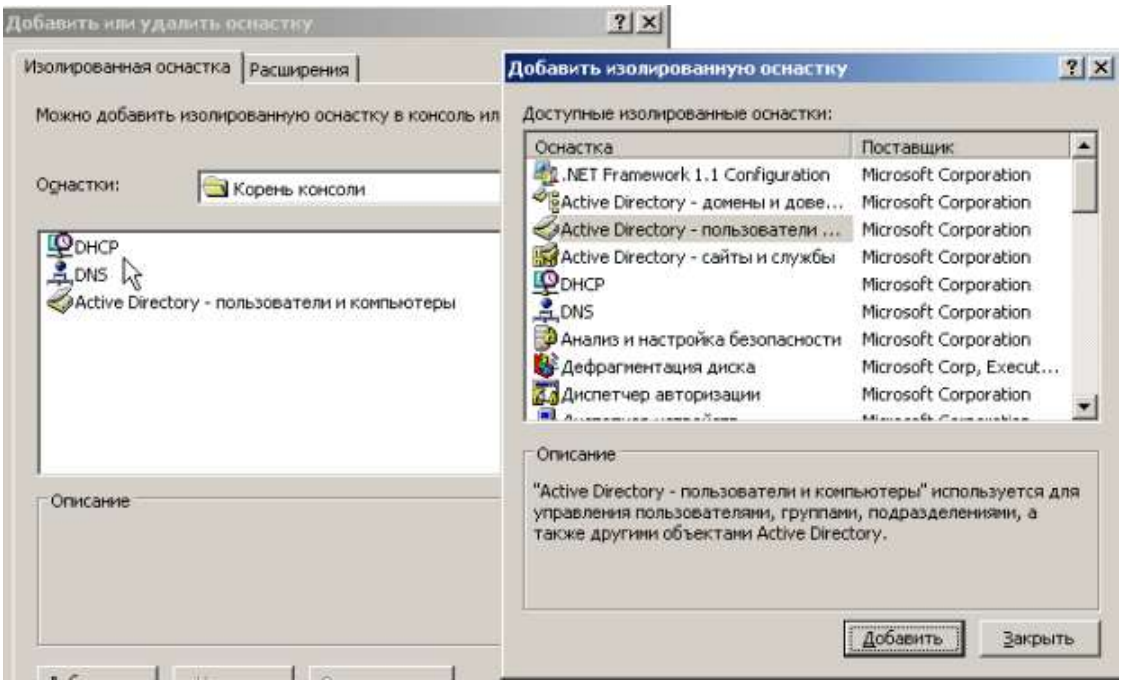

Рис. 39. Создаем консоль с нужными оснастками

Теперь в **AD** щелкните правой кнопкой мыши и выполните команду **Создать - Группа** (рис. 40 и рис. 41).

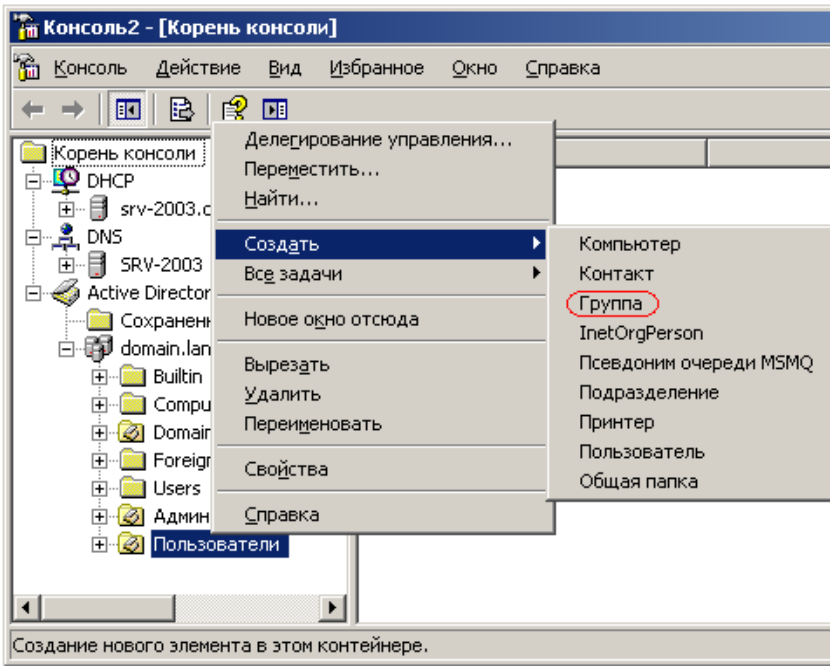

Рис. 40. Создаем группу пользователей

Группа безопасности назначает права доступа к ресурсам сети (администрирует). **Группа распространения** не может заниматься администрированием, она занимается рассылкой сообщений. **Локальная в домене** может содержать в себе пользователя любого домена в лесу, но администрировать эта группа может только в том домене, в котором группа создавалась. **Глобальная** может содержать в себе пользователей из того домена, в котором

она была создана, но администрировать они могут любой домен в лесу (если эти домены доверяют друг другу). **Универсальная** может содержать пользователей из любого домена леса, и эта группа может администрировать любой домен в лесу.

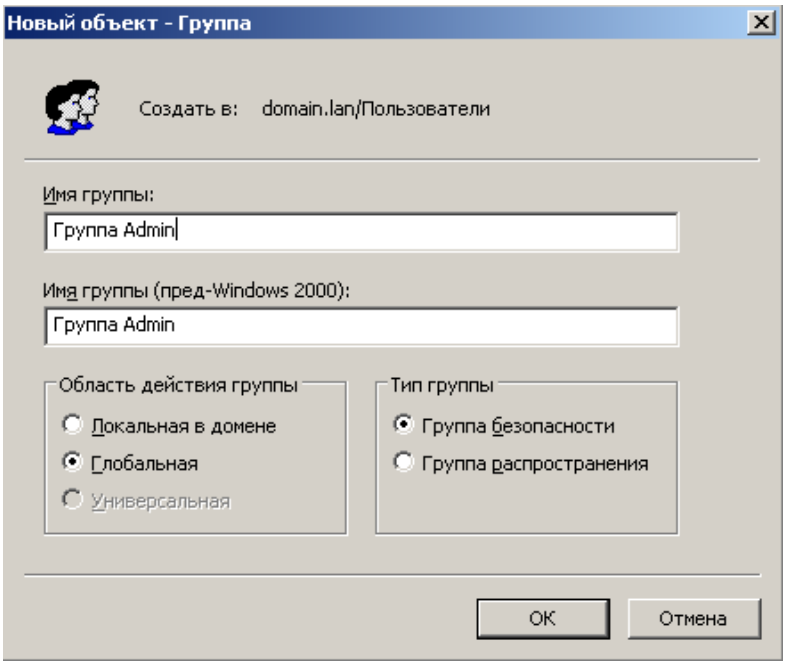

Рис. 41. В этом окне оставим настройки по умолчанию

Группа **Admin** создана (рис. 42). Теперь нужно добавить в нее пользователей.

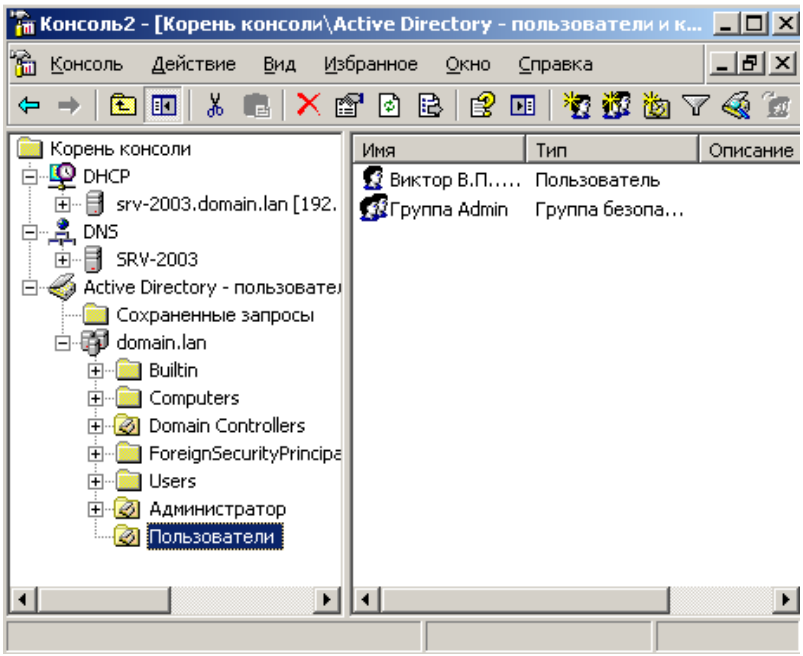

Рис. 42. Группа Admin создана

Щелкнем дважды мышкой на этой группе и на вкладке **Члены группы** нажмем на кнопку **Добавить** (рис. 43).

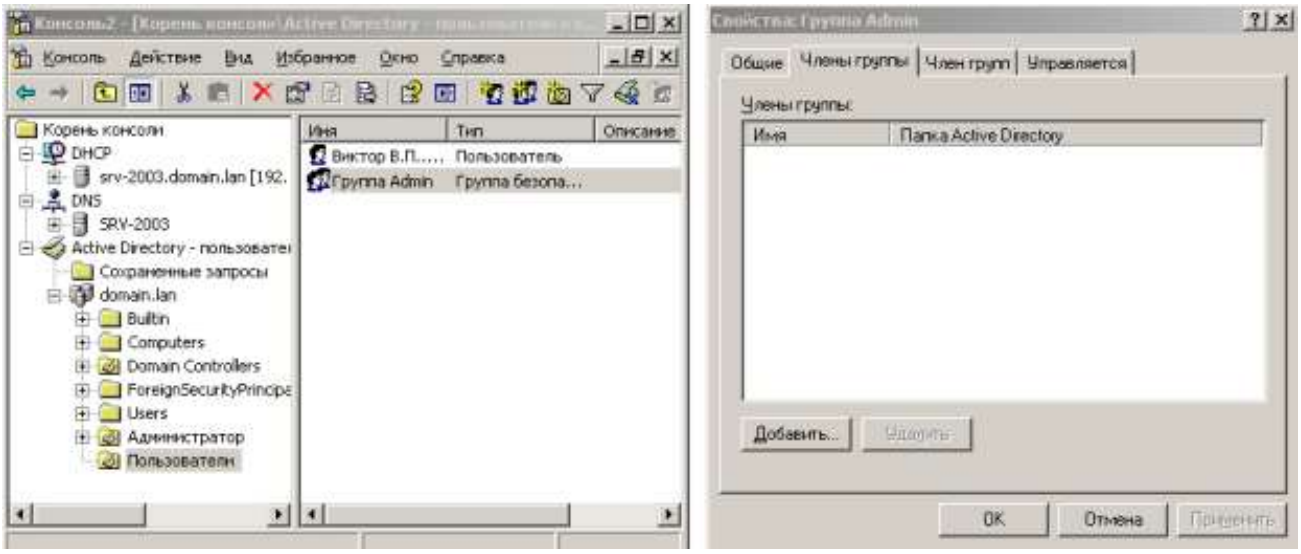

Рис. 43. Активна вкладка Члены группы

Теперь нажмите на кнопку **Дополнительно** и **Поиск** (рис. 44).

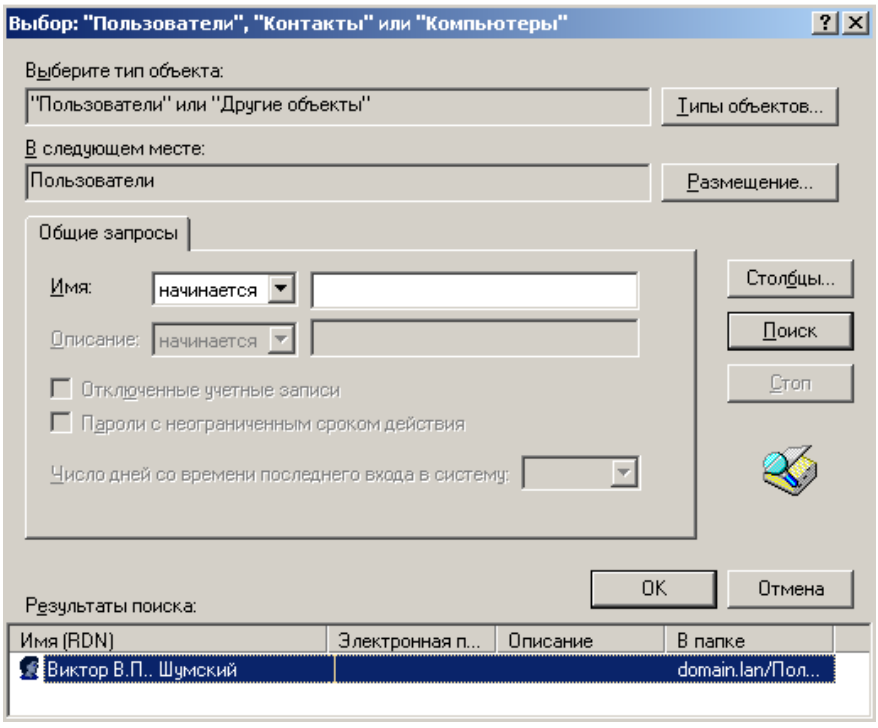

Рис. 44. Виктора Шумского мы добавим в группу Admin кнопкой ОК

Теперь если мы щелкнем мышкой на Виктора Шумского в консоли, то мы увидим в его свойствах, что он является членом сразу двух групп (рис. 45)

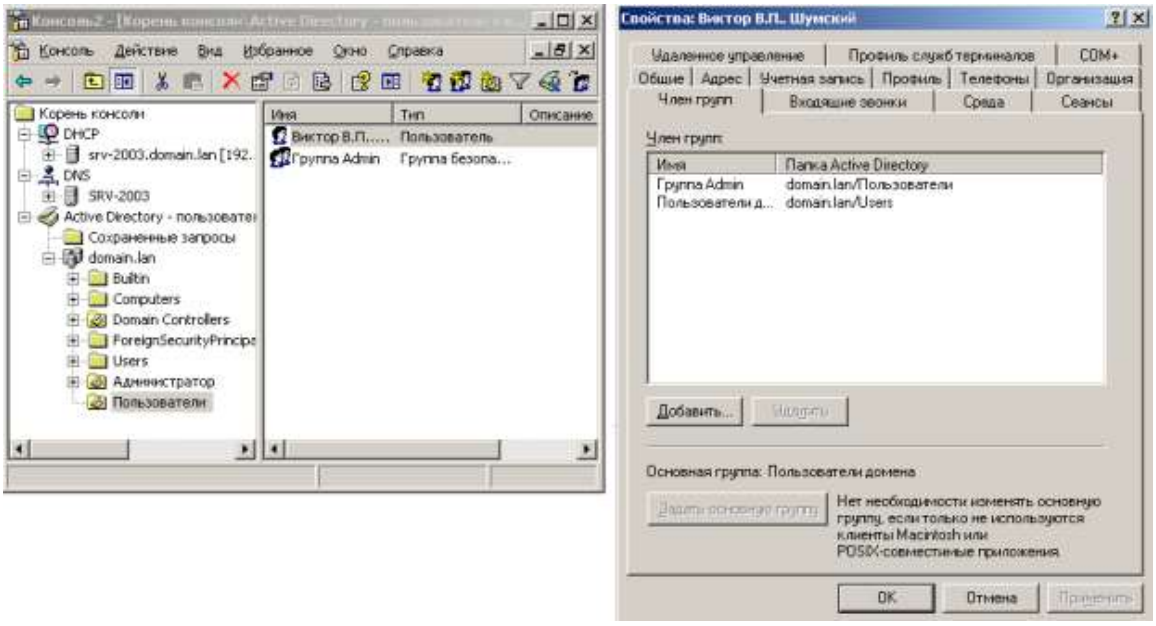

Рис. 45. Виктор - член группы пользователей домена и группы Admin

Созданную нами консоль командой **Консоль - Сохранить как** сохраним на Рабочий стол.

Администрирование групп

Рассмотрим, как какому-то пользователю можно назначить те или иные права на ресурсы сети. Создадим пользователя **Кондратьева Мария (kondrateva_mv),** который будет управлять группой **Admin**, то есть, назначать им права на ресурсы. В консоли щелкаем правой кнопкой мыши и выполняем команду **Создать - Пользователь** (рис. 46).

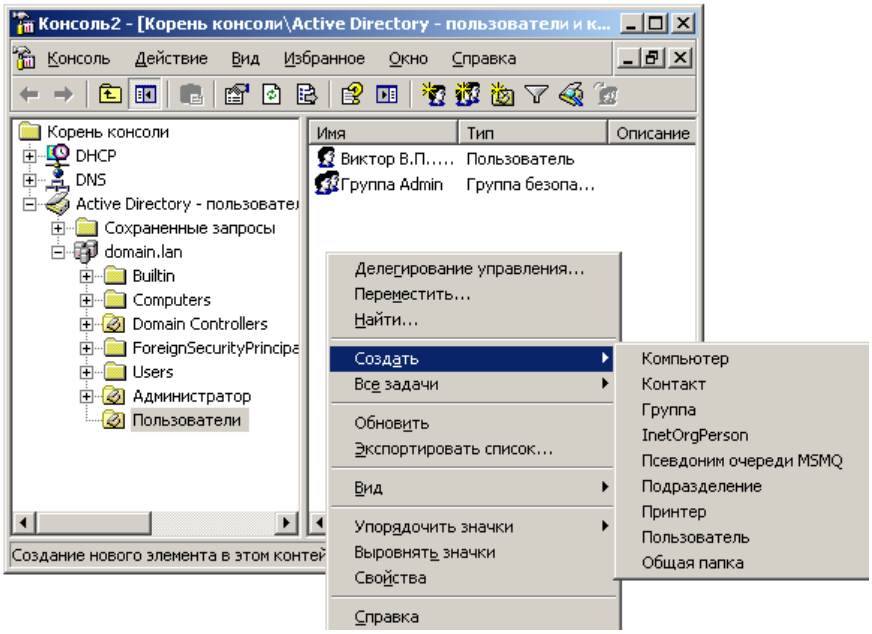

Рис. 46. Создаем нового пользователя

Заполняем форму (рис. 47).

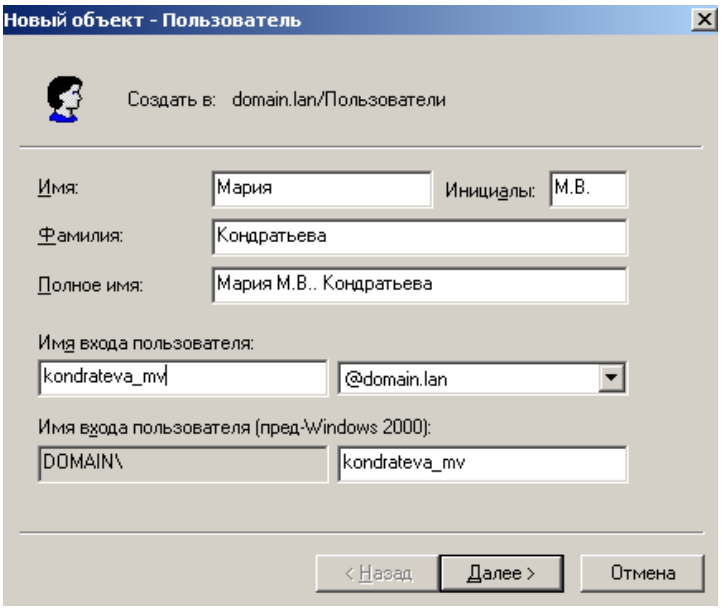

Рис. 47. Вводим данные пользователя

В консоли выполняем команду **Вид - Дополнительные функции**. Далее в консоли заходим в группу **Admin** и активируем вкладку **Безопасность** (рис. 48).

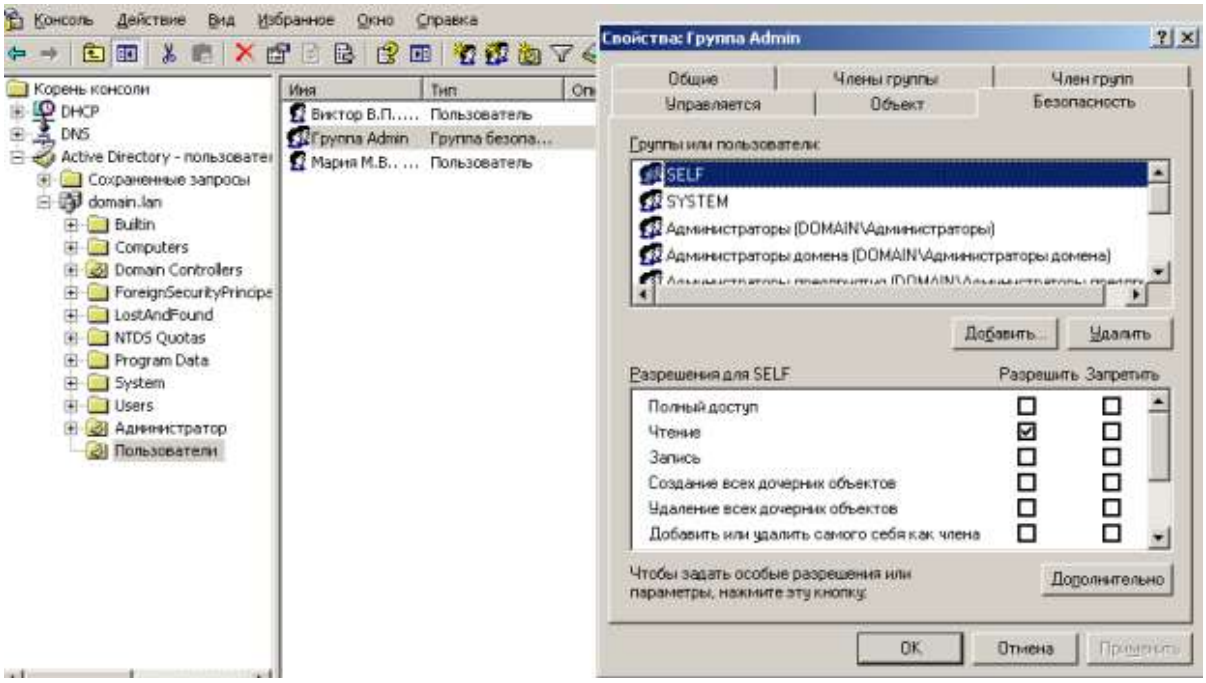

Рис. 48. Окно Свойства группа Admin, вкладка Безопасность

Примечание

Ресурсы хранятся в Active Directory как объекты. Любой объект, с которым связан другой объект, является родительским. В окне на рисунке выше мы видим - дочерний объект. Дочерние объекты в свою очередь могут иметь детей. Хотя родитель может иметь любое количество детей, но ребенок может иметь только одного родителя. Active Directory автоматически выстраивает двусторонние доверительные отношения между родительскими и дочерними доменами в дереве доменов. При создании дочернего домена доверительные отношения автоматически формируются между дочерним доменом и его родителем, что существенно упрощает управление доменами.

Нажимаем на кнопку **Дополнительно - Добавить…Поиск** и добавляем нашего пользователя кнопкой ОК (рис. 49).

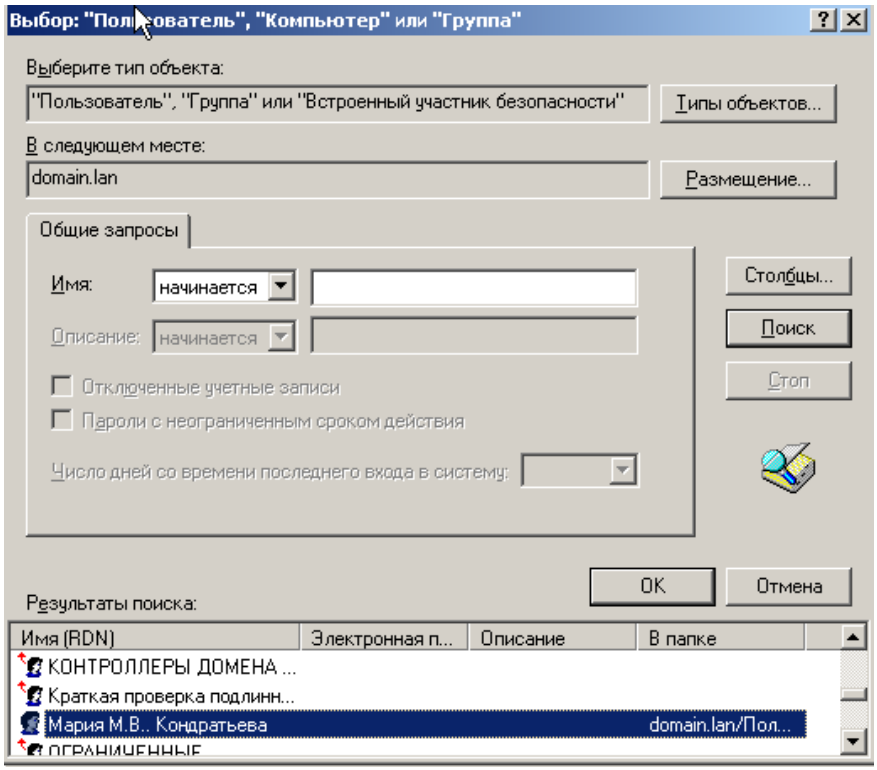

Рис. 49. Выбираем пользователя и нажимаем на кнопку ОК

Теперь зададим, что мы можем разрешить, а что запретить для пользователя (рис. 50).

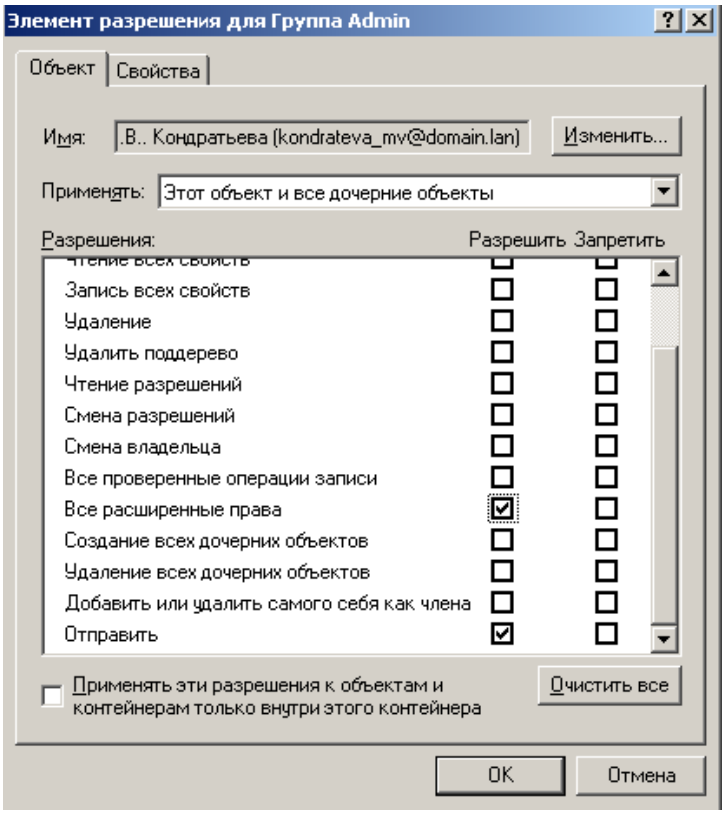

Рис. 50. Определяем права пользователя

Групповая политика

Оснастка **Групповая политика** используется для определения параметров политики, которые будут применяться к компьютерам или пользователям.

Групповая политика — это набор правил или настроек, в соответствии с которыми производится настройка рабочей среды Windows. Продуманное применение объектов групповой политики к объектам каталога Active Directory позволяет создавать эффективную и легко управляемую компьютерную рабочую среду на базе ОС Windows.

Изменение групповой политики всех пользователей и ПК

Чтобы увидеть групповую политику домена по умолчанию, зайдем в консоль и, щелкнув в AD на названии домена, вызываем окно свойств домена (рис. 51). Эта политика применяется ко всем пользователям и ПК в домене.

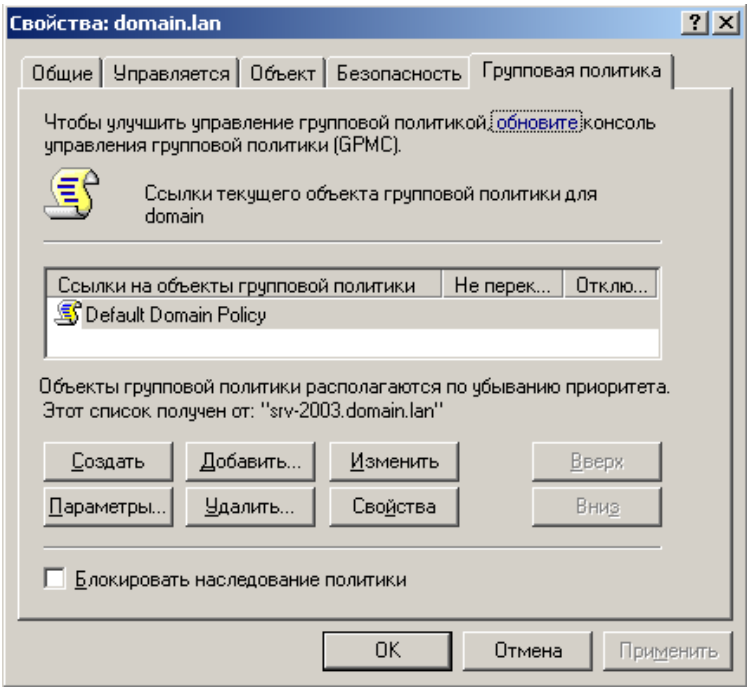

Рис. 51. В окне открыта вкладка Групповая политика

Жмем на кнопку **Изменить** (рис. 52).

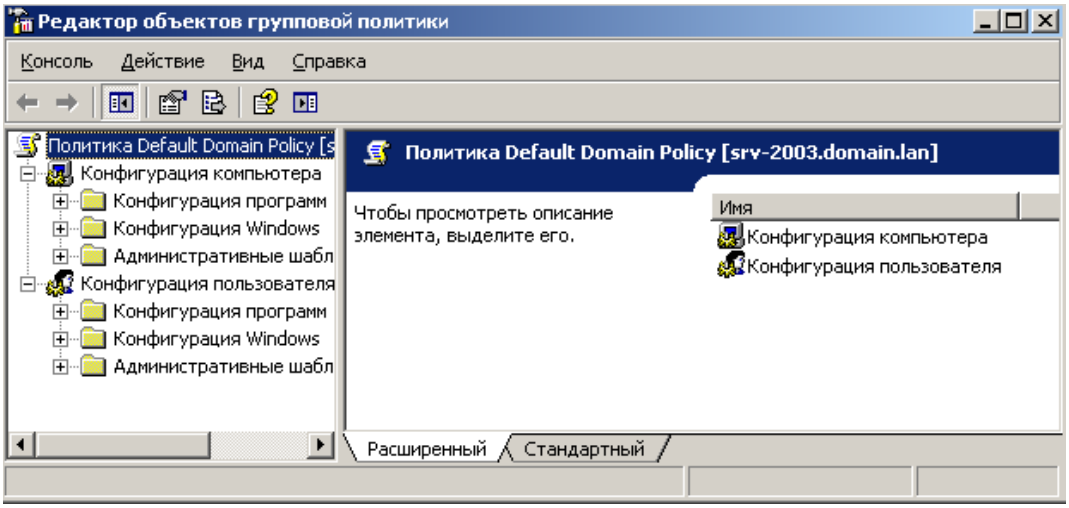

Рис. 52. На рисунке справа мы видим политику компьютера и пользователя

Пусть, например, для всех пользователей домена мы уберем меню **Справка и поддержка**, а также - **Моя музыка** из меню **Пуск** (рис. 53).

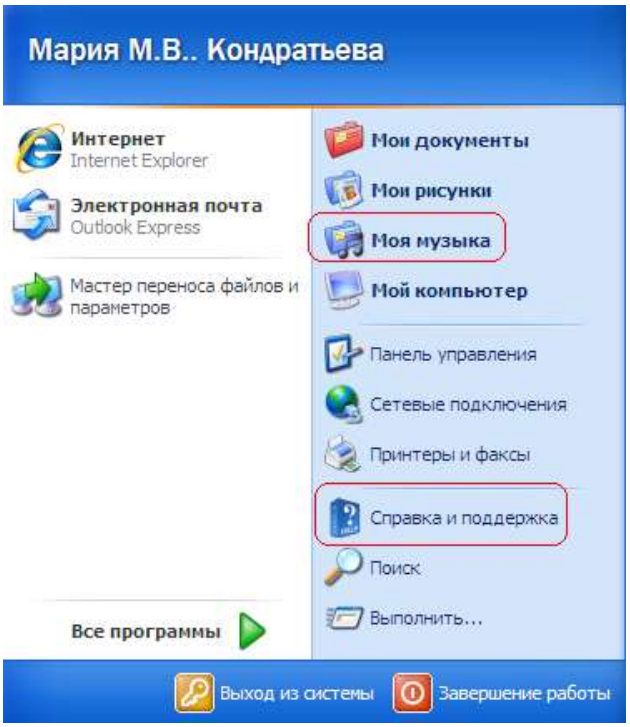

Рис. 53. Отмеченные пункты изначально в меню Пуск присутствуют

Зайдем в **Административные шаблоны** и найдем пункт **Панель задач и меню Пуск**. Находим запись **Удалить значок Моя музыка из главного меню** и задаем переключатель **Включен** (рис. 54). Затем находим запись **Удалить справку из главного меню** и снова устанавливаем переключатель **Включен**.

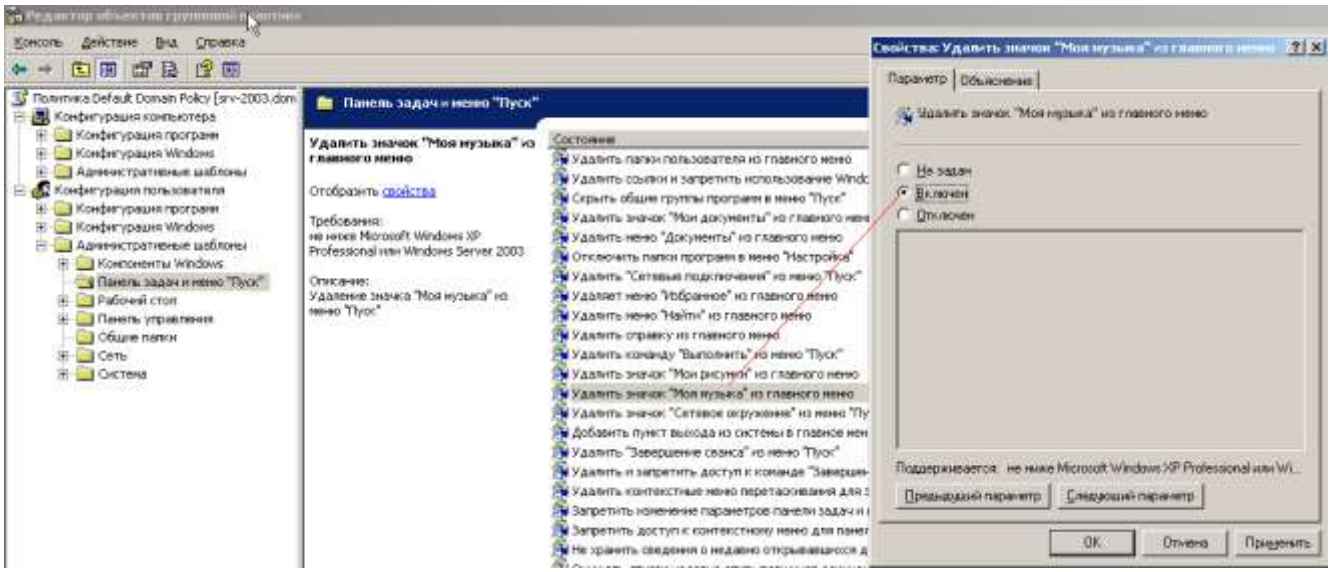

Рис. 54. Меняем два параметра групповой политики по умолчанию

После **ОК** входим на клиента заново (рис. 55).

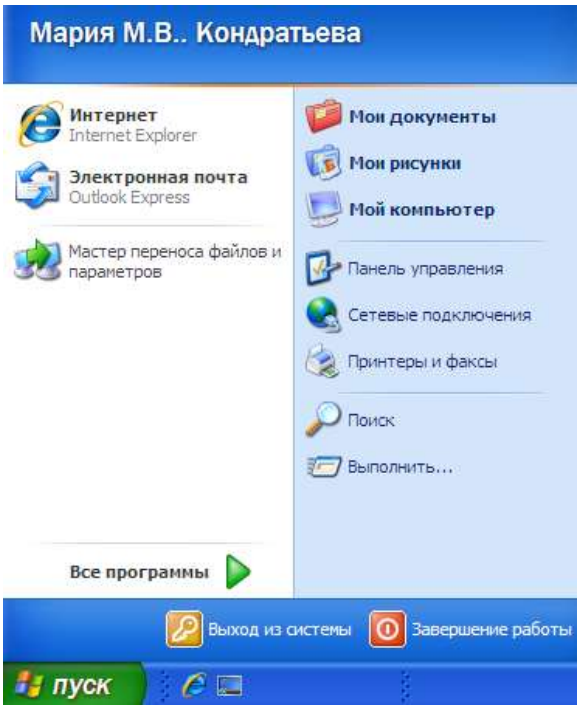

Рис. 55. Меню Пуск изменилось

Таким образом, редактируя групповую политику на сервере, мы можем сразу менять множество параметров на клиентах, что облегчает работу администратора сети.

Контрольные вопросы:

1. Пояснить, в чем заключается процесс администрирования сети
2. Перечислить функции администратора сети
3. Пояснить, что такое группы пользователей
4. Пояснить процесс администрирования на рассмотренном примере

Используемая литература

Основная

№ п/п	Наименование	Автор	Издательство и год издания
1	Операционные системы, среды и оболочки	Партыка Т.Л., Попов И.И.	Учебное пособие - М.: ФОРУМ: ИНФРА-М, 2004.
2	Операционные системы. Концепции построения и обеспечения безопасности	Мартемьянов Ю.Ф., Яковлев Ал.В., Яковлев Ан.В	Учебное пособие для вузов. – М.: Горячая линия-Телеком, 2011

Дополнительная

№ п/п	Наименование	Автор	Издательство и год издания
1	Информационная безопасность.	Партыка Т.Л., Попов И.И.	Учебное пособие для студентов учреждений среднего профессионального образования. – 2-е изд., испр. и доп.. М.: ФОРУМ: ИНФРА-М, 2007.
2	Информационная безопасность.	Мельников В.П., Клейменов С.А., Петраков А.М.	Учеб. пособие для студ. учреждений сред. проф. образования. – М.: Издательский центр «Академия», 2010
3	Программное обеспечение	Голицина О. Л., Партыка Т. Л., Попов И. И.	Уч. пособие – М.: ФОРУМ: ИНФРА-М, 2006
4	Программное обеспечение и операционные системы ПК	Губарев В.Г.	Ростов на Дону: Феникс, 2002
4	Информатика	Симонович С.В. и др	Базовый курс – СПб: Питер, 2002

Интернет-ресурсы

1. http://www.teachvideo.ru
2. http://inf.1september.ru
3. http://ru.wikipedia.org
4. http://www.teachpro.ru
5. http://www.intuit.ru

Для заметок

Для заметок

Для заметок