Томский межвузовский центр дистанционного образования

Ю.Б. Гриценко

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Часть 1

Учебное пособие

ТОМСК — 2009

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

Ю.Б. Гриценко

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Часть 1

Учебное пособие

2009

Рецензенты: канд. техн. наук, доцент кафедры теоретических основ информатики Томского государственного университета **Гусев И.С.**;

д-р техн. наук, профессор, заведующий кафедры теоретических основ информатики Томского государственного университета **Костюк Ю.Л.**

Корректор: Осипова Е.А.

Гриценко Ю.Б.
Операционные системы: Учебное пособие. В 2-х частях. — Томск: Томский межвузовский центр дистанционного образования, 2009. — Ч.1. — 187 с.

Данное учебное пособие содержит первую часть курса «Операционные системы», изучаемого студентами специальности 230102 «Автоматизированные системы обработки информации и управление», обучающимися по дистанционной форме. Рассмотрены вопросы организации и построения операционных систем. Основное внимание уделено понятиям интерфейсов прикладного программирования, обзору реальных операционных систем: ОС Windows (Microsoft), ОС OS/2 (IBM), eComStation (Serenity Systems& Mensys BV), QNX (QNX Software Systems Limited), Unix и их оболочек. В пособии также изложены правила и основы работы с интерфейсами командных строк операционных систем Windows и Unix.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современное общество живет в век информации. Умение качественно управлять информационными ресурсами — одно из важнейших направлений деятельности человека. В настоящий момент идет бурное развитие автоматизированных систем управления. Развивается как аппаратное, так и ***программное обеспечение*** (ПО).

Программное обеспечение — неотъемлемая составляющая любой ЭВМ, без которой невозможно получить необходимые результаты всевозможных вычислительных операций. При всем многообразии и сложности современных программных систем при их разработке в качестве базовой основы используются уже существующие фундаментальные концепции, имеющие много общего в части принципов построения и отличающиеся некоторыми особенностями реализации.

В работах специалистов по рассматриваемой тематике предлагается множество неоднозначных классификаций программного обеспечения. Например, одна из классификаций предлагает все программы, созданные для ЭВМ, разделить на следующие основные классы:

- операционные системы и сервисные программы;
- инструментальные языки и системы программирования;
- прикладные системы.

В работах других авторов, например Дж. Бэкона и Т. Харриса, предлагается выделить класс ***системного программного обеспечения*** из всего множества ПО (другой класс — прикладное программное обеспечение), к которому и будут относиться операционные системы. В некоторых случаях программное обеспечение, в особенности системное, не может рассматриваться отдельно от аппаратного обеспечения, которое поддерживает его работу и в большей мере определяет его структуру.

Изучение дисциплины «Операционные системы» представляет собой основу для изучения всего процесса управления информационными ресурсами.

В содержание дисциплины входит изучение как теоретического материала: структур, методов и алгоритмов построения современных операционных сред и систем (ОС), так и изучение

возможностей функционирования современных популярных ОС. Базовыми категориями в освоении данного курса являются основные понятия и концепции: построения ОС (операционная среда, вычислительный процесс, ресурс, поток, прерывание), управления задачами (функции, стратегии планирования, дисциплины и алгоритмы диспетчеризации), управления внутренней и внешней памятью, организация архитектуры ОС и интерфейсов прикладного программирования.

Развитие принципов построения ОС тесно связано с развитием средств вычислительной техники. Современная архитектура IBM PC-совместимого компьютера представляет собой реализацию так называемой фон-неймановской архитектуры вычислительных машин. Эта архитектура была представлена Джоном фон Нейманом в 1945 году. Фон-неймановская архитектура — не единственный вариант построения ЭВМ, имеются и другие, которые не соответствуют указанным принципам (например, потоковые машины). Однако подавляющее большинство современных компьютеров основано именно на указанных принципах, включая и сложные многопроцессорные комплексы, которые можно рассматривать как объединение фон-неймановских машин. Теория фон Неймана явилась основой для построения первых ОС. Значительная часть теорий построения ОС была разработана в 70—80-х годах прошлого века. В настоящее время наблюдается возросший интерес со стороны ученых и коммерческих фирм к развитию теорий построения ОС.

Для изучения дисциплины «Операционные системы» необходимо иметь навыки программирования на языке высокого уровня Си или Паскаль.

Структура учебного пособия

Учебное пособие состоит из двух частей. Первая часть ориентирована на рассмотрение функциональных и архитектурных особенностей популярных операционных систем. Вторая же часть посвящена общей теории построения операционных систем.

Часть 1.

В первой главе рассмотрены основные понятия операционных среды и системы, оболочки. Приведена классификация и

архитектура операционных систем и интерфейсов прикладного программирования. Рассмотрены стандарты на интерфейсы прикладного программирования. Сформулированы требования к современным операционным системам.

Вторая глава содержит обзор функциональных возможностей популярных операционных систем, таких, как семейства Windows, Unix, OS/2, а также системы реального времени QNX.

В третьей главе изложены правила и основы работы с интерфейсами командных строк операционных систем Windows и Unix.

В конце учебного пособия приведены задания для выполнения контрольных работ по первой части данного курса.

1 ВВЕДЕНИЕ В ОПЕРАЦИОННЫЕ СРЕДЫ, СИСТЕМЫ И ОБОЛОЧКИ

1.1 Основные понятия

Под ***операционной системой*** обычно понимают комплекс управляющих и обрабатывающих программ, который, с одной стороны, выступает как интерфейс между аппаратной частью компьютера и пользователем с его задачами, а с другой — предназначен для наиболее эффективного использования ресурсов вычислительной системы и организации надежных вычислений [1]. Любой из компонентов прикладного ПО обязательно работает под управлением ОС.

Основные функции ОС состоят в следующем [1]:

– прием от пользователя или от оператора системы заданий или команд, сформулированных на соответствующем языке в виде команд оператора или указаний с помощью соответствующего манипулятора (например, с помощью мыши), и их обработка;

– прием и исполнение программных запросов на запуск, приостановку, остановку других программ;

– загрузка в оперативную память подлежащих исполнению программ;

– инициация программы — передача данной конкретной программе управления, в результате чего процессор приступает к ее выполнению;

– идентификация всех программ и данных;

– обеспечение режима мультипрограммирования, то есть выполнение двух или более программ на одном процессоре, создающее видимость их одновременного исполнения;

– организация и управление всеми операциями ввода/вывода;

– исполнение режима жестких ограничений на время ответа в режиме реального времени (характерно для ОС реального времени);

– распределение памяти и организация виртуальной памяти;

– планирование и диспетчеризация задач в соответствии с заданной стратегией и дисциплиной обслуживания;

– организация механизмов обмена сообщениями и данными между выполняющимися программами;

– защита одной программы от влияния другой и обеспечение сохранности данных;

– предоставление услуг в случае частичного сбоя системы;

– обеспечение работы систем программирования, с помощью которых пользователи создают свои программы;

– обеспечение работы систем управления базами данных (СУБД);

– обеспечение работы систем управления файлами (СУФ).

Операционная система, выполняя функции управления вычислительными процессами в вычислительной системе, распределяет ресурсы вычислительной системы между различными вычислительными процессами и образует программную среду, в которой выполняются прикладные программы пользователей, называемую операционной средой [1].

Таким образом, можно сказать, что ***операционная среда*** — это набор соответствующих интерфейсов, необходимых программам и пользователям для обращения к ОС с целью получения определенных сервисов.

Из всех перечисленных функций операционных систем следует остановиться особо на обеспечении работы систем управления файлами, назначение которой состоит в организации удобного доступа к данным, организованным как файлы. Особое внимание к этой функции обусловлено тем, что СУФ можно выделить в отдельную категорию ПО [1], поскольку имеются ОС, позволяющие работать с несколькими файловыми системами, и в этом смысле они самостоятельны. Более того, существуют ОС, которые могут работать и без файловых систем, а значит, им необязательно иметь систему управления файлами. Любая СУФ не существует сама по себе — она предназначена для работы в конкретной ОС и с конкретной файловой системой.

Для удобства взаимодействия с ОС могут использоваться дополнительные ***интерфейсные оболочки***. Их основное назначение — расширение возможностей по управлению ОС и изменение встроенных в систему возможностей под конкретные требования пользователя. В качестве классических примеров интерфейсных оболочек и соответствующих операционных сред

выполнения программ можно назвать различные варианты графического интерфейса X Windows в системах семейства UNIX, PM Shell или Object Desktop в OS/2 с графическим интерфейсом Presentation Manager; разнообразные варианты интерфейсов для семейства ОС Windows компании Microsoft, которые заменяют Explorer и обладают функциями графического интерфейса таких ОС, как UNIX, OS/2 либо MAC OS. Следует отметить, что о семействе ОС компании Microsoft с общим интерфейсом, реализуемым программными модулями с названием Explorer (в файле system.ini, находящемся в каталоге Windows, имеется строка SHELL=EXPLORER.EXE), все же можно сказать, что заменяемой в этих системах является только интерфейсная оболочка, в то время как сама операционная среда остается неизменной (она интегрирована в ОС). Другими словами, операционная среда определяется программным интерфейсом ***API (Application Program Interface)***, включающим в себя управление процессами, памятью и вводом/выводом.

Существуют операционные системы, способные организовывать выполнение программ, созданных для других ОС. Например, в OS/2 наряду с выполнением собственных программ могут использоваться программы, предназначенные для выполнения в среде MS DOS и Windows3.x. Соответствующая операционная среда организуется в ОС в рамках отдельной виртуальной машины. Аналогично, в системе Linux можно создать условия для выполнения некоторых программ, написанных для Windows 95/98/Me. Определенными возможностями исполнения программ, созданных для иной операционной среды, обладают ОС на платформе Windows NT. Эта система позволяет выполнять некоторые программы, созданные для MS DOS, OS/2, Windows3.x.

К сервисным программам ОС относятся и ***эмуляторы***, позволяющие смоделировать в одной операционной системе какую-либо виртуальную машину или операционную систему. Так, известна система эмуляции WMWARE, которая позволяет запустить в среде Linux любую другую ОС, например Windows. Можно, наоборот, создать эмулятор, работающий в среде Windows, который позволит смоделировать компьютер, функ-

ционирующий под управлением любой ОС, в том числе и под Linux.

В составе ОС присутствуют ***сервисные программы (утилиты ОС).*** Это специальные системные программы, с помощью которых можно как обслуживать саму операционную систему, так и подготавливать для работы носители данных, выполнять перекодирование данных, осуществлять оптимизацию размещения данных на носителе и производить некоторые другие работы, связанные с обслуживанием вычислительной системы. В качестве утилит также можно рассматривать такие программы, как программы разбиения на разделы накопителя на магнитных дисках, форматирования, переноса основных системных файлов самой ОС. К утилитам относятся и небезызвестные комплексы программ от фирмы Symantec, носящие имя Питера Нортона (создателя этой фирмы и соавтора популярного набора утилит для первых IBM PC). Естественно, что утилиты могут работать только в соответствующей операционной среде.

1.2 Классификация операционных систем

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями используемых методов проектирования, типов аппаратных платформ, областей применения и многими другими свойствами.

Остановимся на основных классификационных признаках:

1. **По назначению** ОС делятся на универсальные и специализированные. Специализированные ОС работают с фиксированным набором программ (функциональных задач). Применение таких систем обусловлено невозможностью использования универсальной ОС по соображениям эффективности, надежности, защищенности и т.п., а также вследствие специфики решаемых задач [2].

Универсальные ОС рассчитаны на решение любых задач пользователей, но, как правило, форма эксплуатации вычислительной системы может представлять особые требования к ОС, т.е. к элементам ее специализации.

2. **По способу загрузки** выделяют загружаемые ОС и системы, постоянно находящиеся в памяти вычислительной системы. Последние, как правило, специализированные и используются для управления работой специализированных устройств (в ракетах, спутниках, научных приборах и т.п.) [3].

3. **По особенностям алгоритмов управления ресурсами.** Данную классификацию можно разделить на несколько подгрупп:

3.1. ***Поддержка многопользовательского режима*** (рис. 1.1). По числу одновременно работающих пользователей выделяют: однопользовательские (MS DOS, ранние версии OS/2); многопользовательские (UNIX, Windows на платформе NT).

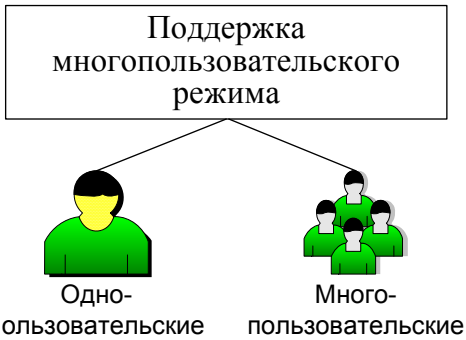

Рис. 1.1 — Классификация ОС с учетом поддержки многопользовательского режима

В однопользовательской системе все функциональные устройства компьютера находятся в полном распоряжении одного пользователя. Многопользовательская система включает компьютер, предназначенный для обслуживания многих пользователей, и набор терминалов, обеспечивающих взаимодействие пользователей с этим компьютером. Как правило, в состав терминала входит монитор, клавиатура и мышь, а в современных системах — еще и мини-компьютер с процессором и памятью для поддержки графического пользовательского интерфейса. Многопользовательские системы часто содержат большое количество

терминалов, объединенных в группы, которыми управляют специальные компьютеры — терминальные концентраторы [4].

Главное отличие многопользовательских систем от однопользовательских — наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей.

3.2. ***Многопроцессорная обработка*** (рис. 1.2). Одним из важных свойств ОС является отсутствие или наличие в ней средств поддержки многопроцессорной обработки. В настоящее время становится общепринятым введение в ОС функций поддержки многопроцессорной обработки данных. Такие функции имеются в операционных системах Solaris фирмы Sun, Open Server компании Santa Crus Operations, OS/2 фирмы IBM, Windows NT фирмы Microsoft и NetWare фирмы Novell.

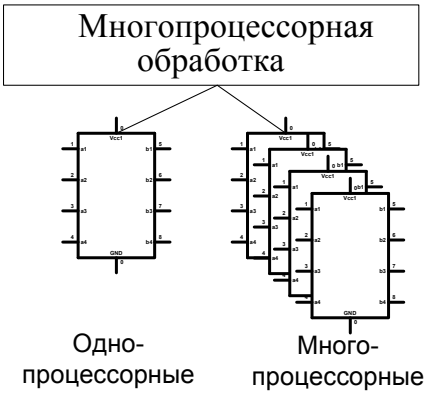

Рис. 1.2 — Классификация ОС с учетом поддержки многопроцессорной обработки

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

3.3. ***Поддержка многозадачности*** (рис. 1.3). По числу одновременно выполняемых задач ОС могут быть разделены на два класса:

– однозадачные (MS DOS, MSX);
– многозадачные (OS/2, UNIX, Windows).

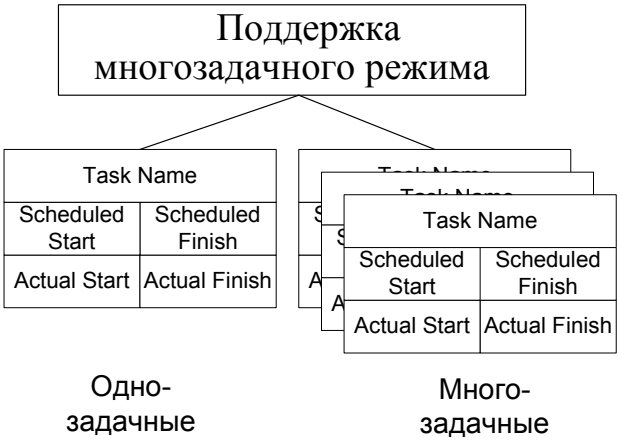

Рис. 1.3 — Классификация ОС с учетом поддержки многозадачного режима

Однозадачные ОС в основном выполняют функцию представления 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.

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

– системы пакетной обработки (ОС ЕС),
– системы разделения времени (UNIX, Windows),
– системы реального времени (QNX).

3.3.1. ***Системы пакетной обработки*** предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки являлась максимальная пропускная способность, то есть решение максимального числа задач в единицу времени. Для достижения этой цели в системах пакетной обработки используется следующая схема функционирования: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из 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 после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя. В настоящее время системы пакетной обработки практически не используются.

3.3.2. ***Системы разделения времени*** призваны исправить основной недостаток систем пакетной обработки — изоляцию пользователя-программиста от процесса выполнения его задач. Каждому пользователю системы разделения времени предос-

тавляется 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 с разделением времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

3.3.3. ***Системы реального времени*** применяются для управления различными техническими объектами (станками, спутниками, научными экспериментальными установками) 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 данных 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.

4. **По аппаратной платформе.** По типу вычислительной техники, для которой предназначаются операционные системы, их делят на следующие группы [3]:

4.1 ***Операционные системы для смарт-карт***. Некоторые из них могут управлять только одной операцией, например электронным платежом. Некоторые смарт-карты являются JAVA-ориентированными и содержат интерпретатор виртуальной машины JAVA. Апплеты JAVA загружаются на карту и выполняются JVM-интерпретатором. Некоторые из таких карт могут одновременно управлять несколькими апплетами JAVA, что приводит к многозадачности и необходимости планирования.

4.2 ***Встроенные операционные системы***. Управляют карманными компьютерами (Palm OS, Windows CE — Consumer Electronics — бытовая техника), мобильными телефонами, телевизорами, микроволновыми печами и т. п.

4.3 ***Операционные системы для персональных компьютеров***, например Windows 9.x, Windows 2000, Linux, Mac OSX и др.

4.4 ***Операционные системы мини-ЭВМ***, например RT-11 для PDP-11 — ОС реального времени, RSX-11 M для PDP-11 — ОС разделения времени, UNIX для PDP-7.

4.5 ***Операционные системы мэйнфреймов*** (больших машин), например OS/390, произошедшая от OS/360 (IBM). Обычно ОС мэйнфреймов предполагает одновременно три вида обслуживания: пакетную обработку, обработку транзакций (например, работа с БД, бронирование авиабилетов, процесс работы в банках) и разделение времени.

4.6 ***Серверные операционные системы***, например UNIX, Windows 2000, Linux. Область применения — ЛВС, региональные сети, Интернет, интранет.

4.7 ***Кластерные операционные системы***. Кластер — слабо связанная совокупность нескольких вычислительных систем,

работающих совместно для выполнения общих приложений и представляющихся пользователю единой системой, например Windows 2000 Cluster Server, Sun Cluster (базовая ОС — Solaris).

1.3 Классификация построений ядер операционных систем

По основному архитектурному принципу ОС разделяются на ***микроядерные и монолитные*** операционные системы. В некоторой степени это разделение тоже условно, однако можно в качестве яркого примера микроядерной операционной системы привести систему реального времени QNX, тогда как в качестве примера монолитной ОС можно назвать ОС Windows или ОС Linux. Ядро ОС Windows нельзя изменить, пользователям недоступны его исходные коды и у них нет программы для сборки (компиляции) этого ядра. А вот в случае работы с ОС Linux пользователи могут сами собрать ядро, отвечающее их потребностям, включив в него программные модули и драйверы, которые целесообразно, по их мнению, включить именно в ядро.

Основой модульных и переносимых расширений является ***микроядро*** — минимальная стержневая часть операционной системы. Существует мнение, что большинство операционных систем следующих поколений будут обладать микроядрами. ***Основная идея, заложенная в технологию микроядра, будь то собственно ОС или ее графический интерфейс, заключается в том, чтобы конструировать необходимую среду верхнего уровня, из которой можно легко получить доступ ко всем функциональным возможностям уровня аппаратного обеспечения*** [1]. При такой структуре ядро служит стартовой точкой для создания системы. Искусство разработки микроядра заключается в выборе базовых примитивов, которые должны в нем находиться для обеспечения необходимого и достаточного сервиса. В микроядре содержится и исполняется минимальное количество кода, необходимое для реализации основных системных вызовов.

Остальные функции, характерные для «обычных» (не микроядерных) ОС, обеспечиваются как модульные дополнения-процессы, взаимодействующие главным образом между собой и

осуществляющие взаимодействие посредством передачи сообщений.

Исполняемые микроядром функции ограничены в целях сокращения его размеров и максимизации количества кода, работающего как прикладная программа.

Микроядро включает только те функции, которые требуются для определения набора абстрактных сред обработки прикладных программ и организации совместной работы приложений при обеспечении сервисов и взаимодействия клиентами и серверами. В результате микроядро обеспечивает, как правило, не более пяти различных типов сервисов:

– управление виртуальной памятью;
– управление заданиями и потоками;
– поддержка межпроцессных коммуникаций IPC (Inter-Process Communication);
– управление поддержкой ввода/вывода и прерываниями;
– поддержка сервисов набора хоста (Host — главный компьютер; в настоящее время этим термином обозначают любой компьютер, имеющий IP-адрес) и процессора.

Наиболее ярким представителем микроядерных ОС является ОС реального времени QNX (Queue Nicks, с англ.: очередь точных моментов времени, или очередь зарубок). Микроядро QNX поддерживает только планирование и диспетчеризацию процессов, взаимодействие процессов, обработку прерываний и сетевые службы нижнего уровня. Микроядро обеспечивает всего лишь около двух десятков системных вызовов, но благодаря этому оно может быть целиком размещено во внутреннем кэше даже таких процессоров, как Intel 486. Как известно, разные версии этой ОС имели и различные объемы ядер (от нескольких килобайт до нескольких десятков килобайт).

Чтобы построить минимальную систему QNX, требуется добавить к микроядру менеджер процессов, который создает процессы, управляет процессами и памятью процессов. Чтобы ОС QNX была применима не только во встроенных и бездисковых системах, нужно добавить файловую систему и менеджер устройств. Эти менеджеры исполняются вне пространства ядра, так что ядро остается небольшим.

Монолитные ОС являются прямой противоположностью микроядерным ОС. В монолитной ОС, несмотря на ее возможную сильную структуризацию, достаточно трудно удалить какой-либо уровень многоуровневой модульной структуры. Добавление новых функций и изменение существующих для монолитных ОС требует глубокого знания всей архитектуры ОС и чрезвычайно больших усилий. Поэтому более современный подход к проектированию ОС, который может быть условно назван «клиент-серверной технологией», позволяет в большем количестве и с меньшими трудозатратами реализовать перечисленные выше принципы проектирования ОС [1].

Модель клиент-сервер предполагает наличие программного компонента, являющегося потребителем какого-либо сервиса, — **клиента** и программного продукта, служащего поставщиком этого сервиса, — **сервера**. Взаимодействие между клиентом и сервером стандартизируется, так что сервер может обслуживать клиентов, реализованных разными способами и, может быть, разными разработчиками. При этом главным требованием является использование единообразного интерфейса. Инициатором обмена обычно является клиент, который посылает запрос на обслуживание серверу, находящемуся в состоянии ожидания запроса. Один и тот же программный компонент может быть клиентом по отношению к одному виду услуг и сервером для другого вида услуг. Модель клиент-сервер является скорее удобным концептуальным средством ясного представления функций того или иного программного элемента в какой-либо ситуации, нежели технологией. Эта модель успешно применяется не только при построении операционных систем, но и на всех уровнях программного обеспечения и имеет в некоторых случаях более узкий, специфический смысл, сохраняя, естественно, при этом все свои общие черты.

При поддержке монолитных ОС возникает ряд проблем, связанных с тем, что все функции макроядра работают в едином адресном пространстве. ***Во-первых,*** это опасность возникновения конфликта между различными частями ядра; ***во-вторых*** — сложность подключения к ядру новых драйверов. Преимущество микроядерной архитектуры перед монолитной заключается в том, что каждый компонент системы представляет собой само-

стоятельный процесс, запуск или остановка которого не отражается на работоспособности остальных процессов. Микроядерные ОС в настоящее время разрабатываются чаще монолитных. Однако следует заметить, что использование технологии клиент-сервер — это еще не гарантия того, что ОС станет микроядерной. В качестве подтверждения можно привести пример с ОС Windows NT, которая построена на идеологии клиент-сервер, но которую трудно назвать микроядерной.

1.4 Представление об интерфейсах прикладного программирования

1.4.1 Общие задачи и функции интерфейсов прикладного программирования

ОС всегда выступает как интерфейс между аппаратурой компьютера и пользователем с его задачами. Под интерфейсами операционных систем здесь и далее следует понимать специальные интерфейсы системного и прикладного программирования, предназначенные для выполнения следующих задач:

- управления процессами, которое включает в себя следующий набор основных функций:
  - запуск, приостановка и снятие задачи с выполнения;
  - назначение или изменение приоритета задачи;
  - организация взаимодействия задач между собой (механизмы сигналов, семафорные примитивы, очереди, конвейеры, почтовые ящики);
  - организации удаленного вызова подпрограмм RPC (Remote Procedure Call);
- управления памятью:
  - запрос на выделение блока памяти;
  - освобождение памяти;
  - изменение параметров блока памяти (например, память может быть заблокирована процессом либо предоставлена в общий доступ);
  - отображение файлов на память (имеется не во всех системах);
- управления вводом/выводом:

– запрос на управление виртуальными устройствами. Напомним, что управление вводом/выводом является привилегированной функцией самой ОС и никакая из пользовательских задач не должна иметь возможности непосредственно управлять устройствами;

– файловые операции (запросы к системе управления файлами на создание, изменение и удаление данных, организованных в файлы).

Здесь были перечислены основные наборы функций, которые выполняются ОС по соответствующим запросам от задач. Что касается пользовательского интерфейса операционной системы, то он реализуется с помощью специальных программных модулей, которые принимают его команды на соответствующем языке (возможно, с использованием графического интерфейса) и транслируют их в обычные вызовы в соответствии с основным интерфейсом системы. Обычно эти модули называют интерпретатором команд. Так, например, функции такого интерпретатора в MS DOS выполняет модуль COMMAND.COM. Получив от пользователя команду, такой модуль после лексического и синтаксического анализа либо сам выполняет действие, либо, что случается чаще, обращается к другим модулям ОС, используя механизм API. Надо заметить, что в последние годы большую популярность получили графические интерфейсы GUI (Graphical User Interface), в которых задействованы соответствующие манипуляторы типа «мышь» или другие. Указание курсором на объекты и щелчок (клик) или двойной щелчок по соответствующим клавишам приводит к каким-либо действиям — запуску программы, ассоциированной с указываемым объектом, выбору и/или активизации пунктов меню и т.д. Можно сказать, что такая интерфейсная подсистема транслирует команды пользователя в обращения к ОС.

Поясним также, что управление GUI — частный случай задачи управления вводом/выводом, не являющийся частью ядра операционной системы, хотя в ряде случаев разработчики ОС относят функции GUI к основному системному API. Следует отметить, что имеются два основных подхода к управлению задачами. Так, в одних системах порождаемая задача наследует все ресурсы задачи-родителя, тогда как в других системах суще-

ствуют равноправные отношения, и при порождении нового процесса ресурсы для него запрашиваются у операционной системы.

Обращение к операционной системе в соответствии с имеющимся API может осуществляться как посредством вызова подпрограммы с передачей ей необходимых параметров, так и через механизм программных прерываний. Выбор метода реализации вызовов функций API должен определяться архитектурой платформы.

Так, например, в операционной системе MS DOS, которая разрабатывалась для однозадачного режима, использовался механизм программных прерываний. При этом основной набор функций API был доступен через точку входа обработчика прерываний int 21h. В более сложных системах имеется не одна точка входа, а множество — по количеству функций API. Так, в большинстве операционных систем используется метод вызова подпрограмм. В этом случае вызов (например, это функция **библиотеки времени выполнения RTL**[1] (Run Time Library), сначала передается в модуль API, который и перенаправляет вызов соответствующим обработчикам программных прерываний, входящим в состав операционной системы. Использование механизма прерываний вызвано, главным образом, тем, что при этом процессор переводится в режим супервизора.

1.4.2 Варианты реализации интерфейсов прикладного программирования

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

[1] *RTL* 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.

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, порождаемая 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. 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, 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 [1]:

– эффективностью выполнения функций API, характеризуемой скоростью выполнения функций и объемом вычислительных ресурсов, требующихся для их выполнения;
– широтой предоставляемых возможностей;

– степенью зависимости 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, а 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.

1.4.3 Характеристики интерфейсов прикладного программирования на различных уровнях реализаций

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 составе динамически загружаемых библиотек, разработанных для данной ОС. Система программирования ответственна только за организацию интерфейса для вызова этого кода [1].

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 архитектуре просто отсутствовать.

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

Перенос можно осуществить при условии унифицирования функций 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 функций 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. Другим 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. Система программирования предоставляет пользователю библиотеку соответствующего языка программирования и обеспечивает

подключение к результирующей программе объектного кода, ответственного за выполнение этих функций [1].

Очевидно, что эффективность функций API в таком варианте будет несколько ниже, чем при непосредственном обращении к функциям ОС. Это происходит вследствие того, что 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.

Например, рассмотрим функции динамического выделения памяти в языках C и Pascal. В языке C это функции malloc, realloc и free (функции new и delete в C++), в Pascal — функции new и dispose. Если использовать эти функции в исходном тексте программы, то с точки зрения исходной программы они будут действовать одинаковым образом и зависеть только от семантики исходного кода. При этом для разработчика исходной программы не имеет значения, на какую архитектуру ориентирова-

на его программа. Вышесказанное имеет значение для системы программирования, которая для каждой из этих функций должна подключить к результирующей программе объектный код библиотеки, выполняющий обращение к соответствующим функциям ОС. Не исключено даже, что для однотипных по смыслу функций в разных языках (например, malloc в C и new в языке Pascal выполняют схожие по смыслу действия) этот код будет выполнять схожие обращения к ОС. Однако для различных вариантов ОС код будет различен даже при использовании одного и того же исходного языка.

Проблема, главным образом, заключается в том, что большинство языков программирования предоставляют пользователю недостаточно широкий набор стандартизованных функций. Поэтому разработчик исходного кода существенно ограничен в выборе доступных функций API. Как правило, набора стандартных функций оказывается недостаточно для создания полноценной прикладной программы. Тогда разработчик может обратиться к функциям других библиотек, имеющихся в составе системы программирования. В этом случае нет гарантии, что функции, включенные в состав данной системы программирования, но не входящие в стандарт языка программирования, будут доступны в другой системе программирования, особенно если она ориентирована на другую архитектуру целевой вычислительной системы. Такая ситуация уже ближе к третьему варианту реализации API.

Например, те же функции malloc, realloc и free в языке C фактически не входят в стандарт языка. Они входят в состав стандартной библиотеки, которая де-факто входит во все системы программирования, построенные на основе языка C. Общепринятые стандарты существуют для многих часто используемых функций языка. Если же взять более специфические функции, такие, как функции порождения новых процессов, то для них ни в C, ни в языке Pascal не окажется общепринятого стандарта.

Реализация функций API с помощью внешних библиотек

Реализация функций API с помощью внешних библиотек осуществляется посредством предоставления пользователю дан-

ных функций в виде библиотеки процедур и функций, созданной сторонним разработчиком [4]. Причем разработчиком такой библиотеки может выступать тот же самый производитель.

Система программирования ответственна только за подключение объектного кода библиотеки к результирующей программе. Причем внешняя библиотека может быть и динамически загружаемой, т.е. 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, 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, кросс-платформенная библиотека компонент CLX (Component Library for Cross-platform, «кликс») производства фирмы Borland, ориентированная на архитектуру ОС типа Linux и ОС типа Windows.

В целом развитие функций прикладного API идет в направлении попыток создания библиотек API, обеспечивающих 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.

Поэтому разработчики системных программ вынуждены оставаться в довольно узких рамках ограничений стандартных библиотек языков программирования.

Что касается прикладных программ, то большую перспективу для них предоставляют технологии, связанные с разработками в рамках архитектуры «клиент-сервер» или трехуровневой архитектуры создания приложений. В этом направлении ведущие производители ОС, СУБД и систем программирования скорее достигнут соглашений, чем в направлении стандартизации API.

Отметим еще один очень важный момент: желательно, чтобы интерфейс прикладного программирования не зависел от системы программирования. Конечно, имелись персональные компьютеры, у которых базовой системой программирования выступал интерпретатор с языка Basic, но это скорее исключение. Обычно базовые функции API не зависят от системы программирования и могут использоваться из любой системы программирования, хотя и с применением соответствующих правил построения вызывающих последовательностей. В то же время в ряде случаев система программирования может сама генерировать обращения к функциям API.

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 из одной ОС в другую. Таким примером может служить известный и, пожалуй, самый распространенный стандарт — стандарт POSIX (см. пункт 1.4.4). 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 команд (команд управления процессами). Использование в приложениях этого стандарта позволяет в дальнейшем легко переносить такие программы из 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: Win16, Win32s, Win32, WinCE. С точки зрения WinAPI, который в силу ряда идеологических причин является обязательным графическим «оконным» интерфейсом пользователя, базовой задачей выступает окно. Таким образом, WinAPI изначально ориентирован на работу в графической среде. Однако базовые понятия дополнены традиционными функциями, в том числе частично поддерживается стандарт POSIX.

1.4.4 Платформенно-независимый интерфейс POSIX

Платформенно-независимый системный интерфейс для компьютерного окружения POSIX (Portable Operating System Interface for Computer Environments) — это стандарт IEEE (Institute of Electrical and Electronics Engineers — институт инженеров по электротехнике и радиоэлектронике), описывающий системные интерфейсы для открытых ОС, в том числе оболочки, утилиты и инструментарии

(http://standards.ieee.org/regauth/posix/index.html).

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. 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 систему виртуальной памяти VMS (Virtual Memory System), многозадачность MPE (MultiProcess Executing) и технологию переноса операционных систем CTOS (An Operating System produced Convergent Technology …). Таким образом, на самом деле POSIX представляет собой множество стандартов, именуемых POSIX.I — POSIX.12. В табл. 1.1 приведены основные направления, описываемые данными стандартами. Следует также особо отметить, что в POSIX. 1 предполагается язык С в качестве основного языка описания системных функций API.

Таблица 1.1 — **Семейство стандартов POSIX**

Стандарт	Стандарт ISO	Краткое описание
POSIX.0	Нет	Введение в стандарт открытых систем. Данный документ не является стандартом в чистом виде, а представляет собой рекомендации и краткий обзор технологий
POSIX.1	Да	Системный API (язык С)
POSIX.2	Нет	Оболочки и утилиты, одобренные IEEE

Окончание табл. 1.1

Стандарт	Стандарт ISO	Краткое описание
POSIX.3	Нет	Тестирование и верификация
POSIX.4	Нет	Задачи реального времени и нити
POSIX.5	Да	Использование языка ADA применительно к стандарту POSIX.1
POSIX.6	Нет	Системная безопасность
POSIX.7	Нет	Администрирование системы
POSIX.8	Нет	Сети «Прозрачный» доступ к файлам Абстрактные сетевые интерфейсы, не зависящие от физических протоколов вызова удаленных процедур RPC (Remote Procedure Calls) Связь системы с протоколо-зависимыми приложениями
POSIX.9	Да	Использование языка FORTRAN применительно к стандарту POSIX.1
POSIX.10	Нет	Super-computing Application Environment Profile (AEP) — профиль для суперкомпьютерных окружений
POSIX.11	Нет	Обработка транзакций AEP
POSIX.12	Нет	Графический интерфейс пользователя GUI

Таким образом, программы, написанные с соблюдением данных стандартов, будут одинаково выполняться на всех POSIX-совместимых системах. Однако стандарт в некоторых случаях носит лишь рекомендательный характер. Часть стандартов описана очень строго, тогда как другая часть только поверхностно раскрывает основные требования.

Нередко программные системы заявляют как POSIX-совместимые, хотя таковыми их назвать нельзя. Причины кроются в формальности подхода к реализации POSIX-интерфейса в различных ОС. К сожалению, достаточно часто с целью увеличения производительности той или иной подсистемы либо из соображений введения фирменных технологий, которые ограничивают использование приложения соответствующей операционной

средой, при программировании используются другие функции, не отвечающие стандарту POSIX.

Реализации POSIX API на уровне операционной системы различны. Если UNIX-системы в своем абсолютном большинстве изначально соответствуют спецификациям IEEE Standard 1003.1-1990, то WinAPI не является POSIX-совместимым. Однако для поддержки данного стандарта в операционной системе MS Windows NT введен специальный модуль поддержки POSIX API, работающий на уровне привилегий пользовательских процессов. Данный модуль обеспечивает конвертацию и передачу вызовов из пользовательской программы к ядру системы и обратно, работая с ядром через Win API. Прочие приложения, созданные с использованием WinAPI, могут передавать информацию POSIX-приложениям через стандартные механизмы потоков ввода/вывода (stdin, stdout).

1.5 Основные принципы построения операционных систем

В заключение к вышеприведенным понятиям и их классификациям, встречающимся в предмете «операционные системы», приведем основные принципы построения современных операционных систем [4]:

Принцип модульности

Под **модулем** в общем случае понимают функционально законченный элемент системы, выполненный в соответствии с принятыми межмодульными интерфейсами. По своему определению модуль предполагает возможность легкой замены его на другой при наличии заданных интерфейсов. Способы обособления составных частей ОС в отдельные модули могут существенно различаться, но чаще всего разделение происходит именно по функциональному признаку. В значительной степени разделение системы на модули определяется используемым методом проектирования ОС (снизу вверх или наоборот).

Особое значение при построении ОС имеют ***привилегированные, повторно входимые и реентерабельные модули*** (рен-

терабельность — (буквально, повторновходимость; специальный термин для обозначения работоспособности программы) — свойство программы — корректно выполняться при рекурсивном (возвращаемом) вызове из прерываний), так как они позволяют более эффективно использовать ресурсы вычислительной системы. В некоторых системах реентерабельность программы получают автоматически, благодаря неизменяемости кодовых частей программ при исполнении вследствие особенностей системы команд машины, а также автоматическому распределению регистров, автоматическому отделению кодовых частей программ от данных и помещению последних в системную область памяти. Естественно, что для этого необходима соответствующая аппаратная поддержка или специальные системные модули.

Принцип функциональной избирательности

В ОС выделяется некоторая часть важных модулей, которые должны постоянно находиться в оперативной памяти для более эффективной организации вычислительного процесса. Эту часть в ОС называют ядром, так как это — основа системы. При формировании состава ядра требуется учитывать два противоречивых требования. С одной стороны, в состав ядра должны войти наиболее часто используемые системные модули, с другой — количество модулей должно быть таковым, чтобы объем памяти, занимаемый ядром, не был слишком большим. В состав ядра, как правило, входят модули по управлению системой прерываний, средства по переводу программ из состояния счета в состояние ожидания, готовности и обратно, средства по распределению таких основных ресурсов, как оперативная память и процессор. Помимо программных модулей, входящих в состав ядра и постоянно располагающихся в оперативной памяти, может быть много других системных программных модулей, которые получают название ***транзитных***. Транзитные программные модули загружаются в оперативную память только при необходимости и в случае отсутствия свободного пространства могут быть замещены другими транзитными модулями.

Принцип генерируемости ОС

Суть принципа состоит в организации (выборе) такого способа исходного представления центральной системной управляющей программы ОС (ядра и постоянно находящихся в оперативной памяти основных компонентов), который позволял бы настраивать эту системную супервизорную часть исходя из конкретной конфигурации конкретного вычислительного комплекса и круга решаемых задач. Эта процедура проводится редко перед достаточно протяженным периодом эксплуатации ОС. Процесс генерации осуществляется с помощью специальной программы-генератора и соответствующего входного языка для этой программы, позволяющего описывать программные возможности системы и конфигурацию машины. В результате генерации получается полная версия ОС.

В настоящее время при использовании персональных компьютеров с принципом генерируемости ОС можно столкнуться разве что только при работе с Linux. В этой UNIX-системе имеется возможность не только использовать какое-либо готовое ядро ОС, но и самому сгенерировать (скомпилировать) такое ядро, которое будет оптимальным для данного конкретного персонального компьютера и решаемых на нем задач. Кроме генерации ядра, в Linux имеется возможность указать и набор подгружаемых драйверов и служб, то есть часть функций может реализовываться модулями, непосредственно входящими в ядро системы, а часть — модулями, имеющими статус подгружаемых, транзитных.

В остальных современных распространенных ОС для персональных компьютеров конфигурирование ОС под соответствующий состав оборудования осуществляется на этапе инсталляции, а потом состав драйверов и изменение некоторых параметров ОС может быть осуществлено посредством редактирования конфигурационного файла.

Принцип функциональной избыточности

Этот принцип учитывает возможность проведения одной и той же работы различными средствами. В состав ОС может входить несколько типов мониторов (модулей супервизора, управ-

ляющих тем или другим видом ресурса), различные средства организации коммуникаций между вычислительными процессами. Наличие нескольких типов мониторов, нескольких систем управления файлами позволяет пользователям быстро и наиболее адекватно адаптировать ОС к определенной конфигурации вычислительной системы, обеспечивать максимально эффективную загрузку технических средств при решении конкретного класса задач, получать максимальную производительность при решении заданного класса задач.

Принцип виртуализации

Построение виртуальных ресурсов, их распределение и использование в настоящее время используется практически в любой ОС. Этот принцип позволяет представить структуру системы в виде определенного набора планировщиков процессов и распределителей ресурсов (мониторов) и использовать единую централизованную схему распределения ресурсов.

Наиболее естественным и законченным проявлением концепции виртуальности является понятие **виртуальной машины**. По сути, любая операционная система, являясь средством распределения ресурсов и организуя по определенным правилам управление процессами, скрывает от пользователя и его приложений реальные аппаратные и иные ресурсы, заменяя их некоторой абстракцией. В результате пользователи видят и используют виртуальную машину как некое устройство, способное воспринимать их программы, написанные на определенном языке программирования, выполнять их и выдавать результаты. При таком языковом представлении пользователя совершенно не интересует реальная конфигурация вычислительной системы, способы эффективного использования ее компонентов и подсистем. Он мыслит и работает с машиной в терминах используемого им языка и тех ресурсов, которые ему предоставляются в рамках виртуальной машины.

Одним из аспектов виртуализации является организация возможности выполнения в данной ОС приложений, которые разрабатывались для других ОС. Другими словами, речь идет об организации нескольких операционных сред. Реализация этого

принципа позволяет данной ОС иметь очень сильное преимущество перед аналогичными ОС, не имеющими такой возможности. Примером реализации принципа виртуализации может служить ***VDM-машина*** (Virtual DOS Machine) — защищенная подсистема, предоставляющая полную среду MS DOS и консоль для выполнения MS DOS-приложений. Одновременно может выполняться практически произвольное число VDM-сессий. Такие VDM-машины имеются и в системах Microsoft Windows, и в OS/2.

Принцип независимости программ от внешних устройств

Этот принцип реализуется сейчас в подавляющем большинстве ОС общего применения. Пожалуй, впервые наиболее последовательно данный принцип был реализован в ОС UNIX. Реализован он и в большинстве современных ОС для ПК. Этот принцип заключается в том, что связь программ с конкретными устройствами производится не на уровне трансляции программы, а в период планирования ее исполнения. В результате перекомпиляция при работе программы с новым устройством, на котором располагаются данные, не требуется.

Этот принцип позволяет осуществлять операции управления внешними устройствами независимо от их конкретных физических характеристик. Например, программе, содержащей операции обработки последовательного набора данных, безразлично, на каком носителе эти данные будут располагаться. Смена носителя и данных, размещаемых на них при неизменности структурных характеристик данных, не принесет каких-либо изменений в программу, если в системе реализован принцип независимости.

Принцип совместимости

Одним из аспектов совместимости является способность ОС выполнять программы, написанные для других ОС или для более ранних версий данной операционной системы, а также для другой аппаратной платформы. Необходимо разделять вопросы двоичной совместимости и совместимости на уровне исходных

текстов приложений. Двоичная совместимость достигается в том случае, когда можно взять исполняемую программу и запустить ее на выполнение на другой ОС. Для этого необходимы совместимость на уровне команд процессора и совместимость на уровне системных вызовов и даже на уровне библиотечных вызовов, если они являются динамически связываемыми. Совместимость на уровне исходных текстов требует наличия соответствующего транслятора в составе системного программного обеспечения, а также совместимости на уровне библиотек и системных вызовов. При этом необходима перекомпиляция имеющихся исходных текстов в новый выполняемый модуль.

Принцип открытости и наращиваемости

Открытая операционная система доступна для анализа как пользователям, так и системным специалистам, обслуживающим вычислительную систему. Наращиваемая (модифицируемая, развиваемая) ОС позволяет не только использовать возможности генерации, но и вводить в ее состав новые модули, совершенствовать существующие и т.д. Другими словами, следует обеспечить возможность легкого внесения дополнений и изменений в необходимых случаях без нарушения целостности системы. Прекрасные возможности для расширения предоставляет подход к структурированию ОС по типу «клиент-сервер» с использованием микроядерной технологии. В соответствии с этим подходом ОС строится как совокупность привилегированной управляющей программы и набора непривилегированных услуг (серверов). Основная часть ОС остается неизменной и в то же время могут быть добавлены новые серверы или улучшены старые. Этот принцип иногда трактуют как **расширяемость системы**. К открытым ОС, прежде всего, следует отнести UNIX-системы и, естественно, ОС Linux.

Принцип мобильности (переносимости)

Операционная система относительно легко должна переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы одного типа, которая включает наряду с типом процессора и способ организации всей аппаратуры ком-

пьютера (архитектуру вычислительной системы), на аппаратную платформу другого типа. Заметим, что принцип переносимости очень близок принципу совместимости, хотя это и не одно и то же. Создание переносимой ОС аналогично написанию любого переносимого кода, при этом нужно следовать некоторым правилам. Во-первых, большая часть ОС должна быть выполнена на языке, имеющемся на всех системах, на которые планируется в дальнейшем ее переносить. Это, прежде всего, означает, что ОС должна быть написана на языке высокого уровня, предпочтительно стандартизованном, например на языке С. Программа, написанная на ассемблере, не является в общем случае переносимой. Во-вторых, важно минимизировать или, если возможно, исключить те части кода, которые непосредственно взаимодействуют с аппаратными средствами. Наконец, если аппаратно-зависимый код не может быть полностью исключен, то он должен быть изолирован в нескольких хорошо локализуемых модулях. Аппаратно-зависимый код не должен быть распределен по всей системе. Например, можно спрятать аппаратно-зависимую структуру в программно задаваемые данные абстрактного типа. Другие модули системы будут работать с этими данными, а не с аппаратурой, используя набор некоторых функций. Когда ОС переносится, то изменяются только эти данные и функции, которые ими манипулируют.

Принцип обеспечения безопасности вычислений

Обеспечение безопасности при выполнении вычислений является желательным свойством для любой многопользовательской системы. Правила безопасности определяют такие свойства, как защиту ресурсов одного пользователя от других и установление квот по ресурсам для предотвращения захвата одним пользователем всех системных ресурсов, таких, например, как память.

Обеспечение защиты информации от несанкционированного доступа является обязательной функцией сетевых операционных систем. Во многих современных ОС гарантируется степень безопасности данных, соответствующая уровню С2 в системе стандартов США. Основы стандартов в области безопасно-

сти были заложены в документе «Критерии оценки надежных компьютерных систем». Этот документ, изданный Национальным центром компьютерной безопасности NCSC (National Computer Security Center) в США в 1983 году, часто называют «Оранжевой книгой».

В соответствии с требованиями «Оранжевой книги» безопасной считается система, которая «посредством специальных механизмов защиты контролирует доступ к информации таким образом, что только имеющие соответствующие полномочия лица или процессы, выполняющиеся от их имени, могут получить доступ на чтение, запись, создание или удаление информации».

Иерархия уровней безопасности, приведенная в «Оранжевой книге», помечает низший уровень безопасности как D, а высший — как A.

В класс D попадают системы, оценка которых выявила их несоответствие требованиям всех других классов.

Основными свойствами, характерными для систем класса C, являются наличие подсистемы учета событий, связанных с безопасностью, и избирательный контроль доступа. Класс (уровень) C делится на 2 подуровня: уровень C1, обеспечивающий защиту данных от ошибок пользователей, но не от действий злоумышленников, и более строгий уровень C2. На уровне C2 должны присутствовать:

– средства секретного входа, обеспечивающие идентификацию пользователей путем ввода уникального имени и пароля, перед тем как им будет разрешен доступ к системе;

– избирательный контроль доступа, позволяющий владельцу ресурса определить, кто имеет доступ к ресурсу и что он может с ним делать. Владелец делает это путем предоставляемых прав доступа пользователю или группе пользователей;

– средства учета и наблюдения (auditing), обеспечивающие возможность обнаружения и фиксирования важных событий, связанных с безопасностью, или любых попыток создать, получить доступ или удалить системные ресурсы;

– защита памяти, заключающаяся в том, что память инициализируется перед повторным использованием.

Системы уровня B основаны на принципах пометки данных и распределения пользователей по категориям, реализующим мандатный контроль доступа. Каждому пользователю присваивается рейтинг защиты, и он может получать доступ к данным только в соответствии с этим рейтингом. Этот уровень в отличие от уровня C защищает систему от ошибочного поведения пользователя. Уровень A является самым высоким уровнем безопасности, он требует в дополнение ко всем требованиям уровня B выполнения формального, математически обоснованного доказательства соответствия системы требованиям безопасности. Различные коммерческие структуры (например, банки) особо выделяют необходимость учетной службы, аналогичной той, что предлагают государственные рекомендации C2. Любая деятельность, связанная с безопасностью, может быть отслежена и тем самым учтена. Это как раз то, чего требует стандарт для систем класса C2 и что обычно нужно банкам. Однако коммерческие пользователи, как правило, не хотят расплачиваться производительностью за повышенный уровень безопасности. A-уровень безопасности занимает своими управляющими механизмами до 90 % процессорного времени, что, безусловно, в большинстве случаев уже неприемлемо. Более безопасные системы не только снижают эффективность, но и существенно ограничивают число доступных прикладных пакетов, которые соответствующим образом могут выполняться в подобной системе. Например, для ОС Solaris (версия UNIX) есть несколько тысяч приложений, а для ее аналога B-уровня — только около ста.

Вопросы для самопроверки

1. Дайте объяснение понятиям операционной среды и операционной системы.
2. Назовите основные функции операционных систем.
3. Для чего используют интерфейсные оболочки? Приведите их примеры.
4. Что представляют собой эмуляторы ОС и сервисные программы ОС?
5. Приведите классификацию операционных систем.

6. Приведите классификацию построения ядер операционных систем.

7. Что понимают под интерфейсом прикладного программирования?

8. Что такое библиотека времени выполнения?

9. Классифицируйте функции API на различных уровнях реализации.

10. Дайте краткое описание стандарта POSIX.

11. Сформулируйте принципы построения операционных систем.

12. Что декларирует документ «Оранжевая книга»?

2 ОБЗОР ПОПУЛЯРНЫХ ОПЕРАЦИОННЫХ СИСТЕМ

2.1 Операционные системы фирмы Microsoft

2.1.1 История разработки операционных систем фирмы Microsoft

История ОС MS DOS

Операционная система MS DOS (Microsoft Disk Operation System) — **однозадачная, однопроцессорная, однопользовательская система для управления 16-разрядным персональным микрокомпьютером IBM PC.** Первой разработкой MS DOS можно считать операционную систему для персональных ЭВМ, созданную фирмой ***Seattle Computer Products*** в 1980 г. В конце 1980 г. система, первоначально названная QDOS, была модифицирована и переименована в 86-DOS. Право на использование операционной системы 86-DOS было куплено Корпорацией Microsoft, заключившей контракт с фирмой IBM и обязавшейся разработать ОС для новой модели персональных компьютеров, выпускаемых фирмой. Когда новый компьютер IBM PC приобрел широкую популярность (1981 г.), его операционная система представляла собой модифицированную версию системы 86-DOS, названную PC DOS, версия 1.0.

Вскоре после выпуска IBM PC на рынке стали появляться персональные компьютеры, «схожие с PC». Операционная система этих компьютеров, предоставленная в распоряжение производивших такие машины фирм корпорацией Microsoft, представляла собой точную копию операционной системы PC DOS и получила название MS DOS версия 1.0. Единственное серьезное различие этих систем состояло в «уровне системы», что означало необходимость приобретения собственной ОС для каждой машины при использовании ОС MS DOS. Отличительные особенности каждой системы мог выявить только системный программист, в чьи обязанности входила работа по «подгонке» операционной системы к конкретной машине. Пользователь, работающий на разных машинах, не ощущал никакой разницы между ними.

С момента выпуска операционные системы PC DOS и MS DOS совершенствовались параллельно и аналогичным образом. В 1982 году появились версии 1.1. Главным преимуществом новой версии была возможность использования двухсторонних дискет (версия 1.0 позволяла работать только с односторонними дискетами), а также возможность пересылки принтеровского вывода на другие устройства. В 1983 году были разработаны *версии 2.0*. По сравнению с предыдущими они давали возможность использовать жесткий диск, обеспечивали усложненный иерархический каталог диска, включали встроенные устройства для дискет и систему управления файлами. MS DOS *версии 3.0*, выпущенной в 1984 г., предоставляла улучшенный вариант обслуживания жесткого диска и подсоединенных к компьютеру микрокомпьютеров. Последующие версии, включая версию 3.3, появившуюся в 1987 г., развивались в том же направлении. MS DOS *версии 5.0* предоставляла возможность использования памяти, расположенной выше 1Мб. В MS DOS *версии 6.0* были расширены возможности использования памяти, расположенной выше 1Мб, добавлены утилита оптимизации использования памяти Memmaker и средство увеличения эффективного дискового пространства DoubleSpace. В комплект поставки включены утилиты проверки и оптимизации жесткого диска ScanDisk и Defrag.

История программного продукта Windows

Программный продукт с именем Windows получил свое развитие в 90-х годах, хотя первая версия была выпущена в ноябре 1985 г, но она не снискала популярности. В мае 1990 г. появилась весьма эффектная и вполне успешная версия Windows 3.0. Появление Windows 3.0 стало тем самым переломным этапом, в ходе которого мир внезапно открыл для себя возможности и достоинства Windows, вследствие чего было продано огромное количество копий системы. Это связано с тем, что аппаратные средства ОС Windows смогли обеспечить должный уровень производительности, достоинства графического интерфейса мгновенно стали очевидны для огромного числа пользователей. Windows 3.1 была выпущена главным образом для того,

чтобы покончить с проблемами, которые были выявлены в ходе широкомасштабного использования Windows 3.0.

Все вышеперечисленные версии Windows не являлись операционными системами. Они играли роль графических интерпретаторов команд (командных процессоров), которые работали на платформе операционной системы MS DOS.

История ОС Windows 9x

Значение ОС Windows на платформе 9x велико, хотя и их история не такая длинная. Первая версия ОС на платформе 9x вышла в 1995 году и имела название Windows 95. **Основная цель создания Windows 95 — разработка системы, обладающей удобным пользовательским интерфейсом**. Миссия Windows 95 состояла в том, чтобы максимально облегчить использование и обслуживание персонального компьютера, а также унифицировать разработку программного обеспечения и аппаратных средств. **Отличительной особенностью системы Windows 95 явилось преобразование ее из Windows 3.1 в полнофункциональную операционную систему.** У пользователя исчезла необходимость в использовании MS DOS. В Windows 95 была предусмотрена поддержка приложений MS DOS при помощи средств совместимости, а также для производителей аппаратных средств предоставлена возможность разрабатывать и совершенствовать свою продукцию не в обязательном строгом соответствии старой архитектуре IBM PC. В Windows 95 доступ к любым аппаратным средствам осуществлялся при помощи драйверов устройств. Пользователь, если у него есть соответствующий драйвер, легко мог доставить к системе новое устройство. Пропала необходимость в совместимости с устаревшими BIOS, если, конечно, данное устройство не должно поддерживать также и работу MS DOS.

В оболочку операционной системы Windows 95 были добавлены **новые возможности**:

– спецификация ***«Plug and Play»***, созданная совместно фирмами Microsoft, Intel, Phoenix Technologies, Compaq и др. Цель ее создания состояла в сведении к минимуму проблем, связанных с настройкой и конфигурированием аппаратных средств.

Интерфейс «Plug and Play» берет на себя все заботы по идентификации подключенного устройства и обеспечению данного устройства необходимыми аппаратными ресурсами, а также конфигурированию соответствующих драйверов устройств;

– универсальные механизмы связывания и встраивания объектов ***OLE 2*** (Object Linking and Embedding), явившиеся первым шагом к документно-ориентированной архитектуре приложений. Оболочка Windows 95 поддерживает функции OLE 2 и полный набор возможностей ***«Drag and Drop»*** (в русскоязычной версии Windows 95 эта технология называется «Перетащить и Оставить»);

– поддержка интерфейса электронной почты;

– поддержка длинных имен файлов;

– наличие большого набора средств просмотра файлов, позволяющих пользователям заглянуть в файл определенного формата без необходимости запуска приложения, которым этот файл был создан;

– поддержка приложения MS DOS. Несмотря на то, что Windows 95 с ее улучшенной оконной средой приближает их конец, все-таки поддержка приложений MS DOS в Windows 95 была заметно усовершенствована. В число новых возможностей входят действия по изменению размеров окон MS DOS, операции копирования и вставки, а также использование в приложениях MS DOS шрифтов True Type;

– наличие программного обеспечения, создающего условия для представления компьютера как полностью оборудованной машины-клиента NetWare с целью наиболее полной адаптации к сетевым средам. Кроме такой поддержки локальных вычислительных сетей, Windows 95 имела много других возможностей, относящихся к области коммуникаций, — от простейших операций, вроде набора номера телефона, до поддержки самых современных сверхпортативных компьютеров.

При этом Windows 95 стремилась наилучшим образом выполнять функции операционной системы машины-клиента и обеспечивала:

– поддержку действий машины-клиента для всех популярных сетей фирм Novell, Banyan, Microsoft и других;

– поддержку различных типов машин-клиентов, что позволяет одновременно подключать один и тот же компьютер к различным сетям, например к локальной сети Novell и к глобальной сети WAN (Wide Area Network), построенной с использованием протокола TCP/IP;

– возможность работы компьютера в качестве сервера в ***одноранговой сети*** (это компьютерные сети, основанные на равноправии участников), благодаря чему рабочие группы или небольшие фирмы избавляются от необходимости выделять специальный компьютер для выполнения функций сервера. В таких сетях отсутствуют выделенные серверы, а каждый узел является как клиентом, так и сервером. В отличие от архитектуры «клиент-сервер», такая организация позволяет сохранять работоспособность сети при любом количестве и любом сочетании доступных узлов;

– поддержку электронной почты, основанную на интерфейсе прикладного программирования сообщений MAPI (Message Application Programming Interface), позволяющем работать как с факсимильными устройствами, так и с популярными сетями электронной почты;

– возможности удаленного взаимодействия и управления, которые обеспечивают эффективный доступ к локальной сети и управление ею посредством низкоскоростных соединений. При этом Windows распознает явление «блуждающего компьютера» при поддержке синхронизации версий файлов и эффективной передаче данных по низкоскоростным каналам.

Однако ОС Windows 95 не была полностью 32-разрядной программой. Она содержала большие куски 16-разрядного ассемблерного кода и продолжала использовать файловую систему MS DOS, практически со всеми ее ограничениями.

В 1998 году вышла ОС Windows 98. **Операционная система Microsoft Windows 98 — это обновление Windows 95, расширяющее функциональные возможности домашнего компьютера:**

– простой доступ к Internet, высокая производительность системы, новые служебные программы и средства диагностики увеличивали эффективность работы. Windows 98 улучшила ка-

чество воспроизведения графики, звука и мультимедийных приложений по сравнению с Windows 95;

– подключение к Internet с общим доступом ICS (Internet Connection Sharing). ICS — это комплекс передовых технологий, дающих возможность пользователям нескольких компьютеров одновременно получать доступ в Internet через одно общее подключение;

– расширение поддержки оборудования. Усовершенствована встроенная поддержка таких стандартов, как:

- шина USB (Universal Serial Bus) — универсальная последовательная шина, предназначенная для подключения периферийных устройств;
- DVD (Digital Versatile Disc) — цифровой многоцелевой диск;
- IEEE 1394 (FireWire, i-Link) — последовательная высокоскоростная шина, предназначенная для обмена цифровой информацией между компьютером и другими электронными устройствами;
- интерфейс управления питанием и конфигурациями ACPI (Advanced Configuration and Power Interface),
- технология Digital Imaging и Microsoft WebTV для Windows, а также широкополосных сетевых подключений.

Динамическая справочная система на основе Web-технологии и 15 программ-мастеров упрощают использование компьютера. Web-совместимый интерфейс пользователя Windows 98 облегчает поиск, унифицируя представление информации в компьютере, локальной сети и Web.

Microsoft Windows 98 Second Edition (второе издание) — это обновление популярной операционной системы Windows 98, в котором использовались на тот момент времени самые современные технологии Internet, имелись средства для создания домашних сетей, поддерживалось новейшее оборудование. Во втором издании Windows 98 обеспечивалась дополнительная поддержка аппаратных средств и предлагался целый ряд новых возможностей.

Кроме того, что в ядре ОС Windows 98 содержался большой модуль 16-разрядного ассемблера кода, у этой системы были еще две серьезные проблемы:

Во-первых, хотя эта система была многозадачной, само ядро не было реентерабельным. Если процесс был занят управлением какой-либо структурой данных в ядре, а затем квант его времени заканчивался и начинал работу другой процесс, новый процесс мог получить структуру данных в противоречивом состоянии. Чтобы предотвратить возникновение подобной проблемы, большинство процессов, зайдя в ядро, первым делом получали гигантский мьютекс, покрывающий всю систему, прежде чем приступить к каким-либо действиям. Хотя такой подход и устранял потенциальную угрозу противоречивости структуры данных, он также уничтожал большую часть преимуществ многозадачности, так как процессам, чтобы войти в ядро, часто приходилось ждать, пока другой процесс ядро покинет.

Во-вторых, у каждого процесса было 4-гигабайтное адресное пространство, в котором первые 2 Гбайт полностью принадлежали процессу. Однако следующий 1 Гбайт совместно использовался (с возможностью записи) всеми процессами системы. Нижний 1 Гбайт также совместно использовался всеми процессами системы, чтобы они могли получать доступ к векторам прерывания MS DOS. В результате ошибка в одной программе могла повредить ключевые структуры данных, используемые посторонними процессами, вследствие чего эти процессы рушились. Что еще хуже, последний 1 Гбайт совместно использовался (с возможностью записи) процессами и ядром и содержал некоторые критические структуры данных. Любая программа, записав поверх этих структур какой-либо мусор (преднамеренно или нет), могла вывести из строя всю систему. Очевидно, решение, заключавшееся в том, чтобы не помещать структуры данных ядра в пространство пользователя, было неприменимо, так как старые программы, написанные для MS DOS, не смогли бы тогда работать в Windows 98.

В конце 1999 года была выпущена последняя версия операционной системы на платформе 9x — **Windows Me (Millennium Edition)**. Она была ориентирована также на домашнее использование и работу с Интернетом и мультимедийными данными.

Хотя в данной версии исправлены некоторые ошибки, а также добавлены новые функции (улучшенные возможности воспроизведения изображений, музыки и фильмов, домашняя

сеть, поддержка кабельных модемов и ADSL (Asymmetric Digital Subscriber Line — асимметричная цифровая абонентская линия), поддержка ***универсальных самонастраивающихся устройств*** UPnP (Universal Plug and Play) и др.), суть системы мало изменилась. Одна интересная новая функция состояла в возможности восстановить прежние настройки компьютера после неверной установки каких-либо параметров.

Далее фирма Microsoft полностью переориентировалась на поддержку линейки операционных систем на платформе Windows NT. Так как вышедшая через два года операционная система Windows XP, построенная на платформе NT, совместила функциональные элементы Windows 9x.

История ОС Windows NT

В 1988 году компания Microsoft решает создать операционную систему нового поколения. Возможности использования ОС MS DOS сильно ограничены (однопользовательская, 16-разрядная аппаратная архитектура, не обеспечивается защита памяти у процессов, ограничена платформой процессоров Intel). Работы по созданию операционной системы нового поколения начались в 1989 году, возглавил их Дэйв Катлер. Результаты разработки были представлены в 1993 году, операционная система получила название Windows NT 3.11. Начальный номер версии был выбран так, чтобы он соответствовал номеру версии популярной тогда Windows 3.11. Она поддерживала процессоры Intel x86, MIPS и Digital Alpha. В 1994 году выпускается версия **Windows NT 3.51** с повышенной производительностью и поддержкой микропроцессора PowerPC.

Первое значительное усовершенствование системы NT появилось в 1996 году в виде версии NT 4.0. Эта система не только обладала мощностью и надежностью современной операционной системы, но и использовала тот же самый пользовательский интерфейс, что и очень популярная в то время Windows 95. Эта совместимость облегчала пользователям переход с Windows 95 на NT, и многие пользователи так и поступили: перешли с Windows 95 на Windows NT Workstation [3].

С самого начала ОС NT разрабатывалась в расчете на переносимость системы на другие платформы, поэтому она была практически полностью написана на языке С с очень небольшими включениями на ассемблере для низкоуровневых функций, как обработка прерываний.

Следом за NT 4.0 предполагалось выпустить версию NT 5.0. Однако в 1999 году фирма Microsoft изменила ее название на Windows 2000, в основном из-за попыток найти нейтральное имя, выглядящее логическим продолжением как для пользователей Windows 98, так и пользователей NT. Таким образом, корпорация Microsoft рассчитывала иметь единую ОС, построенную на основе надежной 32-разрядной технологии, но использующую популярный пользовательский интерфейс Windows 98.

Поскольку ОС Windows 2000 представляет собой NT 5.0, она унаследовала множество свойств системы NT 4.0. Она является полностью 32-разрядной многозадачной системой с индивидуально защищенными процессами. У каждого процесса собственное 32-разрядное виртуальное адресное пространство. ОС работает в режиме ядра, тогда как процессы — в пользовательском режиме, что обеспечивает полноценную защиту. У процессов может быть один или несколько потоков, видимых для операционной системы и управляемых ею. Она удовлетворяет требованиям безопасности уровня С2 Министерства обороны США. Она обладает поддержкой симметричных многопроцессорных систем с числом процессоров от 2 до 32.

Далее рассмотрим более подробно используемые в настоящее время ОС на платформе NT.

2.1.2 Операционная система Windows 2000

ОС Windows 2000 — это не просто улучшенная версия NT 4.0 с интерфейсом Windows 98. Она содержит множество других функций, которые были ранее только в Windows 98: поддержка устройств plug-and-play, шины USB, стандарта IEE1394, IrDA (Infrared Data Association — стандарт на инфракрасную передачу данных и вывод на печать), управление питанием. Кроме то-

го, добавлен ряд новых функций, не присутствовавших ранее в других ОС фирмы Microsoft:

– служба каталогов Active Directory, позволяющая администраторам использовать групповые политики для обеспечения единообразия настройки пользовательской рабочей среды, развёртывать ПО на множестве компьютеров;

– система безопасности Kerberos (компьютерный сетевой протокол аутентификации, позволяющий отдельным личностям общаться через незащищённые сети для безопасной идентификации),

– поддержка смарт-карт;

– инструменты мониторинга системы;

– улучшенная интеграция лэптопов и настольных компьютеров;

– инфраструктура системного администрирования.

Новые свойства получила файловая система NTFS 5.0. Два пользователя могут совместно использовать один связанный файл. Как только один из них начинает запись в этот файл, автоматически создается копия этого файла. Кроме того, новая файловая система NTFS 5.0 допускает шифрование файлов.

Еще одно значительное усовершенствование заключается в интернационализации. ОС NT 4.0 поставлялась в виде отдельных версий для различных языков, так как текстовые строки были внедрены в программный код. ОС Windows 2000 состоит из единого двоичного кода, работающего во всех странах мира. Для каждой установки системы и для каждого пользователя можно выбрать язык, который будет использоваться во время работы системы. Это возможно потому, что все пункты меню, строки диалоговых окон, сообщения об ошибках и другие текстовые строки удалены из операционной системы и помещены в специальные каталоги, по одному для каждого языка. Как и предыдущие версии ОС NT, Windows 2000 использует кодировку Unicode для поддержки языков, не использующих латинский алфавит, например русского, греческого, иврита, японского.

В Windows 2000 нет MS DOS ни в каком виде. Есть интерфейс командной строки, но это новая 32-разрядная программа, включающая функциональность старой системы MS DOS, а также некоторые новые функции.

Несмотря на многочисленные свойства, способствующие переносимости системы, ОС Windows 2000 обладает меньшей переносимостью, чем NT 4.0. Она работает на двух платформах: Pentium и Intel-IA-64.

Windows 2000 поставлялась в виде нескольких уровней продукта: Professional, Server, Advanced Server, Datacenter Server. Однако различия между этими версиями незначительны, и в них используется один и тот же исполняемый двоичный код. При установке системы тип продукта записывается во внутренней базе данных (системном реестре). Во время загрузки ОС проверяет содержимое реестра, определяя версию программного продукта. Различия между ними показаны в таблице 2.1.

Таблица 2.1 — **Функциональные возможности различных версий Windows 2000**

Возможности	**Professional**	**Server**	**Advanced Server**	**Datacenter Server**
Максимальный размер ОЗУ, Гбайт	4	4	8	64
Поддержка нескольких процессоров	2	4	8	32
Максимальное число клиентов	10	Не ограничено	Не ограничено	Не ограничено
Размер кластера	0	0	2	4

Размер кластера означает способность операционной системы Windows 2000 представить для окружающего мира две или четыре отдельные машины в виде одного сервера, что часто бывает полезно, например для Web-серверов. Следует отметить, что в Windows 2000 Professional по-другому (по отношению к серверам) настраиваются параметры по умолчанию. В этой системе интерактивным процессам предоставляется преимущество перед пакетными заданиями, хотя это можно при необходимости изменить. Еще одно отличие серверных систем заключается в том, что с ними предоставляется дополнительное программное

обеспечение, а с системой Datacenter Server поставляются дополнительные средства управления большими заданиями.

Причина существования нескольких версий исключительно коммерческая. Это позволяет корпорации Microsoft получать с крупных компаний больше денег, чем с индивидуальных клиентов, за практически один и тот же программный продукт.

Формально различием в версиях управляют в нескольких местах программы всего две переменные, считываемые из реестра: ProductType и ProductSuite. В зависимости от этих значений выполняется слегка отличный код. Изменение значений этих переменных рассматривается как нарушение лицензии. Кроме того, система перехватывает любые попытки изменить их и регистрирует эти попытки нестираемым способом, так что впоследствии можно доказать факт нарушения лицензии.

Кроме основных операционных систем, корпорация Microsoft разработала несколько инструментальных программ для продвинутых пользователей: Support Tools (средства поддержки), Software Development Kit (SDK — средства разработки программных продуктов), Driver Development Kit (DDK — средства разработки драйверов) и Recourse Kit (набор ресурсов). Это большие наборы утилит для отладки и мониторинга системы. Инструментарий поддержки распространяется на компакт-диске Windows 2000, в каталоге \Support\tools. Кроме того, существует множество утилит для слежения за внутренней работой Windows 2000, разработанных другими компаниями.

2.1.3 Операционная система Windows XP

Windows XP — очередная операционная система, ориентированная на клиента, но и с функциональными возможностями серверной ОС, которая выпущена компанией Microsoft после выпуска Windows 2000 и Windows Millenium.

В прошлом у компании Microsoft было желание поставлять линейку одной ОС или, по крайней мере, линейки ОС, основанных на единственной системе кодов. Windows 2000 должна была стать системой, которая объединила бы Windows NT и Win 9x, но из-за нескольких моментов (наименее значительным из которых является неполное использование потенциальных воз-

можностей) изначальная цель Windows 2000 — слияние двух кодов — была отложена до более поздней версии. С Windows XP компания Microsoft попыталась устранить все имевшиеся ранее проблемы, которые возникали у пользователей с Windows, и это ей в какой-то мере удалось.

Windows XP Professional по сравнению с Windows 2000 обладают усовершенствованными технологиями управления конфигурацией.

Приведем основные характеристики Windows XP:

– Основу системы Windows XP составляет код Windows NT и Windows 2000, характеризуемый 32-разрядной вычислительной архитектурой и полностью защищенной моделью памяти.

– Средство проверки драйверов устройств в ОС Windows XP, созданное на основе аналогичного средства системы Windows 2000 и обеспечивающее более тщательное испытание драйверов.

– Доступность критически важных структур ядра системы только для чтения, благодаря чему драйверы и приложения не могут повредить их. Код драйверов устройств также доступен только для чтения и снабжен защитой на уровне страниц.

– Наличие механизма, позволяющего устанавливать и использовать одновременно несколько версий компонентов системы Windows.

– Возможность безопасной передачи данных через Интернет с помощью системы IP-безопасности.

– Обновленный внешний вид при сохранении ядра Windows 2000, предоставляющий возможность объединения и упрощения типичных задач; добавления новых визуальных подсказок, помогающих пользователю в работе с компьютером; возможность смены обновленного пользовательского интерфейса на классический интерфейс Windows 2000 одним нажатием кнопки администратором или пользователем системы.

– Дистанционное управление рабочим столом. Несколько пользователей могут проводить активные сеансы на одном и том же компьютере. Таким образом, даже при входе в систему других пользователей состояние сеанса Windows каждого пользователя остается неизменным, а запущенные программы попрежнему выполняются.

– При использовании помощника по поиску в Windows XP Professional имеется возможность выполнять поиск всех типов объектов, таких, как изображения, музыкальные файлы, документы, принтеры, компьютеры и люди. Можно выполнять поиск на своем компьютере, на других компьютерах (в случае наличия подключения к сети или рабочей группе), а также в Интернете. Можно выполнить поиск, воспользовавшись помощью анимированного персонажа.

– Установка программного обеспечения, отличного от операционной системы, может потребовать определенного мастерства, особенно если новое программное обеспечение может заменить важные файлы операционной системы. Такие действия способны привести к неустойчивой работе системы и программ, а возможно, и к сбою в работе операционной системы. Средство «Защита файлов Windows» используется для предотвращения замены или удаления системных файлов. Защита файлов Windows выполняется в фоновом режиме, в результате чего защищаются все файлы, установленные программой установки Windows.

– В случае возникновения проблем в работе системы можно восстановить компьютер до последнего устойчивого состояния без потери файлов личных данных (таких, как документы, содержимое папки «Избранное» для работы в Интернете, сообщения электронной почты). Программа «Восстановление системы» ведет наблюдение за изменениями в компьютере и периодически создает легко идентифицируемые точки восстановления. Эти точки восстановления позволяют вернуть систему к предыдущему состоянию. Пользователь также имеет возможность в любое время создавать именованные точки восстановления.

– При помощи Windows XP Professional и подключения к Интернету можно регистрировать системные и программные ошибки в специальной службе корпорации Майкрософт. Если при возникновении ошибки пользователь решает ее зарегистрировать, технические сведения о возникшей проблеме отправляются в специальную службу корпорации Майкрософт через Интернет. Если подобная проблема уже зарегистрирована, могут

быть доступны дополнительные сведения по этому вопросу. Сведения, получаемые корпорацией Майкрософт, используются группами разработчиков для контроля качества и не будут использованы для отслеживания пользователей с целями распространения рекламы.

– Использование технологии ClearType для отображения экранных шрифтов позволяет добиться такого же четкого отображения слов на экране компьютера, как и на бумажной странице. Использование этой технологии позволяет значительно улучшить разрешение отображения шрифтов, что обеспечивает предельную четкость при отображении существующих электронных таблиц, текстовых документов и веб-страниц. Технология ClearType применяется для плоскоэкранных мониторов, поэтому рекомендуется использовать ее для переносных компьютеров и других плоскоэкранных устройств. На мониторах настольных компьютеров шрифты ClearType могут казаться немного смазанными, если экран не является плоским.

– Хотелось бы использовать экран большего размера при работе на переносном компьютере? Технология Dualview позволяет подключить к переносному компьютеру отдельный монитор, чтобы на разных экранах просматривать разные программы. Например, на одном экране можно просматривать сообщения электронной почты, а на другом — электронную таблицу. Технология Dualview похожа на технологию использования нескольких мониторов, однако для нее необходим один видеоадаптер. Не все видеоадаптеры поддерживают технологию Dualview.

– Возможность настройки и оптимизации многочисленных функций операционной системы Windows XP, а также устранения неполадок.

Корпорацией Microsoft изначально предполагалось представить три выпуска операционной системы Windows XP:

1) **Windows XP Professional** — операционная система, предназначенная для корпоративных пользователей и обеспечивающая высокий уровень масштабируемости и надежности.

2) **Windows XP Home Edition** — эффективная платформа для работы с цифровыми мультимедийными материалами, яв-

ляющаяся наиболее удачным выбором для пользователей домашних компьютеров и любителей компьютерных игр.

3) **Windows XP 64-Bit Edition** — 64-разрядная операционная система, способная удовлетворить самых требовательных пользователей, обладающих специальной технической подготовкой.

Впоследствии Microsoft выпустила еще два выпуска:

1) **Windows XP Tablet PC Edition** — эта платформа предназначена для работы на мобильных персональных компьютерах.

2) **Windows XP Media Center Edition** — как следует из названия, данный выпуск ориентирован на организацию из персонального компьютера центра по управлению мультимедийными устройствами.

Требования к оборудованию для 32-разрядных версий Windows XP выглядят следующим образом:

– процессор с рекомендуемой тактовой частотой 300 MHZ или более (233 MHZ — требуемый минимум); рекомендуемый процессор — семейство Intel Pentium/Celeron, семейство AMD K6/Athlon/Duron или совместимые с ними.

– Рекомендуемый объем памяти 128 МБ RAM или выше (поддерживаемый минимум 64МБ; может ограничивать работу и некоторые функции).

– 1.5 GB доступного места на жестком диске.

2.1.4 Операционная система Windows 2003 Server

Windows 2003 Server, созданный на основе доказавших свою надежность продуктов семейства Windows 2000 Server, осуществляет интеграцию многофункциональной среды выполнения приложений в целях создания современных веб-служб и подготовки бизнес-решений, позволяющих значительно повысить эффективность процессов обработки информации. При использовании платформы Windows 2003 Server разработчикам предоставляются следующие преимущества:

1. Создание более эффективных приложений благодаря следующим возможностям:

– собственной поддержке веб-служб XML на основе использования стандартов, таких, как SOAP[2], WSDL[3] и UDDI[4], в результате чего расширяются возможности для обеспечения взаимодействия приложений друг с другом;

– обширному набору интегрированных служб распределенных приложений, оптимизированных по быстродействию и масштабируемости и включающих новые усовершенствования, связанные с развертыванием, управлением и обеспечением безопасности;

– встроенным средствам обеспечения надежности на основе поддержки архитектуры слабо связанных и тесно связанных компонентов;

– тесной интеграции с серверами Microsoft 2003 Enterprise Server.

2. Сокращение времени разработки благодаря следующим возможностям:

– предоставлению интегрированного набора служб;

– собственной поддержке веб-служб XML (SOAP, WSDL, UDDI);

– использованию управляемого кода и других возможностей Microsoft Visual Studio 2003, что позволяет разработчикам сократить время программирования;

– интеграции со средой разработки Visual Studio 2003;

– использованию единого языка программирования;

– возможностям использования существующего оборудования независимо от применяемого языка программирования.

---

[2] SOAP (Simple Object Access Protocol) — стандарт SOAP, описывающий протокол, предназначенный для обмена структурированной информацией в распределенных системах, таких, как Интернет.

[3] WSDL (Web Services Description Language) — язык для описания Интернет-сервисов.

[4] UDDI (Universal Description, Discovery and Integration) — универсальный метод описания, обнаружения и интеграции web-сервисов для систем электронной коммерции. Бизнес-регистр UDDI представляет собой базу данных общего пользования, в которой компании сами себя регистрируют.

Семейство Microsoft Windows 2003 Server включает четыре продукта:

1) **Windows 2003 Web Server** — продукт, представляющий собой веб-сервер, в котором наибольшее внимание уделяется улучшению функциональных возможностей. Он разрабатывался с целью предоставления компаниям многофункциональной и устойчивой платформы для обслуживания и размещения веб-приложений, обеспечивающих при этом удобство развертывания и управления. Благодаря использованию технологии Microsoft ASP2003 — одной из составляющих среды 2003 Framework — сервер Windows 2003 Web-Server предоставляет разработчикам платформу для быстрого создания и развертывания веб-служб XML и веб-приложений.

2) **Windows 2003 Standard Server** — продукт, представляющий собой надежную операционную систему для сети, обеспечивающую быструю и простую реализацию бизнес-решений. Эта гибкая серверная ОС прекрасно подходит для удовлетворения повседневных требований как крупных, так и малых организаций. В операционной системе Windows 2003 Standard Server предоставляются технологии совместного использования файлов и принтеров, безопасного подключения к Интернету, централизованного развертывания приложений для настольных компьютеров и организации эффективной совместной работы между сотрудниками, партнерами и заказчиками. В Windows 2003 Standard Server реализована поддержка до 4 Гб оперативной памяти и симметричной многопроцессорной обработки с использованием двух процессоров.

3) **Windows 2003 Enterprise Server** — продукт, предназначенный для компаний среднего и крупного размера. В нем реализованы функциональные возможности, необходимые для поддержки инфраструктуры организации, бизнес-приложений и транзакций электронной коммерции. Windows 2003 Enterprise Server — операционная система, обладающая полным набором функциональных возможностей, обеспечивающая поддержку до восьми процессоров и предоставляющая возможности корпоративного уровня, такие, как создание и поддержка кластеров, состоящих из четырех узлов, и организация оперативной памяти,

достигающей объема в 32 ГБ. Данная система доступна также для 64-разрядных вычислительных платформ.

4) **Windows 2003 Datacenter Server** — продукт, предназначенный для компаний, предъявляющих высокие требования к масштабируемости и доступности. Предоставляет надежную основу для создания критически важных технических решений, обеспечивающих поддержку баз данных, программ планирования ресурсов предприятия ERP (En-terprise Resource Planning), обработку сложных транзакций в режиме реального времени и консолидацию серверов. Это — самая эффективная и мощная серверная операционная система из всей линейки Windows 2003 Server. В ней реализована поддержка симметричной многопроцессорной обработки с использованием до 32-х процессоров. В качестве стандартных функций предоставляются службы балансировки нагрузки и создания кластеров, состоящих из восьми узлов. Операционная система Windows 2003 Datacenter Server доступна также для 64-разрядных вычислительных платформ.

Windows 2003 Server основывается на достоинствах, унаследованных от платформы Microsoft Windows NT. Продукты семейства Windows 2003 Server полноценно взаимоувязаны с серверами, принадлежащими семейству Windows 2000.

Функциональные возможности различных версий Windows 2003 Server приведены в таблице 2.2.

Таблица 2.2 — **Требования к оборудованию — различных версий Windows 2003**

Возможности	**Windows 2003 Web-сервер**	**Windows 2003 Standard Server**	**Windows 2003 Enterprise Server**	**Windows 2003 Datacenter Server**
Минимальная тактовая частота процессора	133МГц	133МГц	133МГц (Intel x86) 733МГц (Intel Itanium)	400МГц (Intel x86) 733МГц (Intel Itanium)
Рекомендуемая тактовая частота процессора	550МГц	550МГц	733МГц	733МГц

Окончание табл. 2.2

Возможности	Windows 2003 Web-сервер	Windows 2003 Standard Server	Windows 2003 Enterprise Server	Windows 2003 Datacenter Server
Минимальный объем ОЗУ	128Мб	128Мб	128Мб	512Мб
Рекомендуемый минимальный объем ОЗУ	256Мб	256Мб	256Мб	1Гб
Максимальный объем ОЗУ	2Гб	4Гб	32Гб (Intel x86) 64Гб (Intel Itanium)	64Гб (Intel x86) 128Гб (Intel Itanium)
Поддержка нескольких процессоров	1 или 2	1 или 2	До 8	Не менее 8 Не более 32
Место на диске, необходимое для установки	1,5Гб	1,5Гб	1,5Гб (Intel x86) 2,0Гб (Intel Itanium)	1,5Гб (Intel x86) 2,0Гб (Intel Itanium)

2.1.5 Операционная система Windows Vista

Windows Vista одна из последних операционных систем (на момент написания этого учебного пособия) ориентированная на клиента. Она является операционной системой, которой фирма Microsoft планирует заменить Windows XP. Windows Vista по сравнению с Windows XP претерпела ряд существенных изменений. Эти изменения начинаются уже с момента развертывания операционной системы.

Развертывание новой операционной системы Windows Vista на предприятии производится на основе образов, что делает этот процесс наиболее эффективным. Использование образов является наиболее быстрым способом развертывания операционной системы, но исторически сложилось так, что образы не являлись частью стандартного процесса установки операционной системы Windows, требуя использования дополнительного

программного обеспечения и многих часов работы по их поддержке. Чтобы помочь упростить процесс развертывания, корпорация Microsoft положила в основу установки операционной системы Windows Vista использование файл-ориентированного формата образов, называемого Windows Imaging Format (WIM); разделила Windows Vista на отдельные модули, чтобы упростить настройку и развертывание образов; сделала другие значительные усовершенствования в ядре операционной системы, относящиеся к процессу развертывания.

Windows Vista представлена в виде отдельных модулей, что упрощает ее настройку. Во время подготовки Windows Vista к установке в организации ИТ специалисты могут выбирать определенные компоненты системы, которые будут установлены на определенный ряд компьютеров. Например, языки являются отдельными компонентами, таким образом, на одну группу компьютеров может быть установлен английский язык, а на другие группы компьютеров — русский, немецкий или испанский. Драйверы и обновления также являются отдельными компонентами, что позволяет производить обновления образов по мере изменения аппаратных и программных требований.

Формат WIM-образов умеет определять тип используемого аппаратного обеспечения, что позволяет хранить всего один образ для различных аппаратных конфигураций оборудования. Формат WIM позволяет также хранить несколько образов в одном WIM-файле, упрощая управление образами и сохраняя дисковое пространство, поскольку хранит только одну физическую копию каждого файла. Например, можно хранить два образа в одном WIM-файле: первый образ может содержать операционную систему Windows Vista, а второй образ — базовые приложения. Формат WIM существенно уменьшает размер файловых образов, используя методы сжатия и хранения единичных копий.

Поддержка WIM-образов является простой задачей, поскольку драйверы, обновления и некоторые другие компоненты Windows могут быть добавлены и удалены автономно, без распаковки образа операционной системы. Windows Vista содержит инструменты для прямого редактирования образов, позволяющие менять общие и региональные настройки, применять обновления операционной системы, добавлять драйверы, устанав-

ливать обновления, добавлять и удалять приложения. Эта особенность позволяет экономить часы работы по поддержке установочных образов, исключая необходимость их распаковки для изменения конфигурации.

Кроме этого, формат WIM-образов позволяет проводить неразрушающее развертывание. Это означает, что Вам не нужно делать резервную копию информации, находящейся на томе, куда устанавливается образ, так как процедура установки образа не приводит к удалению существующего содержимого диска.

Усовершенствование системы безопасности Windows Vista отразилось на процессе развертывания. Например, настройка поддержки пользователей с «низким уровнем прав», когда пользователь, выполнивший вход в систему, не обладает правами администратора, в Windows Vista выполняется проще. Некоторые приложения не работали в Windows XP, когда пользователь не обладал правами администратора, поскольку эти приложения предполагали наличие прав на запись на диске C: и в любых разделах реестра. В Windows Vista попытки приложений записать данные в защищенную область будут незаметно для пользователя перенаправлены в другие области профиля пользователя.

Второе большое изменение заключается в том, что пользователи, не являющиеся администраторами, могут загружать драйверы. Это позволяет пользователям подключать устройства, не обращаясь с просьбами в службу поддержки.

Иногда может потребоваться использовать права администратора, но это не значит, что нужно работать с правами администратора все время. Таким образом, Windows Vista добавляет функцию управления доступом пользователя (User Access Control, UAC), что приводит к тому, что большинство пользовательских приложений (даже для администраторов) выполняется с ограниченными правами. Для приложений, требующих дополнительных прав, UAC будет выдавать запрос, требуя либо разрешения на выполнение с повышенными правами доступа, либо указания учетных данных другого пользователя, которые будут использованы вместо учетных данных текущего пользователя.

Усовершенствования также коснулись встроенного брандмауэра Windows Vista. Новый брандмауэр может управлять как входящим, так и исходящим трафиком, по-прежнему предостав-

ляя возможность полной настройки средствами групповой политики.

Наконец, функция BitLocker, обеспечивающая шифрование тома целиком, включенная в Windows Vista Enterprise и Ultimate, позволяет полностью зашифровать том, содержащий операционную систему. После этого данные этого тома могут быть прочитаны только в Windows Vista и только при указании надлежащих ключей, которые могут быть взяты из встроенного модуля Trusted Platform Module (TPM) версии 1.2, ключа USB или введены с клавиатуры.

Установка в текстовом режиме не используется. Базовый процесс, используемый для установки Windows XP, оставался неизменным со времени появления Windows NT. Эта длительная процедура состояла из начального этапа установки в текстовом режиме, в ходе которого извлекались из архива и устанавливались все файлы операционной системы, создавались все записи реестра и применялись все настройки безопасности. В Windows Vista этот этап установки в текстовом режиме не используется. Вместо этого новая программа установки выполняет установку, перенося образ Windows Vista на компьютер.

После переноса образа выполняется его настройка для данного компьютера. Настройка выполняется вместо того этапа, который назывался мини-установкой в Windows XP и Windows 2000. Ее цель осталась прежней: операционная система выбирает необходимые настройки и установки, специфические для того компьютера, на котором она была установлена.

Windows Vista (любая версия) поставляется на диске DVD в виде уже установленного, обобщенного (подготовленного утилитой sysprep) образа, готового к развертыванию на любом компьютере. Некоторые заказчики могут выполнять развертывание этого образа в исходном виде (возможно, добавив исправления или драйверы, используя описанные выше возможности обслуживания).

Файл Boot.ini не используется в Windows Vista. Вместо этого новый загрузчик, bootmgr, читает настройки загрузки из специального файла с именем BCD. Совершенно новое средство bcdedit.exe (или отдельный поставщик инструментария управления Windows, WMI) используется для изменения содержимого

файла BCD. Такая гибкость может быть полезной при восстановлении системы или выполнении работ по обслуживанию.

В Windows XP **настройки** хранились в виде различных текстовых файлов. Вместо этих текстовых файлов теперь используется файл в формате XML.

Вместо файла Unattend.txt, используемого для настройки установки Windows XP, теперь используется файл unattend.xml. Файл unattend.xml также заменяет еще три других файла:

1. Sysprep.inf, который использовался для настройки изменения образа Windows XP при развертывании на компьютере в процессе мини-установки.
2. Wimbom.ini, который использовался для настройки Windows PE.
3. Cmdlines.txt, который использовался для указания списка команд, выполняемых в процессе мини-установки.

Новые средства управления событиями. Предыдущие версии Windows имели множество недостатков при регистрации событий и трассировке. К ним относятся ограниченная масштабируемость журнала событий (ограничение общего объема всех журналов объемом доступной памяти), производительность публикации событий (которая ограничивала, например, число событий, которые могли быть опубликованы активным контроллером домена), а также ограниченная защита событий трассировки.

Windows Vista решает многие из этих проблем, предоставляя новую инфраструктуру регистрации событий и трассировки, получившую название Windows Eventing 6.0. Эта служба является расширением ETW (События трассировки Event Tracing for Windows), используемой со времени выхода Windows 2000, и заменяет службу журнала событий и программу просмотра событий. Новая служба Windows Eventing разработана специально для работы с событиями, которые сохраняются в файлах журналов для последующей проверки.

Все усовершенствования предоставлены с сохранением полной совместимости с существующими API журнала событий и ETW, таким образом, все существующие приложения будут работать без изменений.

Новое средство просмотра событий было полностью переработано, и поскольку оно работает в среде консоли управления Microsoft (MMC) версии 3.0, его внешний вид также изменился, но остался вполне узнаваемым для того, чтобы при переходе не возникло затруднений.

По-прежнему присутствуют панель дерева и список событий. Также можно воспользоваться знакомыми журналами приложений, системы и безопасности, расположенными в разделе журналов Windows. Однако к корневому узлу дерева добавлены новые узлы, а в узле журналов Windows появился новый журнал ForwardedEvents.

Самое очевидное новшество — область просмотра, расположенная ниже списка событий. Она содержит свойства просматриваемого в данный момент события. Это значит, что вам больше не нужно открывать событие двойным щелчком мыши, чтобы просмотреть его свойства, и не нужно выстраивать окна, чтобы увидеть рядом список и диалоговое окно свойств события. Разумеется, возможность просмотра свойств в диалоговом окне по двойному щелчку события мышью сохранилась. Но новое диалоговое окно не является модальным, поэтому вы можете одновременно открыть несколько диалоговых окон свойств события.

Новые представления позволяют вам просматривать все интересующие вас события, сделав несколько щелчков мышью. Можно отобрать события из одного или нескольких файлов журнала и просмотреть события по отдельным кодам событий, уровням (серьезности) или периодам времени.

Работа в корпоративных сетях. В ОС Windows Vista реализована новая версия стека TCP/IP, существенным образом улучшающая несколько наиболее важных аспектов сетевой работы и позволяющая добиться повышения производительности и пропускной способности, а также собственная архитектура Wi-Fi и интерфейсы API для проверки сетевых пакетов.

Для максимального использования сетевых возможностей необходима комплексная настройка конфигурационных параметров TCP/IP. В Windows Vista не приходится делать это вручную, так как система сама анализирует сетевые условия и автоматически оптимизирует сетевые параметры. В сетях с больши-

ми потерями данных, например в беспроводных сетях, Windows Vista способна лучше восстанавливать информацию после потери одного или нескольких пакетов. Она может динамически увеличивать или уменьшать окно TCP на прием, что позволяет использовать всю ширину канала. При передаче файлов по высокоскоростной глобальной сети с большим временем отклика или при скачивании файлов из сети Интернет пользователи несомненно заметят существенное сокращение времени передачи файлов.

Кроме того, в состав базового сетевого стека Windows Vista включена собственная архитектура беспроводного соединения (собственный интерфейс Wi-Fi). К числу его преимуществ можно отнести гибкое использование во многих моделях устройств различных торговых марок, схожие приемы работы с разными устройствами и более надежные драйверы беспроводных сетевых карт независимых поставщиков. Беспроводными сетями в ОС Windows Vista можно управлять централизованно, причем соединения по таким сетям поддерживают новейшие протоколы безопасности и позволяют пользователям работать с меньшими задержками.

В стеке TCP/IP нового поколения реализована новая архитектура сетевой защиты ***Windows Filtering Platform (WFP)*** с интерфейсами API, позволяющими независимым разработчикам программного обеспечения участвовать в процессе принятия решений о фильтрации пакетов на нескольких уровнях стека протокола TCP/IP без необходимости написания собственных приложений привилегированного режима. Эта архитектура обеспечивает поддержку таких функций сетевого экрана нового поколения, как проверка подлинности при соединении и динамическое конфигурирование сетевого экрана при использовании приложениями интерфейса ***Windows Sockets API***.

Узнать о состоянии сети, то есть проверить наличие соединения, выяснить провайдера соединения, узнать, в местной сети или в сети Интернет они находятся, пользователи теперь могут из единого центра управления сетевыми возможностями. Кроме того, они могут просматривать состояние различных сетевых служб на своих компьютерах. Виден ли компьютер в местной сети? Какие папки и принтеры открыты у них для доступа по

сети? Пользователь может создать сеть (временную или инфраструктурную беспроводную сеть, сеть VPN или домашнее широкополосное соединение) или подключиться к существующей сети любого типа.

Система Windows Vista способна самостоятельно диагностировать и разрешать многие проблемы со связью, так что пользователю не приходится обращаться в службу техподдержки. Инфраструктура диагностики сетевого соединения (***Network Diagnostics Framework***) позволяет ОС Windows Vista выявлять основные причины проблем со связью в контексте операции приложения. Например, если пользователь не может попасть на какой-либо Интернет-сайт, то данная система диагностики попытается отследить проблему по всей цепочке связи, начиная от определения наличия активного беспроводного соединения и действительного IP-адреса, вплоть до установления связи с DNS-сервером, нахождения прокси-сервера и получения ответа от требуемого веб-сервера.

В случае определения причины проблемы пользователь получает сообщение с четким описанием проблемы и способов ее разрешения. Иногда проблема устраняется простым щелчком мыши на данном сообщении. В некоторых случаях пользователю придется внести изменения в настройки, и диалоговое окно доставит пользователя в необходимое место. А в случаях, когда пользователь просто не может выполнить необходимые действия по причине недостатка знаний или отсутствия прав, в программу «обозреватель событий» записываются более полные сведения.

В ОС Windows Vista используются интерфейсы API ***Network Awareness***, вызываемые приложениями для выяснения состояния соединения и определения типа сети, к которой подключен компьютер в настоящий момент. Если ОС Windows Vista может получить сетевой доступ к контроллеру домена, то она автоматически выбирает профиль «Домен». Другие сети в эту категорию попасть не могут. Все другие сети определяются как сети общего доступа, если пользователь или приложение не укажет, что сеть частная.

Наличие интерфейса Network Awareness позволяет таким приложениям, как сетевой экран с дополнительными функциями безопасности (Windows Firewall with Advanced Security), ис-

пользовать различные настройки для сетей разного типа и переходить на эти настройки автоматически при изменении типа сети. Например, администратор может настроить сетевой экран таким образом, чтобы при подключении компьютера к сети с доменом определенные порты для программы управления рабочим столом были открыты, но автоматически закрывались при работе в общедоступных сетях.

Сетевой экран с дополнительными функциями безопасности обеспечивает новый уровень сетевой защиты в системе Windows с поддержкой фильтрации входящих и исходящих пакетов и функции повышения стойкости служб (***Windows Service Hardening***). Если сетевой экран обнаруживает, что поведение какой-либо службы Windows отклоняется от нормального поведения, описанного в сетевых правилах системы повышения стойкости служб, то он блокирует эту службу. Данный сетевой экран поддерживает и функцию разрешенного обхода (***Authenticated Bypass***), позволяющую некоторым компьютерам после проверки их подлинности службой IPsec обходить правила сетевого экрана для выполнения таких задач, как удаленное управление.

Одно из наиболее существенных изменений в сетевом экране заключается в его объединении со службой IPSec. Раньше для создания многоуровневой совокупности правил сетевой безопасности администраторам приходилось полагаться на два отдельных инструмента — сетевой экран и средство применения протокола IPsec и управления им. В Windows Vista для защиты сети от несанкционированного доступа администраторы могут создавать простые правила сетевой безопасности, объединяющие правила сетевого экрана и правила IPSec. Благодаря такому объединению можно осуществлять сквозную передачу данных по сети после установления подлинности обменивающихся сторон с обеспечением расширяемого многоуровневого доступа к доверенным сетевым ресурсам и/ или защиты конфиденциальности и целостности данных.

Администратор может логически разделить корпоративную сеть на зоны, доступ в которые может быть предоставлен любому компьютеру (в том числе с правами гостя) или только компьютерам, прошедшим аутентификацию в домене (отделение домена) (рис. 2.1).

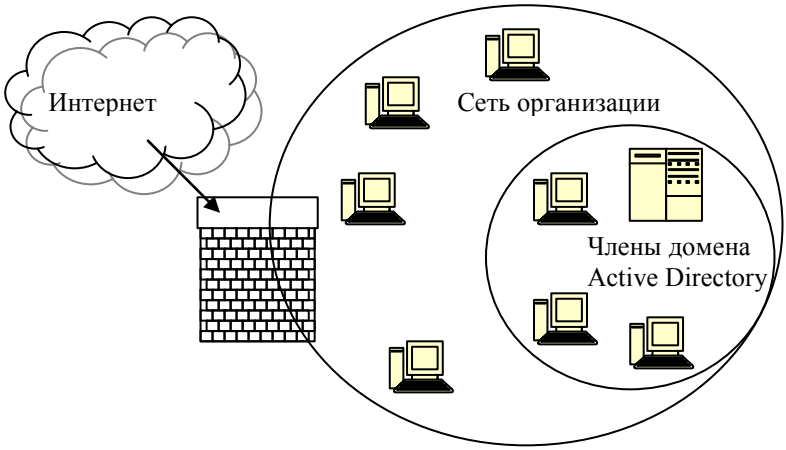

Рис. 2.1 — Демонстрация изоляции серверов и доменов в Windows Vista

Организациям масштаба предприятия часто приходится решать задачи расширения своей сети. Например, могут закончиться допустимые IP-адреса, особенно в тех случаях, когда в интересах дела каждому пользователю необходимо предоставить несколько таких сетевых устройств, как дополнительные портативные компьютеры или смартфоны. Или, другой пример, необходимо запустить дополнительные сетевые службы, например IPsec, но беспокоит чрезмерная загрузка процессора. ОС Windows Vista позволяет снять опасения по поводу расширяемости сети благодаря поддержке протокола ***IPv6*** и передаче нагрузки аппаратным устройствам.

Проблему нехватки свободных адресов IPv4 многие правительственные организации, поставщики интернет-услуг и другие организации решают путем перехода на протокол IPv6, новую версию сетевого протокола, лежащего в основе сети Интернет. ОС Windows Vista поддерживает оба протокола IPv4 и IPv6 с использованием двухуровневой архитектуры стека IP. Протокол IPv6 разрешен по умолчанию, а поддержка двухуровневого стека позволяет вам осуществить постепенный переход по предусмотренной в протоколе IPv6 технологии, направляющей трафик в формате протокола IPv6 по туннелю в частной сети IPv4 или в сети Интернет.

Windows Vista поддерживает передачу функций обработки сетевого трафика специализированным сетевым адаптерам. К числу новых возможностей передачи нагрузки можно отнести протокол IPv6 и архитектуру ***TCP Chimney***. Эти нововведения позволяют оптимизировать параметры и пропускную способность сети с максимальным использованием всех преимуществ современных высокоскоростных сетей. Благодаря применению совместимых сетевых адаптеров можно исключить узкие места в обработке сетевых пакетов, например непроизводительную загрузку процессора и ограничение пропускной способности памяти, без внесения изменений в существующие приложения или средства управления сетью.

Для устранения возможных узких мест при обработке сетевого трафика в сетевом стеке предусмотрена и поддержка распределения нагрузки на принимающей стороне, что позволяет динамически выравнивать входящий сетевой трафик с распределением нагрузки между несколькими процессорами или ядрами.

В Windows Vista **планировщик заданий** подвергся значительной переработке по сравнению с Windows XP. Изменения коснулись не только интерфейса, но и функциональности, которая значительно расширилась, позволяя осуществлять более гибкую настройку задач. Но, пожалуй, основное изменение произошло в назначении планировщика — теперь это важный системный инструмент, потому что на него возложено большое количество задач, обеспечивающих нормальную работу системы и ее оптимизацию.

Занимаясь оптимизацией, многие пользователи отключают «ненужные» системные службы с целью ускорения работы ОС. Если в Windows XP такой подход еще имел право на жизнь, то в Windows Vista он уже не оправдан. Действительно, ряд заданий выполняется при запуске системы или при входе в нее пользователя, поэтому можно подумать, что, отключив планировщик, получится начинать работу в ОС на несколько секунд быстрее. Однако не все так примитивно, да и блокируя выполнение системных задач, вы можете воспрепятствовать правильной работе ОС и ухудшить ее производительность.

Служба установщика ActiveX в Windows Vista. Элемент управления ActiveX — это часть исполняемого кода (обычно в

файл OCX, упакованный в файле CAB), устанавливаемая и запускаемая пользователем через обозреватель Internet Explorer. Веб-разработчики создают элементы управления ActiveX для добавления к веб-приложениям функций, которые невозможно осуществить с помощью стандартного кода HTML или простого сценария.

Главной функцией элементов управления ActiveX является их модель развертывания «загрузить и выполнить». Элементы управления ActiveX устанавливаются и выполняются через тег объекта HTML. В этом случае Internet Explorer загружает соответствующий пакет установки, выполняет доверенную проверку объекта и запрашивает разрешение пользователя на установку через панель информации Internet Explorer. Во время установки элемент управления регистрируется и выполняется страницей отображения. После установки элемент управления могут запустить все стандартные пользователи. Этот простой механизм распространения и выполнения предоставляет разработчикам простой способ распространения своих компонентов пользователям их веб-приложений. Проблема этого метода распространения заключается в том, что стандартные пользователи не могут устанавливать элементы управления ActiveX напрямую, поскольку для установки необходимы права администратора.

AxIS в Windows Vista позволяет корпоративным администраторам управлять элементами управления ActiveX, обеспечивая надежную защиту, поскольку пользователи выступают в качестве стандартных пользователей с параметрами файловой системы по умолчанию. AxIS предоставляет групповой политике возможности настройки доверенных источников элементов управления ActiveX и брокерский процесс для установки элементов управления из этих доверенных источников стандартными пользователями. Ключевым преимуществом является возможность сохранения неадминистративной безопасности рабочих станций пользователей с помощью централизованного административного управления. AxIS опирается на определение доверенных источников (обычно URL-адреса из Интернета или интрасети) элементов управления ActiveX администраторами отделов ИТ. Если тег объекта сообщает обозревателю Internet

Explorer о необходимости выполнения элемента управления, AxIS выполняет следующие действия:

1) Проверка установки элемента управления. Если элемент отсутствует, он должен быть установлен для его использования.

2) Проверка параметров политики AxIS на предмет того, является ли источник элемента управления доверенным. Отдельная проверка устанавливает соответствие имени узла URL-адреса, указанного в атрибуте CODEBASE тега объекта, указанному в политике списку доверенных мест.

3) Загрузка и установка элемента управления от имени пользователя.

Если имя узла URL-адреса источника отсутствует в параметре политики AxIS, будет выполнен обычный запрос контроля учетных записей, указывающий необходимость прав администратора для выполнения установки. Кроме того, AxIS зарегистрирует в журнале событий приложений событие с идентификатором 4097 от источника AxInstallService с указанием попытки установки элемента управления ActiveX, а также путь загрузки элемента управления. Данные этой записи журнала событий могут использоваться корпоративным администратором для изменения групповой политики с целью разрешения установки элемента управления AxIS при последующих посещениях веб-узла.

Технология SuperFetch нацелена снизить время ожидания, пытаясь предсказать, какие данные нужны приложению и пользователю, и предварительно разместить их в оперативной памяти. Звучит похоже на алгоритмы кэширования Windows, которые появились ещё во времена NT, но новая технология намного их превосходит. Если у обычной Windows кэш сбрасывается при перезапуске системы, то SuperFetch во время старта Vista «заселит» память популярными приложениями. Например, если вы часто работаете с Outlook, Word и Visual Studio, Vista и SuperFetch загрузят эти приложения в память при старте системы.

Первичный запуск Windows Vista при этом может замедлиться, по сравнению с Windows XP, именно потому, что с диска в память загружаются данные, которые возможно понадобятся. Вероятно, именно поэтому стандартным состоянием выключения для ноутбуков в Windows Vista стал режим «Sleep» (режим ожидания), а не Power Off (выключить), как в Windows XP. При старте из режима ожидания ноутбук израсходует меньше

энергии, чем в ситуации, когда ему пришлось бы заново с жёсткого диска прокачать гигабайты драйверов, утилит и программ.

Технология ReadyBoost предназначена для компьютеров с небольшим объёмом оперативной памяти. Если вставить USB-брелок в порт, или карту памяти во встроенный картовод ноутбука, Windows может использовать его ёмкость для дополнения оперативной памяти. Фирма Microsoft утверждает, что технология ReadyBoost совершенно безопасная: можно удалять брелок (карту) в любое время. Технология работает в паре с SuperFetch, буферизуя часто используемые данные приложений на флэш-память. Для компьютера с 2 Гбайт ОЗУ подобная система вряд ли что-то ускорит, но если ОЗУ всего 512 Мбайт или 1 Гбайт, то прирост наверняка будет, поскольку SuperFetch для успешной работы требует больше 1 Гбайт памяти. Рекомендуется подключать USB-брелок такого объёма, сколько ОЗУ установлено, или большего.

Откат. Windows Vista, как и Windows XP до неё, поддерживает точки восстановления. Это временные образы системы, которые позволяют вернуться к предыдущему состоянию, скажем, если новый драйвер оказался сбойным или приложение повредило установку Windows. Функция отката Windows Vista более мощная, она позволяет восстанавливать предыдущее состояние отдельных файлов или папок. Необходимо нажать правой клавишей мыши на документ, который нужно восстановить, и перейти на закладку «Previous Versions/Предыдущие версии».

Элементы интерфейса. При запуске Windows Vista остались привычные элементы интерфейса, например ***главное меню*** (рис. 2.2). Главное меню предлагает возможности поиска по всему компьютеру с помощью новой функции быстрого поиска, которая позволяет найти и запустить почти все приложения, что имеются на компьютере. Более того, новое главное меню упрощает доступ к установленным на компьютере приложениям.

Новые мощные и удобные ***обозреватели*** обеспечивают единообразную работу с файлами в Windows Vista (рис. 2.3). Обозреватели предоставляют больше информации и возможностей управления и упрощают работу с файлами. Просмотр фотографий и документов и даже использование новой панели управления — все эти действия характеризуются удобством и единообразием.

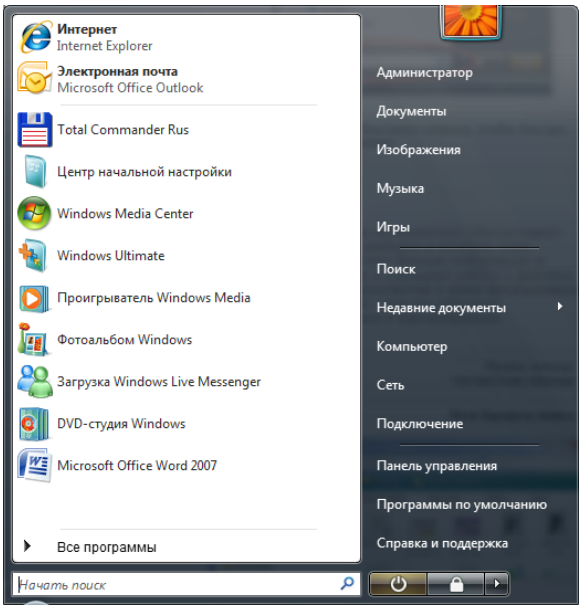

Рис. 2.2 — Пример главного меню в Windows Vista

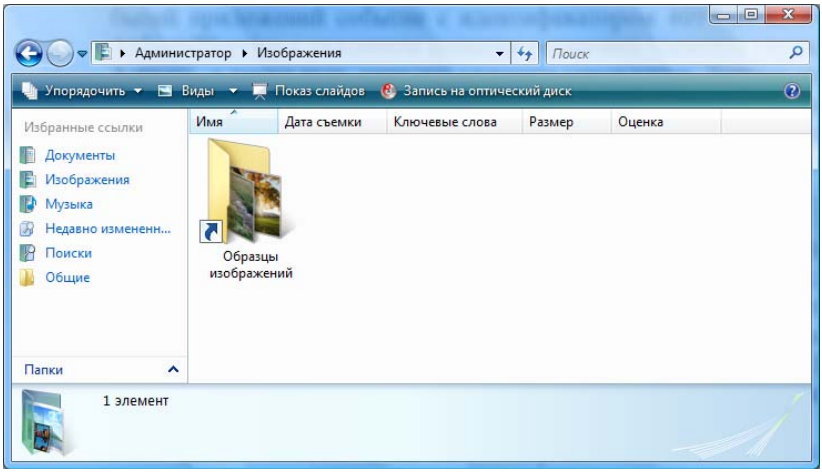

Рис. 2.3 — Пример обозревателя файлов

После установки в глаза сразу же бросается то, что Windows Vista оснащена трёхмерным пользовательским интерфейсом. Microsoft называет новый интерфейс ***AeroGlass***, он выводит 3D-объекты там, где это имеет смысл, а также обеспечивает 3D-ускорение визуальной анимации вместо 2D-ускорения Windows XP. Благодаря этому возможны и некоторые новые, ранее недоступные возможности. Например, переключение задач через трёхмерный интерфейс, при этом сами задачи не замирают, и вы в процессе переключения видите, что та или иная программа делает. Так происходит, даже если Windows Media Player показывает видео.

Файлы и папки в Vista Explorer являются трёхмерными объектами, текстурированными их содержанием. А у рамок окон Windows появилась некоторая степень прозрачности, причём под рамками изображение как бы размывается, благодаря чему текст в строке заголовка расположенных друг над другом окон ничуть не менее читаем, чем в Windows XP, где окна были непрозрачными. По сравнению с некоторыми программами для XP, прозрачность окон в Windows Vista реализована гораздо правильнее.

Если увеличить в Vista Explorer размер иконки, то можно посмотреть «внутри» содержимое папки, документ Word или любой другой тип данных, который понимает Windows. Отличие от Windows XP сразу же становится очевидным, когда замечаешь, что размер иконки можно произвольно менять, а объекты можно наслаивать.

После запуска Windows Vista над рабочим столом возникнет меню ***Welcome Center***, который весьма полезен для новичков. В нём можно посмотреть детали о компьютере, настроить и персонализировать Vista, выйти в панель управления, узнать основы работы с Windows и т.д.

Windows Vista предлагает две совершенно новые возможности работы с окнами: ***Windows Flip*** и ***Windows Flip 3D***. Функция Flip позволяет переключаться между открытыми окнами (с помощью сочетания клавиш Alt+Tab). При этом отображается активный эскиз каждого окна, а не просто общий значок и имя файла (рис. 2.4). Активные эскизы помогают быстро находить нужное окно, особенно когда открыто несколько окон одного типа. Функция Flip 3D дает возможность переключаться между

открытыми окнами, расположенными друг за другом, с помощью колеса прокрутки мыши, быстро находить и выбирать для работы нужное окно (рис. 2.5).

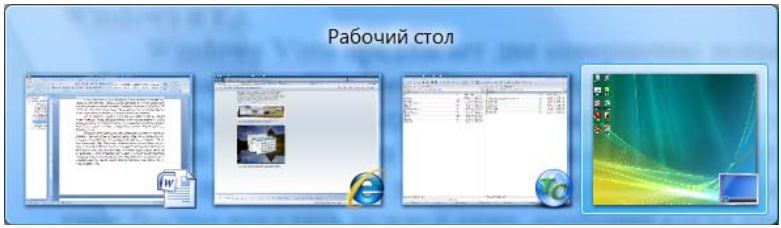

Рис. 2.4 — Пример переключения между открытыми окнами с использованием функции Flip

Интерфейс AeroGlass, наверное, является самой визуально привлекательной чертой по сравнению с Windows XP. Он требует, чтобы ваша графическая система поддерживала ряд функций 3D, причём требуется ускоритель с поддержкой DirectX 9. Другими словами: если вашей видеокарте больше двух лет, то запустить AeroGlass вы не сможете. Vista будет работать в базовом режиме без 3D.

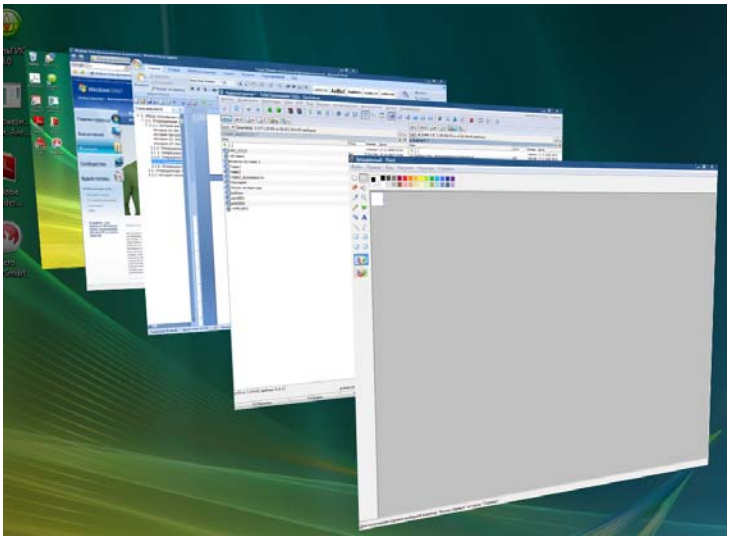

Рис. 2.5 — Пример переключения между открытыми окнами с использованием функции Flip 3D

Существует пять выпусков операционной системы Windows Vista:

1) **Vista Starter** — это наиболее доступный способ использования возможностей системы Windows Vista. Система Windows Vista Starter предназначена исключительно для развивающихся рынков и разработана для начинающих пользователей компьютеров. Она снабжена дополнительными средствами и обучающими программами, делающими ее проще в использовании.

2) **Vista Home Basic** — это выпуск Windows, предназначен для обычных домашних компьютеров. Если компьютер планируется использовать только для таких задач, как работа в Интернете, использование электронной почты или просмотр фотографий, тогда система Windows Vista Home Basic является правильным выбором. Хотя этот выпуск лишен многих преимуществ системы Windows Vista Home Premium, он позволяет сделать использование компьютера проще и безопаснее, чем при работе с Windows XP.

3) **Vista Home Premium** — это выпуск Windows для домашних настольных и переносных компьютеров. Система Windows Vista Home Premium обеспечивает производительность и развлекательные возможности, которые требуются от компьютера дома или в пути. Этот выпуск содержит Windows Media Center и облегчает работу с цифровыми фотографиями, ТВ и фильмами, а также музыкой. Кроме того, использование данного выпуска повышает безопасность и надежность компьютера.

4) **Vista Business** — этот выпуск операционной системы, разработан специально для удовлетворения потребностей малого бизнеса. Улучшенный, простой в использовании интерфейс, который позволяет легко и быстро выполнять поиск необходимых данных как на компьютере, так и в Интернете. Новые возможности обеспечения безопасности помогают контролировать и защищать ключевые сведения, которые важны для организации.

5) **Vista Ultimate** — это наиболее полная версия системы Windows Vista, сочетающая в себе производительность, безопасность и мобильность, необходимые для работы, с полным спектром функций для развлечения. Windows Vista Ultimate включает в себя все компоненты системы Windows Vista Home Premium и все компоненты системы Windows Vista Business.

6) **Vista Enterprise** — этот выпуск предназначен для больших предприятий и организаций (доступен только для участников программы Microsoft Software Assurance).

Ниже в таблице 2.3 приведены сравнительные возможности первых четырех выпусков операционной системы Windows Vista (http://www.microsoft.ru/rus/).

Таблица 2.3 — **Сравнительные возможности различных версий Windows Vista**

Возможности	Home Basic	Home Premium	Business	Ultimate
Наличие защитника Windows и брандмауэра Windows	+	+	+	+
Функции мгновенного поиска и обозревателя Windows Internet Explorer 7	+	+	+	+
Рабочий стол Windows Aero	–	+	+	+
Наличие центра мобильности Windows и поддержки планшетных ПК	–	+	+	+
Совместная работа и совместное использование документов через конференц-зал Windows	–	+	+	+
Наличие Windows Media Center	–	+	–	+
Средство от сбоев оборудования — резервное копирование	–	+	+	+

Окончание табл. 2.3

Возможности	Home Basic	Home Premium	Business	Ultimate
Средство от сбоев оборудования — восстановление Windows Complete PC	–	+	–	+
Упрощенные возможности подключения к сети в организациях с помощью сетевого центра и удаленного рабочего стола	–	–	+	+
Улучшенная защита данных от потерь с помощью шифрования диска Windows BitLocker	–	–	–	+
DVD-студии Windows	–	+	–	+
Игры: Chess Titans, Mahjong Titans и «Инкбол»	–	+	+	+
Наличие программы Windows Movie Maker в высоком разрешении	–	+	–	+

2.1.6 Операционная система Windows 2008 Server

Windows Server 2008 — это новое поколение серверной операционной системы, которое должно помочь ИТ-подразделениям максимально увеличить контроль над инфраструктурой, обеспечивая при этом новый уровень доступности и управляемости. Windows Server 2008 базируется на ключевых элемен-

тах популярной серверной платформы Windows Server 2003 и инновациях, включенных в состав второго пакета обновлений (Service Pack 2) для Windows Server 2003 и версии Windows Server 2003 R2.

Новые и расширенные компоненты операционной системы должны позволить упростить задачи управления серверами и облегчить конфигурацию и сопровождение:

Службы Internet Information Services 7.0 (IIS7) — это самый мощный созданный веб-сервер Майкрософт, оснащенный набором новых функций, которые существенно улучшают процесс разработки и развертывания веб-решений, а также управления ими. Модульная структура IIS 7.0 дает администраторам полный контроль над их веб-сервером. IIS7 обладает гибкой и расширяемой архитектурой служб, мощными возможностями диагностики и разрешения проблем. Internet Information Server (IIS) 7 и .NET Framework 3.0[5] формируют комплексную платформу для разработки приложений, которые связывают работников друг с другом и с данными, обеспечивая в результате наглядное представление, совместное использование и обработку информации.

Windows PowerShell — новая оболочка с интерфейсом командной строки, поддерживающая более 130 средств и встроенный язык программирования, позволяющая администратору автоматизировать выполнение рутинных операций по управлению системами, особенно на нескольких серверах.

---

[5] Microsoft .NET Framework — программная технология, предназначенная для создания как обычных программ, так и веб-приложений. Одной из основных идей Microsoft .NET является совместимость различных служб, написанных на разных языках. Например, служба, написанная на C++ для Microsoft .NET, может обратиться к методу класса из библиотеки, написанной на Delphi; на C# можно написать класс, наследованный от класса, написанного на Visual Basic .NET, а исключение, созданное методом, написанным на C#, может быть перехвачено и обработано в Delphi. Каждая библиотека (сборка) в .NET имеет сведения о своей версии, что позволяет устранить возможные конфликты между разными версиями сборок.

Первая версия выпущена в 2006 году и сейчас доступна для Windows XP SP2, Windows Server 2003, Windows Vista и встроена в Windows Server 2008 как опциональный компонент.

Windows PowerShell интегрирован с .NET Framework и предоставляет окружение для выполнения административных задач путём выполнения командлетов (cmdlets) — особых .NET классов, реализующих отдельные операции, сценариев, построенных из командлетов, исполняемых файлов самостоятельных приложений или экземпляров обычных классов .NET. Оболочка имеет доступ к различным хранилищам данных, таким, как файловая система или реестр, через механизм поставщиков Windows PowerShell.

Управление несколькими серверами можно автоматизировать, используя программное обеспечение Windows PowerShell, состоящее из командной строки и скриптового языка, созданного специально для автоматизации администрирования серверных ролей, таких, как веб-сервер и служба каталога Active Directory.

Windows Server Manager (Диспетчер серверов) — новый компонент операционной системы Windows Server 2008. Он представляет собой единый интерфейс, через который ИТ-администратор может выполнять все действия по установке, настройке серверных ролей и функций Windows Server 2008 и управлению ими. Диспетчер серверов объединяет в себе функции ряда компонентов Microsoft Windows Server 2003, таких, как «Управление сервером», «Мастер настройки сервера» и «Установка и удаление программ».

С помощью диспетчера серверов можно настраивать на компьютере разные функции и роли. Роль сервера в Windows Server 2008 описывает основное назначение сервера. Администратор может выделить под определенную роль весь сервер или установить несколько ролей на одном и том же компьютере. Например, на одном сервере могут быть одновременно развернуты роли сервера DHCP[6] и сервера DNS[7]. Функция сервера

[6] DHCP (Dynamic Host Configuration Protocol) — протокол динамической конфигурации узла, позволяет компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Для этого компьютер обращается к специальному серверу,

описывает не его основное назначение, а вспомогательное, или резервное. Так, после установки роли файлового сервера администратор может принять решение о развертывании функции отказоустойчивости кластеров, чтобы обеспечить большую избыточность файлового сервера.

Диспетчер серверов поддерживает графический интерфейс пользователя и средства с интерфейсом командной строки, позволяющие эффективно устанавливать, настраивать роли и функции Windows Server 2008 и управлять ими.

Обновленная консоль управления сервером Server Manager многократно упрощает администрирование сервера и его ролей.

ИТ-специалисты могут использовать инструмент **Windows Remote Shell (WinRS)** для удаленного управления серверами или сбора данных через интерфейс **Windows Management Instrumentation (WMI)**.

Типовые сценарии использования новых инструментов позволяют:

– выполнить полное конфигурирование сервера через единый интерфейс;

– безопасно добавлять и удалять серверные роли или возможности сервера, проверять статус роли сервера;

– автоматизировать управление несколькими серверами средствами PowerShell, ориентированного на выполнение административных задач, а так же таких средств управления, как WMI, ADSI, COM, Certificates и конфигурационные файлы в формате XML.

Windows Server 2008 может быть установлен одним из двух способов: полная установка или установка ядра сервера (server core). **Установка Server Core** устанавливает подмножество ис-

---

называемому сервером DHCP. Новая версия DHCP, предназначенная для использования в среде IPv6, носит название DHCPv6.

[7] DNS (Domain Name System) — система доменных имён, способная по запросу, содержащему доменное имя хоста (компьютера или другого сетевого устройства), сообщить IP-адрес или (в зависимости от запроса) другую информацию. DNS работает в сетях TCP/IP. Как частный случай, DNS может хранить и обрабатывать и обратные запросы, определения имени хоста по его IP-адресу.

полняемых файлов, необходимых для работы ядра ОС. Никакие дополнительные службы не устанавливаются и не активируются. При такой установке используется только один пользовательский интерфейс — командная строка.

Целью Server Core является снижение общего риска атак и количества необходимых для сервера обновлений. Поскольку многие обновления безопасности Windows часто включают сервисы и приложения, которые даже не используются на сервере (например, Windows Media Player или Internet Explorer), соответственно нет нужды обновлять эти компоненты. А в результате сниженного количества приложений и сервисов, запущенных на сервере, снижается потенциал атак.

Windows Server 2008 предлагает новые преимущества, основанные на использовании технологии **Hyper-V**. Основные сценарии использования технологии Hyper-V — это консолидация серверов; возможность предварительного тестирования вносимых в ИТ-инфраструктуру изменений; создание динамичных центров обработки данных и отказоустойчивых систем.

Hyper-V — это гипервизор Windows Server 2008, позволяющий вам запускать виртуальные машины на Windows Server 2008 компьютере. Hyper-V заменяет Virtual Server 2005 и является интегрированной частью операционной системы, в рамках которой решается широкий круг задач, ранее требовавший привлечения значительных ресурсов:

– решение проблемы поддержки инфраструктуры устаревших серверов;

– оптимизация нагрузки на физических серверах. Один физический сервер можно использовать для нескольких виртуальных серверов;

– решение задачи тестирования нового ПО и обкатки решений в виртуальной среде без риска для рабочей среды.

В ОС Windows Server 2008 в службы терминалов был внесен ряд улучшений и нововведений, например решение **Terminal Services RemoteApp** (TS RemoteApp). Благодаря ему пользователи в сеансе терминала могут обращаться к отдельным приложениям, а не ко всему компьютеру. Эти приложения работают на размещающем компьютере (например, на сервере), а

пользователю посылаются только их диалоговые окна, что требует меньших ресурсов на стороне клиента, снижает расходы на администрирование и развертывание и повышает защищенность данных.

Управление удаленными узлами, например филиалами предприятия, может оказаться непростой задачей. Часто там нет собственного ИТ-персонала или выделенного серверного помещения с обеспечением контроля доступа. Затраты времени и средств на развертывание и поддержку программного обеспечения и обновлений безопасности существенно увеличиваются.

Windows Server 2008 позволяет осуществлять удаленное управление почти так же эффективно, как при физическом присутствии администратора в филиале. Многие проблемы становится возможным разрешать удаленно. Новый контроллер домена только для чтения позволяет сохранять целостность и безопасность службы каталогов даже в самых незащищенных офисах компании. Встроенное физическое шифрование данных с помощью технологии **BitLocker** минимизирует риски утечки данных.

Поддержка IPv6. Windows Server 2008 — это первая версия семейства Windows Server, которая имеет родную поддержку IPv6 как часть одного IP-стека. В предыдущих версиях Windows, до выпуска Vista, поддержка IPv6 выполнялась параллельно с IPv4, и не существовало интегрированной поддержки для IPv6, включенного в службу инфраструктуры сети, такой, как DNS или DHCP. Теперь все поменялось, и IPv6 жестко вплетен в сетевой стек и сервисы инфраструктуры Windows Server 2008.

С распространением офисов филиалов в организациях стали возникать проблемы аутентификации. Во многих офисах филиалов использовался контроллер доменов, в котором пользователи могли аутентифицироваться, вместо того чтобы входить через медленные, а иногда даже не работающие сетевые соединения, которые могли вызвать сбой аутентификации и невозможность получения доступа даже к локальным ресурсам.

Windows Server 2008 решил эту проблему через создание контроллера домена с доступом только чтения **Read Only Domain Controller (RODC)**. RODC содержит доступную только для чтения копию базы данных Active Directory, а единственны-

ми данными учетных записей, хранящихся на этом контроллере, являются учетные записи сотрудников офиса филиала. Поскольку невозможно проделать никаких изменений в Active Directory с контроллера RODC, нет риска того, что некомпетентный пользовать причинит вред Active Directory. А так как в офисах филиалов, как правило, нет пользователей с правами администраторов, значительно снижается риск того, что RODC в таких офисах будут содержать учетные записи администраторов, которые могут быть подвергнуты опасности в случае кражи RODC.

Network Access Protection (NAP — защита доступа к сети). NAP позволяет вам контролировать доступ компьютеров, подключеных к сети. NAP позволяет вам создавать политики, которые определяют минимальный уровень надежности клиента, прежде чем компьютеру будет разрешен доступ подключения к сети.

NAP зависит от инфраструктуры Windows Server 2008. Для его успешной работы необходим Windows Server 2008 Network Policy Server, который хранит политики безопасности. Хостам, которые не отвечают требованиям конфигурации безопасности, запрещается доступ к сети. Однако NAP позволяет создавать карантинную сеть, к которой не соответствующие хосты могут подключаться, чтобы исправлять свои недостатки. Как только клиентское ПО NAP обнаруживает, что машина соответствует, оно отправляет информацию компонентам стороны NAP сервера, который затем информирует NAP-клиента, что ему разрешен доступ подключения к сети.

Самовосстанавливающаяся NTFS. Если в предыдущих версиях Windows операционная система обнаруживала ошибки в файловой системе тома NTFS, она отмечала том как «грязный»; исправление ошибок на томе не могло быть выполнено немедленно. С самовосстанавливающейся NTFS, вместо блокировки всего тома, блокируются только поврежденные файлы/папки, остающиеся недоступными на время исправления. Благодаря этому больше нет необходимости перезагрузки сервера для исправления ошибок файловой системы.

Интерфейс системы тот же, что и в Windows Server 2003, Windows 2000 Server и т.д. При желании пользователь может

добавить необходимые ему элементы интерфейса и сделать его абсолютно идентичным стилю Aero, который имеется в Windows Vista. Кроме того, Windows Server 2008 можно установить вообще без графического интерфейса, только действительно необходимые службы. В этом случае управление сервером осуществляется в консольном режиме.

В августе 2008 года Microsoft объявила, что Windows Server 7 будет официально называться Windows Server 2008 R2 и рассматриваться как неосновной релиз, дата выхода назначена на 2010 год. Windows Server 2008 R2 будет представлен только в 64-битной версии.

Далее в таблицах 2.4 и 2.5 приведены функциональные возможности пяти выпусков операционной системы Windows Server 2008 (http://www.microsoft.ru/rus/).

Таблица 2.4 — **Функциональные возможности различных версий Windows Server 2008. Поддержка серверных ролей**

Серверная роль	Enterprise/ Datacenter	Standard	Itanium	Web
Веб-службы	+	+	+	+
Сервер приложений	+	+	+	–
Службы печати	+	+	–	–
Hyper-V	+–	+–	–	–
Службы доменов Active Directory	+	+	–	–
Облегченная служба каталога Active Directory (ADLDS)	+	+	–	–
Службы управления правами Active Directory (ADRMS)	+	+	–	–
DHCP-сервер	+	+	–	–
DNS-сервер	+	+	–	–
Факс-сервер	+	+	–	–
Службы UDDI	+	+	–	–
Службы развертывания Windows (WDS)	+	+	–	–

Окончание табл. 2.4

Серверная роль	Enterprise/ Datacenter	Standard	Itanium	Web
Службы сертификации Active Directory (ADCS)[8]	+	+ –	–	–
Файловые службы	+	+ –	–	–
Службы сетевых политик и доступа	+	+ –	–	–
Службы терминалов	+	+ –	–	–
Службы федерации Active Directory (ADFS)[9]	+	–	–	–

Редакции Datacenter и Enterprise отличаются поддерживаемым количеством процессоров и правами на использование виртуализации.

Для пользователей, которым не нужна виртуализация, редакции Windows Server 2008 Standard, Enterprise и Datacenter доступны без технологии Windows Server Hyper-V.

Редакция Datacenter также поддерживает горячую замену и добавление процессоров.

Минимальная скорость процессора: 1 ГГц.

Минимальное пространство на диске: 8 Гбайт.

Минимальная память: 512 Мбайт.

Более подробную информацию о возможностях и характеристиках операционных систем Windows и другого программ-

[8] **Службы сертификатов Active Directory в Windows Server 2008** — это настраиваемые службы для создания и управления сертификатами открытого ключа, которые применяются в программных системах безопасности, использующих технологии открытых ключей. Организации могут использовать службы сертификатов Active Directory для повышения безопасности за счет связывания идентификационных данных пользователя, устройства или службы с соответствующим закрытым ключом. Службы сертификатов Active Directory также позволяют управлять подачей заявок на сертификаты и отзывом сертификатов в различных масштабируемых средах.

[9] Веб-агент **службы федерации Active Directory (ADFS)** — компонент службы федерации Active Directory. Он используется для приема маркеров безопасности и принятия решения о предоставлении или запрете доступа пользователя к веб-приложению.

ного обеспечения фирмы Microsoft можно получить на Интернет-сайте корпорации Microsoft — www.microsoft.com — англоязычный ресурс, www.microsoft.com/rus — русскоязычный ресурс.

Таблица 2.5 — **Функциональные возможности различных версий Windows Server 2008. Технические возможности**

Технические возможности	**Windows Server 2008 Datacenter**	**Windows Server 2008 Standard**	**Windows Server 2008 Enterprise**	**Windows Server 2008 Itanium IA-64**	**Windows Web Server 2008**
Число поддерживаемых процессоров	До 32 на платформе x86 До 64 на платформе x64	До 4	До 8	До 64	До 4
Количество поддерживаемой памяти	До 64 Гбайт на платформе x86 До 2 Тбайт на платформе x64	До 4 Гбайт на платформе x86 До 32 Гбайт на платформе x64	До 64 Гбайт на платформе x86 До 2 Тбайт на платформе x64	До 2 Тбайт	До 4 Гбайт на платформе x86 До 32 Гбайт на платформе x64
Кластеризация	До 16 узлов	Нет	До 16 узлов	До 8 узлов	Нет
Права на использование виртуальных машин (ВМ)	Не ограничены	1 ВМ	До 4 ВМ	Не ограничены	Не поддерживается

2.2 Операционные системы семейства Unix

2.2.1 История разработки систем UNIX

В настоящее время существует множество клонов и версий системы UNIX включая AIX, BSD, 1BSD, HP-UX, Linux, MINIX, OSF/1, SCO UNIX, SYSTEMV, Solaris, XENIX и др., в том числе их подверсии [3,4]. Фундаментальные принципы и системные вызовы для этих систем во многом совпадают. Сходными являются и общие стратегии реализации, алгоритмы и структуры данных, хотя имеются некоторые различия.

История UNIX началась в 1969 году и связана с появлением первой системы с разделением времени CTSS (Compatible Time Sharing System — совместимая система разделения времени), разработанной в Массачусетском технологическом институте (МТИ). После шумного успеха этой системы в научных кругах исследователи МТИ объединили усилия с лабораторией Bell Labs и корпорацией General Electric и начали разработку системы MULTICS (Multiplexed Information and Computer Service — мультиплексорная информационная и вычислительная служба).

Вскоре лаборатория Bell Labs вышла из проекта, и один из инженеров этой лаборатории, Кен Томпсон, решил сам написать на ассемблере усеченный вариант MULTICS для компьютера PDP-7. Впоследствии еще один инженер Bell Labs, Брайен Керниган, как-то в шутку назвал эту систему UNICS (Uniplexed Information and Computer Service — примитивная информационная и вычислительная служба). Позже при том же произношении название системы превратилось в UNIX.

Далее к работе над этой системой присоединился Денис Ритчи, а затем весь его отдел. Во-первых, UNIX была перенесена с PDP-7 на более современные компьютеры PDP-11/20, а затем PDP-11/45 и PDP-11/70 (ОП до 2 Мбайт, 16-разрядные, с аппаратной защитой памяти). Второе усовершенствование касалось переписывания системы на языке высокого уровня, который разработал Томпсон и назвал языком B. Этот язык представлял собой упрощенную форму языка BCPL, который, в свою очередь, был упрощенным языком CPL. Эта попытка оказалась

неудачной из-за слабости языка B, в первую очередь из-за отсутствия в нем структур данных.

Тогда Д. Ритчи разработал следующий язык, являющийся преемником языка B, который, естественно, получил название C, и написал для него прекрасный компилятор. Вместе K. Томпсон и D. Ритчи переписали UNIX на языке C, который оказался как раз тем языком, и был нужен в то время, и который сохраняет лидирующие позиции в области системного программирования до сих пор.

В 1974 году D. Ритчи и K. Томпсон опубликовали ставшую важной вехой в истории компьютеров статью об ОС UNIX. За эту работу им позднее Ассоциацией по вычислительной технике ACM была присуждена престижная премия Тьюринга. Публикация этой статьи привела к тому, что многие университеты встали в очередь в лабораторию Bell Labs за копией системы UNIX. По случайному стечению обстоятельств машина PDP-11 использовалась на факультетах кибернетики практически каждого университета, а операционные системы, которые поставлялись с этим компьютером, считались неудачными.

И операционная система UNIX быстро заполнила имеющийся вакуум, не в последнюю очередь благодаря тому, что поставлялась с полным комплектом исходных текстов.

Версия UNIX содержала всего 8200 строк на языке C и 900 строк на ассемблере. В результате новые идеи и усовершенствования системы распространялись с огромной скоростью. Первая переносимая версия UNIX (версия 7) состояла из 18800 строк на зыке C и 2110 ассемблерных строк (1976 г.).

К середине 80-х годов ОС UNIX широко применялась на мини-компьютерах и инженерных рабочих станциях. В 1984 г. компания AT&T выпустила на рынок первый коммерческий вариант системы UNIX — System III, а затем System V. Вплоть до 1993 г. AT&T занималась продвижением на рынке UNIX System V, а потом продала свой бизнес, связанный с UNIX, корпорации Novell, которая, в свою очередь, в 1995 году перепродала его компании Santa Cruz Operation.

Большое влияние на развитие ОС UNIX оказал Калифорнийский университет в Беркли, который приобрел UNIX Version 6 практически с момента ее выхода. При финансовой поддержке

агентства ARPA Министерства обороны США университет разработал и выпустил улучшенную версию UNIX для мини-ЭВМ PDP-11, названную 1BSD (First Berkley Software Distribution), затем 2BSD, после чего последовали 3BSD и 4BSD уже для 32-разрядных ЭВМ VAX. В 4-й версии появилась поддержка сетей, в результате чего используемый в 4BSD протокол TCP/IP стал стандартом де-факто в мире UNIX, а затем в Интернете.

Таким образом, к концу 80-х годов широкое распространение получили две в чем-то несовместимые версии системы UNIX: 4.3 BSD и System V Release 3. Для примирения двух вариантов UNIX был разработан проект POSIX (Portable Operating System — переносимая операционная система). Буквы IX в конце добавлены, чтобы имя проекта выглядело юниксообразно. В рамках этого проекта был разработан стандарт 1003.1 (а затем 1003.2). Идея стандарта POSIX заключалась в том, чтобы производитель программного обеспечения при написании программы использовал только процедуры, описанные в стандарте 1003.1, таким образом гарантируя, что эта программа будет работать на любой версии UNIX, поддерживающей данный стандарт. Достигается это тем, что за точку отсчета объединения множеств всех свойств System V и BSD комитет IEEE взял пересечение множеств, т.е. то общее, что есть как в System V, так и в BSD.

Стандарт 1003.1 описывает только системные вызовы. Принят также ряд документов, стандартизирующих потоки, утилиты, сетевые ПО и многие другие особенности UNIX. Кроме того, язык С также был стандартиирзован Национальным институтом стандартизации США ANSI и Международной организацией стандартизации ISO.

К сожалению, после успешного принятия стандарта, объединившего System V и BSD, в мире UNIX вновь произошел раскол. Группа производителей (IBM, DEC, Hewlett-Packard) организовала консорциум OSF (Open Software Foundation), чтобы создать систему, удовлетворяющую всем стандартам IEEE, но с множеством других свойств. Среди этих свойств — оконная система (X11), графический интерфейс пользователя (MOTIF), распределенные вычисления (DCE — Distributed Computing Environment — среда распределенных вычислений), распределенное управление

(DME — Distributed Management Environment — среда распределенного управления), а также многое другое.

В ответ корпорация AT&T создала собственный консорциум UI (UNIX International), чтобы заниматься тем же. Версия UI системы UNIX основывалась на System V. Однако рынок лучше принял System V, кроме того, популярной осталась линия Университета Berkley — BSD. Многие фирмы на основе этих версий разрабатывают свои системы, например Solaris корпорации Sun и др.

На рис. 2.6 представлена хронологическая последовательность создания важнейших версий операционной системы UNIX и производных от нее систем [4]. Классической UNIX принято считать седьмую версию ОС, которая является исходной точкой двух основных ветвей развития данной архитектуры: System V и BSD.

Широкое распространение UNIX породило проблему несовместимости его многочисленных версий. Очевидно, что для пользователя весьма неприятен тот факт, что пакет, купленный для одной версии UNIX, отказывается работать на другой версии UNIX. Периодически делались и делаются попытки стандартизации UNIX, но они пока имели ограниченный успех. Процесс сближения различных версий UNIX и их расхождения носит циклический характер. Перед лицом новой угрозы со стороны какой-либо другой операционной системы различные производители UNIX-версий сближают свои продукты, но затем конкурентная борьба вынуждает их делать оригинальные улучшения и версии снова расходятся. В этом процессе есть и положительная сторона — появление новых идей и средств, улучшающих как UNIX, так и многие другие операционные системы, перенявшие у него за долгие годы его существования много полезного.

Наибольшее распространение получили две весьма несовместимые линии версий UNIX: линия AT&T — UNIX System V и линия университета Berkeley — BSD. Многие фирмы на основе этих версий разработали и поддерживают свои версии UNIX: SunOS и Solaris фирмы Sun Microsystems, UX фирмы Hewlett-Packard, XENIX фирмы Microsoft, AIX фирмы IBM, UnixWare фирмы Novell (проданный теперь компании SCO) и т.д. [2].

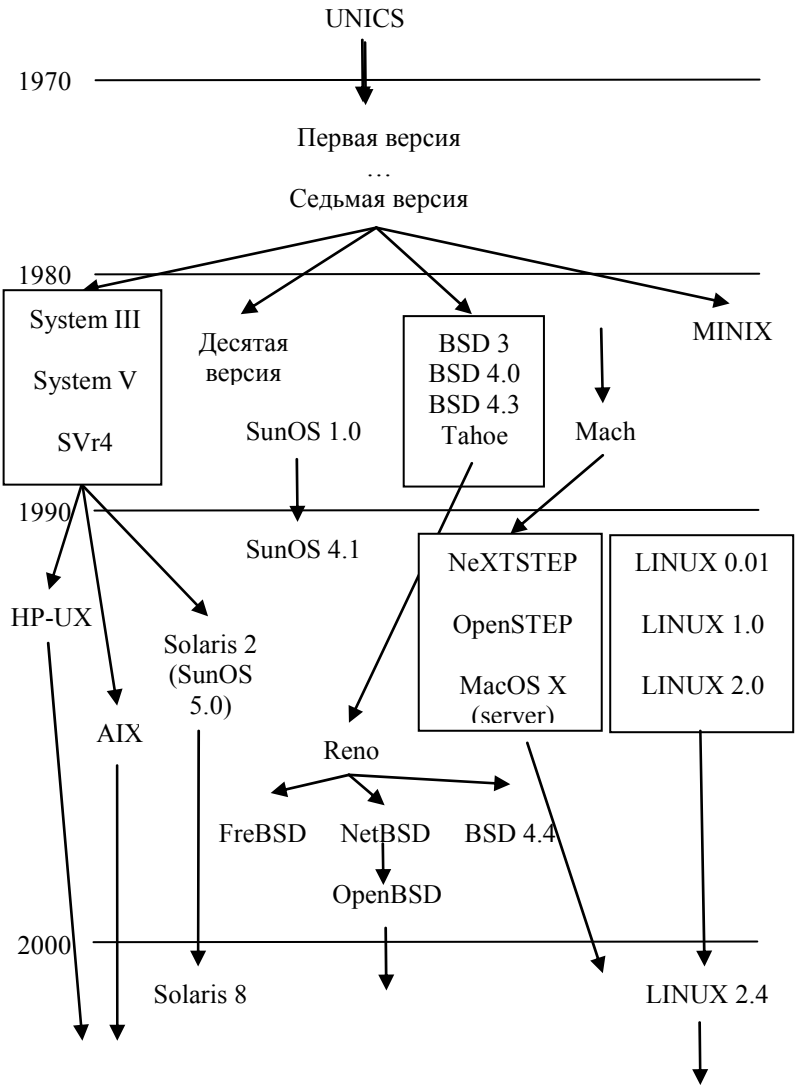

Рис. 2.6 — Хронология создания версий ОС UNIX

Наибольшее влияние на унификацию версий UNIX оказали такие стандарты, как SVID фирмы AT&T, POSIX, созданный

под эгидой IEEE, и XPG4 консорциума X/Open. В этих стандартах сформулированы требования к интерфейсу между приложениями и ОС, что дает возможность приложениям успешно работать под управлением различных версий UNIX.

Независимо от версии, общими для UNIX чертами являются:

– многопользовательский режим со средствами защиты данных от несанкционированного доступа,

– реализация мультипрограммной обработки в режиме разделения времени, основанная на использовании алгоритмов вытесняющей многозадачности (preemptive multitasking),

– использование механизмов виртуальной памяти и свопинга для повышения уровня мультипрограммирования,

– унификация операций ввода-вывода на основе расширенного использования понятия «файл»,

– иерархическая файловая система, образующая единое дерево каталогов независимо от количества физических устройств, используемых для размещения файлов,

– переносимость системы за счет написания ее основной части на языке С,

– разнообразные средства взаимодействия процессов, в том числе и через сеть,

– кэширование диска для уменьшения среднего времени доступа к файлам.

2.2.2 Примеры различных версий Unix

Версия **UNIX System V Release 4** (SVR4) — это незаконченная коммерческая версия операционной системы, т.к. в ее кодах отсутствуют многие системные утилиты, необходимые для успешной эксплуатации ОС, например утилиты администрирования или менеджер графического интерфейса. Версия SVR4 является скорее стандартной реализацией кода ядра, вобравшей в себя наиболее популярные и эффективные решения из различных версий ядра UNIX, такие, как виртуальная файловая система VFS, отображаемые в память файлы и т.п. Код SVR4 (частично доработанный) лег в основу многих современных коммерческих версий UNIX, таких, как HP-UX, Solaris, AIX и т.д.

UnixWare представляет собой полную реализацию наиболее современной версии системы UNIX для Intel-совместимых платформ — UNIX System V Release 4.2 (SVR4.2). Система сочетает высокую производительность, удобный графический интерфейс и возможности гибкой интеграции с сетями NetWare. Реализованная в ядре поддержка протокола IPX предоставляет пользователям UnixWare прозрачный доступ к сетевым ресурсам NetWare. DOS-клиенты сети получают при этом терминальный доступ к приложениям на сервере UnixWare и возможность коллективного использования файлов, хранящихся на сервере NetWare. Система выпускается в двух вариантах: UnixWare Personal Edition — для работы в качестве клиента и одноранговогo сервера на 2 соединения, UnixWare Application Server — для построения мощного многопользовательского сервера приложений.

Версия UNIX SVR4.2 была создана фирмой UNIX System Laboratories (USL) в 1992 году как развитие версии UNIX System V Release 4. Для совместимости этой версии с наиболее популярными в секторе локальных сетей операционными системами Novell NetWare было создано совместное предприятие USL и Novell Univel, которое разработало и выпустило на рынок операционную систему UnixWare.

Дополнительные свойства UnixWare по сравнению с UNIX System V Release 4:

Уменьшение требований к оперативной памяти и повышение производительности ядра. Одной из важнейших особенностей UNIX SVR4.2 является возможность эффективно работать на ЭВМ с процессором 386SX и 6 МБ оперативной памяти. Эта возможность появилась в результате работы, направленной на уменьшение размера и увеличение скорости важнейших программных компонентов системы, включая ядро и средства графики. Была проделана работа по улучшению программ загрузчика системы и закрытия системы, а также и драйверов устройств SCSI.

Изменения в структуре ОС и повышение производительности снизили минимальные требования к оперативной памяти на 30 %. Преимущества UNIX SVR4.2 по требованиям к объему оперативной памяти еще более заметны по сравнению с системами с аналогичными возможностями. Так, для работы ПО

Solaris фирмы SUN требуется минимум 12 МБ памяти, причем для нормальной работы SUN рекомендует использовать 16 МБ ОЗУ.

В ОС UNIX SVR4.2 производительность при нормальной загрузке, при «грязной» загрузке после неаккуратного закрытия, а также при закрытии системы значительно увеличилась по сравнению с предыдущими версиями. В частности, время закрытия системы сократилось на 58 % (с 38 до 17 секунд) на типичной аппаратной конфигурации ЭВМ. Загрузка системы при нормальных условиях эквивалентна физическому включению машины после аккуратного закрытия. Время нормальной загрузки сократилось на 48 % (с 65 до 38 секунд). При «грязной» загрузке эти времена составляют соответственно 140 и 40 секунд (71 %).

Отказоустойчивая файловая система Veritas. В дополнение к стандартным файловым системам (BFS, UFS, S5) UnixWare поддерживает: CD-ROM File System (CDFS), NetWare UNIX Client File System (NUCFS) и Veritas Fault Resilient File System. Система Veritas, основанная на транзакционном механизме операций с файловой системой, обеспечивает не только улучшенную производительность, но и высокую устойчивость к отказам системы.

Переносимость приложений. Унифицированная программная среда UnixWare обеспечивает поддержку широкого спектра приложений различных систем UNIX, включая SCO, ISC UNIX System V R3, SCO XENIX и BSD UNIX. Совместимость приложений обеспечивается строгим соблюдением промышленных стандартов UNIX System V Application Binary Interface (ABI), System V Interface Definition (SVID), iBSC2 и др.

Графический интерфейс. Стандарт X-Window, на основе которого построен мощный и удобный графический пользовательский интерфейс (GUI) UnixWare, в сочетании с сетевыми возможностями системы позволяет эффективно использовать перспективные архитектуры типа «клиент-сервер». Графическая среда Desktop Manager позволяет выбирать одну из двух стандартных систем графического интерфейса (OSF/Motif или OPEN LOOK) и обеспечивает при работе с графическими объектами на экране доступ к приложениям, большинству системных про-

грамм и развитой системе подсказок. Предусмотрена также возможность непосредственного программирования функций Desktop Manager.

Поддержка национальных алфавитов. В UnixWare предусмотрен широкий набор средств «интернационализации», включающий поддержку различных раскладок клавиатуры, наборов символов и языков пользовательского интерфейса.

Масштабируемые шрифты. В комплект поставки UnixWare входит система Adobe Type Manager, обеспечивающая доступ к тысячам существующих масштабируемых шрифтов формата Type 1.

Средства управления доступом. В дополнение к средствам идентификации пользователей по имени и паролю UnixWare имеет развитые средства управления доступом к ресурсам системы. Имеется возможность полного протоколирования работы системы, включая регистрацию выполняемых команд и доступа к информации.

Интеграция с NetWare. UnixWare обеспечивает полную интеграцию с сетью NetWare, благодаря которой рабочие станции UnixWare имеют доступ к ресурсам (файловая система, принтеры, почта) сети NetWare, как и другие ее клиенты, а остальные пользователи локальной сети получают также терминальный доступ к серверу приложений UnixWare. При этом как на уровне клиента, так и на уровне сервера приложений операционная система UnixWare использует традиционный для NetWare сетевой протокол IPX. Пользователям UnixWare в локальной сети NetWare предоставляются следующие виды поддержки:

– прозрачный доступ к файлам, принтерам и электронной почте;

– протоколы IPX, SPX и NCP, встроенные в ядро операционной системы;

– поддержка протоколов SAP (Service Advertising Protocol) и RIP (Routing Information Protocol);

– графический пользовательский интерфейс с функциями NetWare.

Поддержка мультипроцессирования. Начиная с версии 2.0, UnixWare поддерживает симметричное мультипроцессиро-

вание (SMP). Оба варианта UnixWare 2.01 (Application Server и Personal Edition) поддерживают в базовой поставке 2 симметричных процессора Intel. UnixWare Application Server может поддерживать (за счет добавления модулей поддержки дополнительных процессоров) до 8 процессоров Intel. UnixWare 2.01 является многонитевой операционной системой.

Solaris 2.x — это операционная система компании Sun, базирующаяся на UNIX System V Release 4. Она включает:

– базовую операционную систему SunOS 5.x и систему сетевой поддержки ONC (Open Network Computing);

– оконную систему Open Windows версии 3.x (построенную на базе X11R5) с интерфейсом в стандарте OPEN LOOK;

– набор вспомогательных утилит (диспетчер файлов, почту, печать, календарь и другие) DeskSet версии 3.x.

Компания Sun доработала исходный код UNIX System V Release 4 в соответствии со своими потребностями. Новая ОС Solaris 2.x имеет несколько основных отличий от базовой операционной системы:

– реализована многонитевость;

– поддерживается симметричная многопроцессорная обработка;

– предусмотрен режим реального времени: допускаются прерывания процессов в системной фазе, что обеспечивает гарантированное время ответа на запросы.

Сетевая среда Solaris 2.x включает в себя известную и уже ставшую стандартом сетевую файловую систему NFS, глобальную справочную службу и средства разработки распределенных приложений. Сегодня Solaris стал одной из самых распространенных версий UNIX. Эта ОС работает на платформах SPARC, Intel x86.

Осенью 1994 года компания Sun Microsystems выпустила версию операционной системы Solaris для платформ SPARC и Intel x86 —Solaris 2.4. Эта версия появилась в результате тщательного и долгого тестирования предыдущих версий. Новое качество выражается не только в устранении всех замеченных в ходе тестирования недостатков, но и в более высокой производительности, чем у Solaris 2.x. В частности, увеличена средняя производительность при работе с СУБД за счет реализации

асинхронных операций ввода-вывода в ядре, а не в библиотеках. Производительность файлового сервера NFS увеличилась в результате более эффективного использования механизма многонитевой обработки. Кроме того, гораздо быстрее стали работать протоколы TCP/IP и программы, реализующие пользовательский интерфейс. Важным свойством Solaris 2.4 является переносимость — программы, написанные для SPARC, могут выполняться на x86 и наоборот.

SCO UNIX System V/386. Варианты ОС UNIX, производимые компанией SCO[10] и предназначенные исключительно для использования на Intel-платформах, до сих пор базируются на лицензированных исходных текстах System V 3.2. Однако SCO довела свои продукты до уровня полной совместимости со всеми основными стандартами (в тех позициях, для которых существуют стандарты). Консерватизм компании объясняется прежде всего тем, что ее реализация ОС UNIX включает наибольшее количество драйверов внешних устройств и поэтому может быть установлена практически на любой Intel-платформе. Естественно, при переходе на другой вариант опорных исходных текстов ядра системы могла бы потребоваться массовая переделка драйверов. Тем не менее SCO имеет соглашение с французской компанией Chorus Systems о разработке новой версии SCO UNIX, базирующейся на микроядре Chorus и предназначенной для использования в системах реального времени.

В настоящее время компания SCO приобрела у Novell ОС UnixWare и работает над версией UNIX, совмещающей особенности SCO UNIX и UnixWare в рамках одной системы.

ОС BSD/OS 2.0 BSDi/386 — недорогая коммерческая операционная система. Хорошо поддерживается. Поддерживает

[10] The SCO Group — американская компания, держатель патента на некоторую часть исходного кода UNIX. Ранее была известна как Caldera Systems и занималась разработкой и внедрением своего дистрибутива GNU/Linux. В 2000 компания купила у Santa Cruz Operation права на операционные системы UnixWare и OpenServer, тем самым получив авторские права на исходный код этих систем, восходящих к оригинальному UNIX. Новое название компания получила в 2002 и на данный момент, несмотря на совпадающие аббревиатуры, не имеет никакого отношения к Santa Cruz Operations.

бинарную совместимость с SCO Unix. 386bsd BSD 4.3 для платформы Intel. NetBSD производная от 386bsd.

ОС FreeBSD 4.10 — операционная система, обладающая самой качественной сетью. Дабы не связываться с USL, в нем полностью заново переписаны куски ядра, на которых стоял копирайт AT&T[11].

ОС Linux 2.6.13 — самая популярная операционная система среди бесплатных ОС Unix. Число инсталляций оценочно — десятки миллионов. Непрерывное совершенствование силами сотен добровольцев довело его до уровня довольно надежной, быстрой, качественной и удобной системы, пригодной для работы как в качестве графической рабочей станции, так и интернет-сервера. Поддерживает больше всех приложений и аппаратного обеспечения.

Поддерживает спецификации ***iBCS (intel Binary Compatibility Specification)*** — спецификация по совместимости бинарных файлов между разными операционными системами на процессоре Intel x86, и потому может выполнять коммерческие приложения для SCO, в частности Oracle и Informix.

В Linux реализованы клиент и сервер Netware и Samba. Эмулятор MS Windows WABI — работоспособен.

Существует ряд операционных систем UNIX, поставляемых производителями компьютеров. Перечень наиболее значимых приведен в таблице 2.6.

Выбирая платформу, мы автоматически получаем с ней «ее собственный» Unix. С точки зрения удобства и цельности администраторского управления, самые качественные из них: HP-UX 11.11 и AIX. А самые распространенные: Linux 2.6 и FreeBSD 4.10 — среди некоммерческих Solaris 2.10 и HP-UX/11 — среди коммерческих.

---

[11] USL «засудила» BSD за использование фрагментов кода с копирайтом AT&T, что в немалой степени способствовало закрытию проекта BSD. Предпоследний из крупнейших поставщиков коммерческих BSD-подобных систем — Sun/SunOs перешли на линию SVR4. Торжественно и официально объявлено, что SVR4 является единственным наследником, объединившим лучшие решения, пришедшие из Unix V и BSD. Последним из могикан остается DEC: его OSF/1 для Alpha AXP — немного искалеченный, но все же достаточно близкий к корням BSD.

Таблица 2.6 — **Перечень ОС UNIX, поставляемых производителями компьютеров**

Машина	Операционная система	Оригинальная ОС
HP 9000	HP-UX 11	BSD 4.2 & SV 3.2 & SVR4.0
Sun 1,2,10,...	SunOS 4.1.2	BSD 4.3
Sun 10,20,1000	Solaris 2.10	SVR4.0
IBM RS/6000	AIX	SV 3.2
DEC Alpha AXP	OSF/1	BSD 4.3
DEC Alpha AXP	Tru64	OSF/1
Sequent	Dynix	SV 4.0
Motorola 922	SVR4/88	SVR4.0
Беста-88	Bestix	SV 3.1
DEC	Ultrix	BSD 4.2
SGCS Silicon Graphic	IRIX 6.0	

2.2.3 Программное обеспечение X Window

Программное обеспечение X Window состоит из двух компонент: одна умеет рисовать на графическом мониторе и обслуживать аппаратное обеспечение. Она называется X-сервер. Вторая программа — ее клиент X Window, рисовать физически не умеет, но она знает, что именно надо рисовать, и умеет командовать. Команды типа: «нарисовать прямоугольник», «провести линию», «открыть окно», «вывести символ в заданном фонте», «опросить координаты мыши» и т.п. передаются X-серверу, а тот их исполняет — рисует. Формат и спецификации этих команд опубликованы, стандартизованы, и широко известны. Все вместе они называются «Протокол X Window». Программное обеспечение X Window использует следующие основные понятия:

– **X-сервер.** Программа, которая написана специально под конкретное физическое устройство (имеется в виду — монитор, графконтроллер, мышь и клавиатура), умеет на нем рисовать и понимать команды рисования по протоколу X Window System.

– **X-клиент.** Прикладная программа, обеспечивающая графический интерфейс с пользователем. Команды для рисования на экране передает X серверу по протоколу X Window System.

– **X-терминал.** «Ящик» (монитор, графконтроллер, мышь, клавиатура, ну и, естественно, процессор и оперативная память), на котором запущена программа X-сервер. X-терминал может быть специализированным, запускать только X-сервер и ничего кроме, а может быть обыкновенной графической Unix-рабочей станцией, на которой X-сервер выполняется как одна из многих прикладных задач. Можно использовать персональный компьютер под ОС MS-DOS, на котором запущена DOS-программа, реализующая X-сервер.

– **Xlib** — библиотека функций языка C, реализующих протокол X Window System. С помощью этой библиотеки можно писать графические программы — X-клиенты.

При использовании программного обеспечения X Window пользователи получают универсальный GAPI (Graphical Application Programming Interface) — средство программирования графических приложений. Пользовательские графические программы при этом полностью отвязаны от аппаратного обеспечения, от конкретного графического контроллера.

Протокол X Window может функционировать в сети, например по TCP/IP или по DEC-net. Поэтому X-сервер может работать на одной машине, а X-клиент — на другой. То есть изображение рисуется на одной машине, а программа, которая ее обеспечивает, работает на другой.

X-сервер способен обслуживать сразу много клиентов, причем всех — одновременно. На графическом экране может быть открыто сразу много окон — каждое окно порождается его собственным X-клиентом. Клиенты эти могут быть запущены и на машине, к которой подключена графическая подсистема, и на удаленных машинах, соединенных с ней по сети.

Можно писать «X-овые» программы, используя библиотеку Xlib, хотя это весьма тяжело, так как уровень библиотеки Xlib невысок. В помощь программистам было создано несколько toolkit (наборов инструментов) — библиотек более высокого

уровня, в которых реализованы различные визуальные компоненты — widgets[12].

Известны следующие библиотеки toolkits:

Xaw — Anthena Widgets. Черно-белый, плоский и весьма не богатый набор. На нем реализованы такие программы, как xterm, xedit, xman.

Xview — набор библиотек и объектов, использованных в реализации набора пользовательских утилит Sun-овской версии X Window — «Open Windows». Весьма симпатичная библиотека, использующая круглые кнопки, «шприцы-иголки», очень хорошо продуманный интерфейс пользователя, в том числе активно используемая трехкнопочная мышь. Исходные тексты библиотек xview открыты, предоставляются бесплатно. Однако набор DeskSet — 15 пользовательских утилит, входящих в OpenWindows, сделанный на библиотеке xview, SunSoft готов предоставить только за плату.

Motif — платная библиотека поставляется организацией OSF[13]. Исходные тексты библиотек недоступны либо чрезмерно дороги. Приложения, реализованные с использованием библиотеки Motif, напоминают приложения MS Windows.

В борьбе BSD и Unix V победил, как известно, System V, а в войне круглых и квадратных кнопок выиграл Motif.

2.3 Операционная система OS/2

2.3.1 История разработки системы OS/2

История разработки операционной системы OS/2 всегда противопоставляется как борьба между Microsoft Windows и

---

[12] Widget — непереводимый термин X Window. Склеен из двух слов — Window (окно) и Gadget (приспособление), является графическим объектом с привязанными к нему свойствами и реакциями на действия пользователя.

[13] OSF (The Open Software Foundation) — фонд открытого программного обеспечения. Независимая некоммерческая научно-исследовательская организация, созданная Bull, DEC, IBM, HP, Hitachi, Philips, Siemens и рядом других фирм, для разработки открытых стандартов и UNIX-продуктов, базирующихся на стандартах X/Open и POSIX и независимых от корпорации AT&T.

IBM OS/2. Некоторые специалисты в области информационных технологий предполагают, что Microsoft имеет преимущество. Но не все согласны с такой точкой зрения. OS/2 v.2.0 была первой доступной и работающей 32-битной операционной системой для персональных компьютеров. И она первой начала очередной круг состязаний — версия OS/2 Warp, предназначенная для клиентских машин сетей клиент-сервер и одноранговых сетей, появилась на рынке раньше Windows 95, позиционированной аналогичным образом. OS/2 Warp была также первой системой, включившей набор средств поддержки Internet, а также средств объектной ориентации [2].

Операционная система OS/2 начиналась как совместная разработка IBM и Microsoft. Изначально она была задумана как замена ОС MS DOS. Уже тогда было ясно, что MS DOS с ее ограничениями по памяти и по возможностям файловой системы не может воспользоваться вычислительной мощностью появляющихся компьютеров. OS/2 была хорошо продуманной системой. Она должна была поддерживать вытесняющую многозадачность, виртуальную память, графический пользовательский интерфейс, виртуальную машину для выполнения DOS-приложений. Фактически она выходила за пределы простой многозадачности с ее концепцией, названной многонитевостью.

Первые версии OS/2 не оказали значительного влияния на рынок. **Версия OS/2 1.0**, выпущенная в 1987 году, содержала большинство технических свойств, необходимых для многозадачной ОС. Однако у нее не было менеджера графического представления PM (presentation manager), а также отсутствовали драйверы для многих популярных принтеров и других устройств.

Версия OS/2 1.1, появившаяся в 1989 году, делала возможным использование графических приложений в нескольких окнах. Однако в этой версии у менеджера графического представления не хватало многих свойств, которые присущи развитому графическому интерфейсу, кроме того, по-прежнему отсутствовали многие драйверы принтеров.

Выпущенная в 1990 году **версия 1.2** имела улучшенный PM, хотя он и не следовал общепринятым концепциям графиче-

ского интерфейса. Появились драйверы для большинства принтеров и других периферийных устройств.

Однако дискредитация OS/2 уже произошла. Версия 1.2 не была существенно лучше предыдущих версий и все еще предъявляла значительные требования к аппаратуре. К этому времени многие пользователи решили перейти на новую платформу Windows 3.0 или подождать, пока не появится что-нибудь принципиально лучшее. Продажи OS/2 по-прежнему были вялыми, и рынок не интересовался ею. Это объяснялось наличием у OS/2 ряда существенных недостатков:

– Виртуальная машина DOS, которая должна была бы обладать способностью выполнять немодифицированные приложения DOS, с самого начала имела технические изъяны. Эта виртуальная машина была разработана на базе виртуальных возможностей процессора i286, который позволял выделять сегмент памяти в 640 Кб для отдельного DOS-приложения. Однако процессор i286 в этом виртуальнои режиме работал слишком медленно, поэтому виртуальная DOS-машина была реализована на основе реального режима процессора. При этом требовался перезапуск процессора для переключения между реальным и защищенным режимами. Хотя эта операция и выполнялась очень быстро и незаметно для пользователя, она была сложной и вносила путаницу.

– Microsoft и IBM не смогли в полной мере реализовать концепцию виртуальной обработки в режиме i8086 — в этом режиме DOS-приложения, которые непосредственно читали или писали в аппаратные порты, переставали работать. В связи с этим не могли использоваться и популярные сетевые операционные системы на базе DOS.

– Память в этом режиме использовалась нерационально — если пользователь конфигурировал OS/2 с возможностью DOS-совместимости, то 640 КБ памяти всегда выделялись для этих целей и не могли использоваться для задач OS/2.

– Еще одним недостатком было отсутствие возможности обмена данными между DOS- и OS/2-приложениями.

– В каждый момент времени могло выполняться только одно DOS-приложение, и это приложение не могло использовать расширенную память.

В результате для пользователей OS/2 многие популярные DOS-приложения оказались недоступными, а те, что были доступны, не могли вообще взаимодействовать со средой OS/2. Время показало, что для пользователей это обстоятельство оказалось весьма важным, так как многие отказались от покупки OS/2, оставаясь с проверенной, хотя и не очень совершенной DOS.

Семейство 32-разрядных ОС для IBM-совместимых компьютеров начало свою историю с появления первой **OS/2 версии 2.0** в 1992 году.

В конце 1994 года IBM выпустила **третью главную версию OS/2**, которую назвала OS/2 Warp 3[14].

OS/2 Warp имеет хорошо продуманный объектно-ориентированный интерфейс с применением техники drag-and-drop при выполнении операций копирования, удаления, печати, а также некоторых других. Перечни свойств объектов легко доступны в меню, вызываемых щелчком правой клавиши мыши. Имеется специальная панель для размещения часто используемых документов или прикладных программ.

В состав OS/2 Warp входит набор утилит BonusPack, который содержит IBM Works — интегрированный программный пакет начального уровня и Internet Access Kit — самый полный набор средств для сети Internet из всех средств, поставляемых в составе операционных систем, Web Browser и почта Internet Mail. В публикациях встречаются утверждения, что он более совершенен, чем набор для доступа к Internet, реализованный в Windows 95. В феврале 1995 года IBM начала продавать пакет OS/2 Warp 3 Full Pack, который содержит библиотеки Win-OS/2. Эти библиотеки дают возможность выполнять Windows-программы, не приобретая лицензионных копий Microsoft Windows.

Одним из часто критикуемых недостатков OS/2 Warp являлось то, что она не поддерживает 32-битные приложения Windows (точнее, она поддерживает API Win32s, но не поддерживает полный API Windows NT, который называется Win32 и который почти полностью поддерживает Windows 95).

---

[14] Warp — основа (пер. с англ.)

В OS/2 Warp ощущался недостаток сетевых функциональных возможностей. Положение немного изменилось, когда летом 1995 года IBM начала продавать следующую версию OS/2 — Warp Connect, которая содержала важнейшие драйверы и утилиты. В число новых средств входили редиректоры для операционных систем NetWare 3.x и 4.1 и OS/2 LAN Server. Версия OS/2 Warp Connect работал с протоколами IPX, NetBIOS и TCP/IP. Кроме того, Warp Connect предоставлял давно ожидаемые в OS/2 средства одноранговой сетевой связи. Согласно сообщению фирмы IBM в эту версию входило большое число собственных драйверов, которые могли работать более чем с 70 % существующих адаптеров Ethernet и более чем с 90 % адаптеров Token Ring. То же самое программное обеспечение дает возможность клиенту Warp Connect подключаться к серверу LAN Server 4.0 [2].

В отличие от Windows 95 ОС Warp Connect не содержала средств, поддерживающих удаленный доступ через коммутируемые телефонные сети.

Что касается почтовых услуг, то IBM выбрала для Warp Connect пакет Lotus Notes Express, а не свой собственный Ultimedia Mail/2. Notes Express позволяет соединиться с любым сервером Notes.

Как и другие версии Warp, Warp Connect поставлялась в двух версиях: одна без Windows-библиотек, другая, подобно Full Pack, с библиотеками Win-OS/2.

Четвертая версия ОС OS/2 практически представляет собой OS/2 Warp 3.0 с несколько улучшенными параметрами для DOS-задач и обновленными элементами объектно-ориентированного интерфейса. Для операционной системы OS/2 Warp 4.0 характерны:

– вытесняющая многозадачность (preemptive multitasking) и поддержка DOS- и Windows-приложений;

– интуитивно понятный и действительно удобный объектный пользовательский интерфейс;

– поддержка стандарта открытого объектного документооборота OpenDoc;

– поддержка стандарта OpenGL;

– поддержка и встроенная разработка на языке Java;

– поддержка шрифтов True Type (TTF);

– управление голосом без предварительной подготовки (технология Voice Type);

– полная поддержка глобальных сетей Интернет и технологии Интранет, доступ в CompuServe (американская почтовая служба);

– средства построения одноранговых сетей и клиентские части для IBM LAN Server, Windows, Lantastic, Novell Netware 4.1, в том числе поддержка службы каталогов;

– наличие системы удаленного доступа через модемные соединения;

– Mobile File System для поддержки мобильных пользователей;

– стандарт авторасознавания аппаратных устройств Plug-and-Play;

– набор офисных приложений (базы данных, электронные таблицы, текстовый процессор, генератор отчетов, деловая графика, встроенная система приема/передачи факсимильных сообщений, информационный менеджер);

– полная MultiMedia-поддержка, включающая систему работы с видеокамерой, расширенную систему помощи Warp-Guide.

Однако наиболее заманчивы не перечисленные из рекламного буклета возможности системы, а удобная и надежная среда при работе с базами данных, возможность работы в сетях, организованной как клиентское рабочее место при взаимодействии с большими системами.

OS/2 Warp предлагает единый интерфейс для программирования прикладных программ (API), совместимый с рядом операционных систем, что позволяет снизить стоимость разработок. Все версии OS/2 и LAN Server, включая версии OS/2 Warp и OS/2 Warp Server 4.5, совместимы по восходящей линии, что позволяет экономить средства, необходимые для поддержания уже существующих прикладных программ.

Чрезвычайно важным для пользователей является тот факт, что компания IBM для всех версий своей ОС регулярно выпускает пакеты обновления (FixPak). Эти пакеты исправляют обнаруженные ошибки, а также вносят новые функции.

Очень полезным как для управления приложениями, так и для создания несложных собственных программ является наличие системы программирования на языке высокого уровня ***Rexx***, который иногда называют языком процедур. Можно сказать, что это встроенный командный язык, служащий для тех же целей, что и язык для пакетных (batch) файлов в среде DOS, но он обладает несравнимо большими возможностями. Это язык высокого уровня с нетипизированными переменными. Язык легко расширяем, любая программа OS/2 может добавлять в него новые функции. Помимо встроенного интерпретатора с языка Rexx, имеется система программирования Visual Rexx. Есть и объектно-ориентированная версия языка Rexx с соответствующим интерпретатором.

Наиболее сильное впечатление, которое можно получить при работе в OS/2, оставляет объектно-ориентированный графический пользовательский интерфейс, а особой популярностью у программистов эта система пользовалась вследствие достаточно хорошей организации виртуальных машин и высокого быстродействия при выполнении обычных DOS-приложений.

2.3.2 Особенности архитектуры и интерфейса OS/2 Warp

В OS/2 имеется несколько видов виртуальных машин для выполнения прикладных программ. **Собственные 32- и 16-разрядные программы OS/2 выполняются на отдельных виртуальных машинах в режиме вытесняющей многозадачности и взаимодействуют между собой с помощью средств DDE[15] OS/2.** Прикладные программы DOS и Win16 могут запускаться на отдельных виртуальных машинах в многозадачном режиме. При этом они поддерживают полноценные связи DDE и OLE 2.0 друг с другом и связи DDE с 32-разрядными программами OS/2. Кроме того, при желании можно запустить несколько программ Win16 на общей виртуальной машине Win16, где они работают

[15] DDE (Dynamic Data Exchange) — универсальные механизмы динамического обмена данными. Используются разработчиками в качестве средства интеграции компонентов ПО.

в режиме невытесняющей многозадачности, как это реализовано в Windows 3.x.

Разнообразные сервисные функции API OS/2, в том числе модель системных объектов SOM (System Object Model), обеспечиваются с помощью системных динамических библиотек DLL, к которым можно обращаться без требующих затрат времени переходов между кольцами защиты. Ядро OS/2 предоставляет многие базовые сервисные функции API, обеспечивает поддержку файловой системы, управление памятью и имеет диспетчер аппаратных прерываний. **В ядре виртуальных DOS-машин (VDM-ядре) осуществляется эмуляция DOS и процессора 8086, а также управление VDM.** Драйверы виртуальных устройств обеспечивают уровень аппаратной абстракции. Драйверы физических устройств напрямую взаимодействуют с аппаратурой.

Модуль реализации механизмов виртуальной памяти в ядре OS/2 поддерживает большие, постраничные, разбросанные адресные пространства, составленные из объектов памяти. **Каждый объект памяти управляется так называемым «пейджером» — задачей вне ядра, обеспечивающей резервное хранение страниц объекта памяти.** Адресные пространства управляются отображением или размещением объектов памяти внутри них. Ядро управляет защитой памяти и ее распределением на основе объектов памяти абстрактным образом вне зависимости от каких-либо конкретных аппаратных средств трансляции процессорных адресов. В частности, ядро интенсивно использует режим копирования при записи для придания программам способности делить объекты памяти без копирования большого числа страниц, когда новое адресное пространство получает доступ к объекту памяти. Новые копии страниц создаются только тогда, когда программа в одном из адресных пространств обновляет их. Когда ядро принимает страничный сбой в объекте памяти и не имеет страницы памяти в наличии, или когда оно должно удалить страницы из памяти по требованию других программ, работающих в машине, оно с помощью механизма межпроцессного взаимодействия уведомляет пейджер об объекте памяти, в котором произошел сбой. Теперь дело пейджера сервера приложений определить, каким образом предоставить или

сохранить данные. Это позволяет системе устанавливать различную семантику для объектов памяти, основываясь на потребностях программ, которые их используют.

Ядро управляет средами исполнения для программ, обеспечивающих выполнение множественных заданий и потоков. Каждое задание имеет свое собственное адресное пространство, или отображение. Оно назначает объекты памяти, которые задание отобразило на диапазон адресов внутри адресного пространства. Задание также является блоком размещения ресурсов и защиты, при этом заданиям придаются возможности и права доступа к средствам межпроцессного взаимодействия системы. Для поддержки параллельного исполнения с другой программой в пределах одного адресного пространства ядро отделяет среду исполнения от действительно идущего потока инструкций. **Потоки вычислений, включая процессорные ресурсы, необходимые для их поддержки, называются потоками.** Таким образом, программа может быть загружена в задание и может быть исполнена в нескольких различных местах в коде в одно и то же время на мультипроцессоре или параллельной машине. Это приводит к повышению быстродействия приложения.

Система межпроцессного взаимодействия обеспечивает базовый механизм, позволяющий потокам работать в различных заданиях для связи друг с другом. Система межпроцессного взаимодействия поддерживает надежную доставку сообщений на порты. **Порты представляют собой защищенные каналы между заданиями.** Каждому заданию, использующему порт, приписывается набор прав на этот порт. Права могут быть различными для разных заданий. Только одно задание может получать данные по какому-либо порту, хотя любой поток внутри задания может выполнять операцию приема. Одно или более заданий могут иметь права передачи в порт. Ядро позволяет заданиям применять систему межпроцессного взаимодействия на передачу друг другу прав на порт. Вместо того чтобы копировать данные, сообщение содержит указатель на них, он называется указателем на данные вне линии. Когда ядро передает сообщение от передатчика к приемнику, оно заставляет передаваемую память появиться в адресном пространстве приемника

и, как вариант, исчезнуть из адресного пространства передатчика. Ядро само структурировано как задание с потоками, и большинство системных сервисов реализованы как механизмы межпроцессного взаимодействия к ядру, а не как прямые системные вызовы.

В OS/2 Warp в качестве стандартной графической оболочки используется среда WPS (Workplace Shell), организованная более логично и удобно, чем известный Windows-интерфейс. **Оболочка Workplace Shell основана на мощной системно-объектной модели SOM IBM-технологии,** специально разработанной для решения таких проблем, как жесткая привязка объектов к их клиентам и необходимость использования одного и того же языка программирования. Объекты Workplace Shell работают в среде SOM, доступ в которую можно реализовать почти на всех языках программирования, предусматривающих внешние процедуры, в том числе и на Rexx.

В отличие от GUI Windows, в которой ярлыки объектов никак не связаны между собой, в WPS объекты, имеющие аналогичные ярлыки (shadow в терминологии WPS), просто имеют дополнительные свойства быть многократно отображенными почти как самостоятельные объекты. Можно сделать несколько shadow-значков с уже существующего shadow-значка или объекта. При этом любые shadow-значки могут быть перемещены в любое место, и их связи с основным объектом не теряются. Аналогично и в GUI Windows. Но в WPS можно переместить основной объект, и его shadow-значки тоже изменят свои параметры, тогда как в GUI Windows произойдет разрушение связей, поскольку связи являются односторонними.

Про SOM можно сказать, что это не связанная ни с одним конкретным языком объектно-ориентированная технология для создания, хранения и использования двоичных библиотек классов. Ключевые слова здесь «двоичные» и «не связанная ни с одним конкретным языком». Хотя теперь многие считают OS/2 технологией прошлого, модель SOM на самом деле представляет собой одну из наиболее интересных разработок в области компьютерной индустрии даже на сегодняшний день. Объектно-ориентированное программирование (ООП) заслужило безоговорочное признание в качестве основной парадигмы, однако его

применению в коммерческом программном обеспечении препятствуют отсутствие в языках ООП средств для обращения к библиотекам классов, подготовленным на других языках, и необходимость поставлять с библиотеками классов исходные тексты. Многим независимым разработчикам библиотек классов приходится продавать заказчикам исходные тексты, поскольку разные компиляторы по-разному отображают объекты. Настоящий потенциал SOM заключается в ее совместимости практически с любой платформой и любым языком программирования. **SOM соответствует спецификации CORBA** (Common Object Request Broker Architecture) — архитектуре посредника стандартного объектного запроса, которая определяет стандарт условий взаимодействия между прикладными программами в неоднородной сети.

Интересно отметить тот факт, что существует довольно много альтернативных оболочек для OS/2, начиная с FileBar, примитивной, но зато отлично работающей на компьютерах с 4 Мбайт памяти, и кончая мощной Object Desktop, которая значительно улучшает внешний вид экрана OS/2 и делает работу с системой более удобной.

Помимо оболочек, улучшающих интерфейс OS/2, имеется также ряд программ, расширяющих ее функциональность. Это, прежде всего, **Xfree86 для OS/2** — полноценная система X Window, которая может использоваться как X-терминал при работе в сети с UNIX-машинами, а также для запуска программ, перенесенных из UNIX в OS/2. К сожалению, таких программ немного, однако большое количество UNIX-программ поставляется вместе с исходными кодами, которые, как правило, практически не нужно изменять для перекомпиляции под Xfree86/OS2.

2.3.3 Серверная операционная система OS/2 Warp 4.5

Серверная операционная система компании IBM, выпущенная в 1999 году, носит название OS/2 WarpServer for e-Business, что подчеркивает ее основное назначение, но поскольку в процессе ее создания она носила кодовое название «Аврора» (Aurora), фактически все ее так теперь и называют. К ней на данный момент времени выпущены два пакета обновлений

Aurora Convenience Pack: ACP1 (версия Warp 4.51) и ACP2 (версия Warp 4.52).

Как известно, предыдущие версии системы OS/2 могли предоставить программисту только 512 Мбайт виртуального адресного пространства для 32-битовых задач. В свое время это было очень много. Однако сегодня, хотя и крайне редко, еще встречаются задачи, требующие столь большого объема оперативной памяти. Некоторые считают серьезным недостатком ограничение в 512 Мбайт. Поэтому в последней версии системы это ограничение снято, и теперь объем виртуальной памяти может достигать 3 Гбайт (напомним, что в Windows NT 4.0 объем виртуального адресного пространства для задач пользователя составляет 2 Гбайт).

В этой системе разработчики постарались все прежние остатки старого 16-битового кода, который еще оставался в предыдущих версиях системы, заменить на полностью 32-битовые реализации, что повышает скорость работы системы. Прежде всего, сделана поддержка 32-битовых драйверов инсталлируемых файловых систем (IFS), ибо в предыдущих системах работа с ними велась через трансляцию вызовов 32bit–>–l6bit–>32bit. В то же время для обеспечения совместимости со старым программным обеспечением, кроме 32-битового, используется и 16-битовый API.

Для повышения надежности файловой подсистемы создана **новая журналирующая файловая система JFS** (Journaling File System). JFS введена для удовлетворения потребности в более живучей файловой системе для OS/2 Warp. JFS имеет большую безопасность в структурах данных благодаря технике, разработанной для СУБД. Работа с файловой системой происходит в режиме транзакций с ведением журнала транзакций. В случае системных сбоев имеется возможность обработки журнала транзакций, позволяющая производить занесение или сброс каких-либо изменений, произошедших во время системного сбоя. Эта система также повышает скорость восстановления файловой системы после сбоя. Сохраняя целостность файловой системы, эта система управления файлами не гарантирует восстановление пользовательских данных. Следует отметить, что файловая система JFS обеспечивает самую высокую скорость работы с фай-

лами из всех известных систем, созданных для ПК, что очень важно для серверной ОС.

Для работы с дисками создан специальный менеджер дисков — LVM (Logical Volume Manager). Все устанавливаемые файловые системы содержатся в LVM. LVM осуществляет определение имен дисков для программ, которые этого требуют. Это позволяет избирательно назначить любую букву любому разделу. И даже больше — ОС не будет сама использовать имена дисков. LVM в совокупности с JFS позволяет объединять несколько томов и далее несколько физических дисков в один большой логический том.

Семейство операционных систем OS/2 Warp, созданных фирмой IBM, является одним из самых лучших ОС для ПК по очень большому числу параметров. Эти операционные системы появились раньше своих основных конкурентных систем, но тем не менее они не смогли стать самыми распространенными. Основная причина сложившейся ситуации заключается в отсутствии широкой рекламы и системы продвижения этого продукта на рынок, хотя качество операционной системы было достаточно высоким [1].

Во-первых, компания IBM не сочла необходимым продвигать свою ОС на рынок программного обеспечения, ориентированный на конечного пользователя, а решила продолжить свою практику работы исключительно с корпоративными клиентами. Рынок корпоративного ПО оказался существенно уже для ПК, чем рынок ПО для конечного пользователя, ибо компьютеры типа IBM PC, прежде всего, являются персональными.

Во-вторых, основные доходы компания IBM получала не от продажи системного ПО для ПК, а за счет продаж дорогостоящих серверов и другого оборудования. Доходы от продажи своей ОС не представлялись руководству компании IBM значимыми [1]. Для успеха на рынке ОС для ПК необходимо было обеспечить всестороннюю поддержку своей системы соответствующей учебной литературой, широкой рекламой, заинтересовать разработчиков программного обеспечения. Этого не произошло, и сегодня уже практически мало кто знает о системах OS/2. В то же время следует отметить, что фирмы, которые в свое время освоили эту систему и создали для нее соответст-

вующее ПО, до сих пор не переходят на ныне чрезвычайно популярные ОС Windows NT, поскольку последние требуют существенно больше системных ресурсов и при этом функционируют медленнее.

В 2006 году компания IBM закончила поддержку ОС OS/2.

2.3.4 Эпоха eComStation

В 1999 году американская компания Serenity Systems и голландская компания Mensys BV объявляют о том, что ими достигнуто партнерское соглашение с IBM о выпуске новой клиентской версии OS/2 Warp 4.5.

Компания Serenity Systems договорилась с IBM о новом OEM OS/2 клиенте, основанном на компонентах из OS/2 Warp 4 (Merlin), OS/2 Warp Server for e-Business и технологии WiseManager Managed Client от TouchVoice. Будет 2 версии — eComStation Entry (начального уровня) и Professional (профессиональная).

Обе версии должны были включать поддержку мобильного пользователя. Система должна быть совместима с существующими приложениями для OS/2.

Клиентская версия полностью изменила свое название — новый клиент называется eComStation (сокращенно eCS). Необходимость изменить название была обусловлена (как сказали в IBM) причинами технического, юридического и коммерческого характера. Название новой системы eComStation состоит из слов: «e-Commerce Station» (система для электронной коммерции).

Важным моментом является то, что теперь операционная система была переориентирована с корпоративных пользователей на домашних и небольшие компании. С этого момента изменился характер работы разработчиков — теперь они работают в прямом контакте с пользователями.

В 2001 году началась продажа eComStation 1.0. По сути, это новая модификация OS/2, в которой изменен интерфейс за счет включения в базовую систему приложения Styler/2, а также улучшен процесс инсталляции. Добавлено новое средство распределенной инсталляции ПО под названием WiseMachine.

Выпущено две версии: eComStation Entry (начального уровня) и eComStation Professional (профессиональная). В числе прочих в состав eCS вошли: журналируемая файловая система JFS, поддержка симметричных многопроцессорных систем (до 64 процессоров), программы из бонус-пакета OS/2 Warp 4 (в том числе программа голосового управления и диктовки VoiceType), два офисных пакета (Lotus SmartSuite и StarOffice). eComStation совместима со всеми существующими приложениями для OS/2, DOS, Windows 3.1 полностью.

Летом 2002 году была выпущена русская версия ***eComStation 1.05.*** Текущая версия eComStation/Rus 1.2.5 выпущена в 2005 году. В этой версии команда разработчиков eComStation заметно упростила процедуру установки системы и улучшила поддержку оборудования.

Основные новшества в базовой версии системы:

– Обновленный пользовательский интерфейс eWorkPlace, который повышает эффективность работы пользователей и разработчиков.

– Средства для работы в сети Интернет:

- Обновленный веб-браузер. Система оснащена браузерами Mozilla 1.7 и IBM Web Browser 2.03. Почтовый клиент, входящий в состав браузера, имеет встроенный спам-фильтр и позволяет работать как с электронной почтой, так и с новостными группами.
- Innotek Web Pack для OS/2 расширяет возможности веб-браузеров IBM Web Browser и Mozilla/Firefox:
  - The Innotek OS/2 Kit for Java обеспечивает поддержку в OS/2 самых последних версий виртуальной машины Sun Microsystems Java.
  - Механизм The Innotek Font Engine может использоваться приложениями для отображения сглаженных шрифтов.
  - Проигрыватель The Innotek Macromedia Flash Player позволяет просматривать flash-анимации.
  - Просмотрщик файлов PDF The Innotek OS/2 Kit для Adobe Acrobat reader поставляется как отдельная программа и плагин для веб-браузера.

- eCSCoNet (ISDNPM) обеспечивает несколько способов подключения к сети Интернет.

– Утилиты для работы с текстом:
  - Новый текстовый редактор AE. На смену старому системному редактору пришел новый, с возможностью печати, улучшенным интерфейсом пользователя.
  - Технология управления буфером обмена DCPack позволяет перекодировать текст в нужный вид (в комплекте поставляется русско-английский и англо-русский словарь).

– Файловые системы и дисковые утилиты:
  - Драйвер NTFS.IFS предоставляет доступ к дискам с файловой системой NTFS в режиме чтения.
  - Оболочка управления LVM[16]-томами MiniLVM имеет простой и понятный интерфейс.
  - Обновленный LVM не позволяет Windows 2000 отключать загрузочный том eComStation.
  - Утилита JRescuer позволяет восстанавливать удаленные файлы с томов JFS.

– Улучшенная поддержка оборудования: новые лицензированные драйверы, автоматическая установка драйверов для беспроводных и гигабитных сетевых карт; поддержка IDE-контроллеров SATA; универсальный видеодрайвер SciTech SNAP:
  - Улучшенная поддержка ноутбуков. Ядро системы поддерживает режимы энергосбережения Speedstep и Enhanced Speedstep (технология Intel Centrino).

---

[16] Logical Volume Manager — менеджер логических томов операционной системы GNU/Linux LVM представляет собой дополнительный уровень абстракции между физическими/логическими дисками и файловой системой. Это достигается путём разбивки изначальных разделов на небольшие блоки (экстенты, обычно 4—32 МБайт) и объединения их в единый виртуальный том, точнее группу томов (volume group), которая далее разбивается на логические тома (logical volume). Для файловой системы логический том представлен как обычное блочное устройство, хотя отдельные экстенты тома могут находиться на разных физических устройствах (и даже сам экстент может быть распределён подобно RAID).

- Обновленные драйверы принтеров.

– Мультимедиа:
  - Новый инсталятор мультимедиа, обновленные классы мультимедиа.
  - Хранитель экрана Escape GL повышает общую привлекательность системы; пакет игрушек sNOa game pack демонстрирует мультимедийные возможности системы.
  - coolfm — полнофункциональная программа с современным интерфейсом для управления FM-радиоприемниками.
  - Видео-оверлей WarpOverlay! предоставляет современным мультимедийным программам наиболее быстрый механизм вывода изображений на экран.
  - Программа для записи CD-дисков, ограниченная (по скорости) версия известной программы для записи компакт-дисков RSJ, позволяет записывать диски без каких-либо дополнительных утилит. Поддержка всех современных записывающих устройств.
– Удаленное управление:
  - Удаленное управление рабочим столом PMVNC (IBM Desktop On Call также остается в составе системы).
  - Сервер HOBLink X/11 для OS/2 обеспечивает удаленный доступ к unix-приложениям для X-Window, при этом такие приложения работают на рабочем столе, как любые другие программы.
– Инсталляторы:
  - Система миграции предыдущих версий ОС поможет обновить OS/2 Warp 4, MCP1, MCP2, eComStation 1.0 или eComStation 1.1 до eComStation 1.2;
  - Популярный инсталлятор программных продуктов WarpIn устанавливается по умолчанию.

В 2008 году компания Adobe разрешила распространять Flash player в составе eComStation.

2.4 Операционные системы реального времени. Операционная система QNX

2.4.1 Общее представление об операционных системах реального времени

Английский термин «real-time» (и соответствующее ему в русском языке понятие «реальное время») является наиболее спорным и сложным термином. Данное понятие применяется в различных научно-технических областях и подразумевает некие действия, продолжительность которых определяется внешними процессами.

Специфическая особенность **систем реального времени заключается в том, что к ним предъявляются строгие временные требования, диктуемые окружением или определяемые их назначением** [4].

Вышеприведенное определение не является единственным, для понимания смысла данного понятия, приведем еще несколько определений:

– *Система называется системой реального времени (СРВ), если правильность ее функционирования зависит не только от логической корректности вычислений, но и от времени, за которое эти вычисления производятся. То есть для событий, происходящих в такой системе, КОГДА эти события происходят, так же важно, как логическая корректность самих событий.*

– *Реальное время (программное обеспечение) — относится к системе или режиму работы, в котором вычисления проводятся в течение времени, определяемого внешним процессом, с целью управления или мониторинга внешнего процесса по результатам этих вычислений (IEEE 610.12-1990).*

– *Системы реального времени — это системы, которые предсказуемо (в смысле времени реакции) реагируют на непредсказуемые (по времени появления) внешние события.*

Одной из функций таких систем может быть выполнение определенных действий в ответ на сигналы тревоги, и очень важно, чтобы они отвечали на них с определенной скоростью. В связи с этим существует разделение систем реального времени на два типа:

1. Системы с жесткими временными характеристиками — системы ***жесткого реального времени***.

2. Системы с нежесткими временными характеристиками — системы ***мягкого реального времени***.

Системой ***жесткого*** реального времени называется система, где неспособность обеспечить реакцию на какие-либо события в заданное время является отказом и ведет к невозможности решения поставленной задачи. Многие теоретики ставят здесь точку, из чего следует, что время реакции в жестких системах может составлять и секунды, и часы, и недели. Однако большинство практиков считают, что время реакции в системах жесткого реального времени должно быть все-таки минимальным. Большинство систем жесткого реального времени являются системами контроля и управления. Такие СРВ сложны в реализации, так как для них предъявляются особые требования в вопросах безопасности.

Точного определения для ***мягкого*** реального времени не существует, поэтому отнесем сюда все СРВ, не попадающие в категорию жестких. Так как система мягкого реального времени может не успевать ВСЕ делать ВСЕГДА в заданное время, возникает проблема определения критериев успешности (нормальности) ее функционирования. Вопрос этот совсем не простой, так как в зависимости от функций системы это может быть максимальная задержка в выполнении каких-либо операций, средняя своевременность обработки событий и т.п. Более того, эти критерии влияют на то, какой алгоритм планирования задач является оптимальным.

Также СРВ можно разделить на системы ***специализированные*** и ***универсальные***:

Специализированной СРВ называется система, где конкретные временные требования определены. Такая система должна быть специально спроектирована для удовлетворения этих требований.

Универсальная СРВ должна уметь выполнять произвольные (заранее не определенные) временные задачи без применения специальной техники. Разработка таких систем, безусловно, является самой сложной задачей, хотя обычно требования,

предъявляемые к таким системам, мягче, чем требования для специализированных систем.

Если СРВ строится как программный комплекс, то в общем виде она может быть представлена как комбинация трех компонентов: прикладное программное обеспечение, операционная система реального времени (ОСРВ) и аппаратное обеспечение.

В целом ряде задач автоматизации программные комплексы должны работать как составная часть более крупных автоматических систем без непосредственного участия человека. В таких случаях СРВ называют встраиваемыми.

Встраиваемые системы (Embedded system) можно определить как программное и аппаратное обеспечение, составляющее компоненты другой, большей системы и работающее без вмешательства человека.

В течение длительно времени основными потребителями СРВ были военная и космическая области. Сейчас ситуация изменилась, и СРВ можно встретить даже в товарах народного потребления.

Основные области применения СРВ:

- Военная и космическая области:
  - бортовое и встраиваемое оборудование;
  - радары, системы измерения и управления;
  - цифровые видеосистемы, симуляторы;
  - ракеты, системы определения местоположения и привязки к местности.
- Промышленность:
  - автоматические системы управления производством; автоматические системы управления технологическими процессами;
  - автомобилестроение: симуляторы, системы управления мотором, автоматическое сцепление …
  - энергетика: сбор информации, управление данными и оборудованием …
  - телекоммуникации: коммуникационное оборудование, сетевые коммутаторы, телефонные станции …
  - банковское оборудование: банкоматы …
- Товары широкого применения:
  - мобильные телефоны;

- цифровое телевидение: мультимедиа, видеосервисы, цифровые телевизионные декодеры …
- компьютерное и офисное оборудование.

Часто ОСРВ существуют в нескольких вариантах, например в полном и сокращенном, когда объем системы составляет несколько килобайтов.

Вычислительные установки, на которых используются СРВ, можно разделить на следующие платформы:

«Обычные» компьютеры. По логическому устройству совпадают с настольными компьютерами. Аппаратное устройство несколько отличается. Для обеспечения минимального времени простоя в случае технической неполадки процессор, память и другие элементы размещаются на съемной плате, вставляемой в специальный разъем так называемой «пассивной» платы. В другие разъемы этой платы вставляются платы периферийных контролеров и другое оборудование. Сам компьютер помещается в специальный корпус, обеспечивающий защиту от пыли и механических повреждений. В качестве мониторов используются жидкокристаллические дисплеи, иногда с сенсорочувствительным покрытием.

Основное доминирующее положение на этих компьютерах занимают процессоры Intel 80x86.

Подобные вычислительные системы обычно не используются для непосредственного управления промышленным или иным оборудованием. Они служат как терминалы для взаимодействия с промышленными компьютерами и встроенными контролерами, для визуализации состояния оборудования и технологического процесса.

На таких компьютерах, как правило, в качестве операционной системы (ОС) используют классические ОС (с разделением времени) с дополнительными программными комплексами, адаптирующими их к требованиям реального времени.

Промышленные компьютеры. Состоят из одной платы, на которой размещены процессор, контролер памяти и память различных видов (ОЗУ, ПЗУ, статическое ОЗУ, флэш-память).

Несмотря на наличие контроллеров SCSI (Small Computer System Interface), очень часто СРВ работает без дисковых накопителей. Это связано с тем, что дисковые накопители не отве-

чают требованиям, предъявляемым к системам реального времени, таким, как надежность, устойчивость к вибрациям, габаритам и времени готовности после включения питания.

Плата помещается в специальный корпус, в котором установлены разъемы шины и источник питания. Корпус обеспечивает специальный температурный режим, защиту от пыли и механических повреждений. В этот же корпус вставляются цифроаналоговые и аналогово-цифровые преобразователи, через которых осуществляется ввод/вывод управляющей информации, управление электромоторами и т.п.

Среди промышленных процессоров доминируют процессоры семейств PowerPC (Motorola — IBM), Motorola 68xxx (Motorola). Так же широкую нишу занимают процессоры семейства SPARC (SUN), Intel (Intel), ARM (ARM).

При выборе процессора определяющими факторами являются получение требуемой производительности при наименьшей тактовой частоте, а также время между переключением задач и реакции на прерывания.

Промышленные компьютеры используются для непосредственного управления промышленным или иным оборудованием. Они часто не имеют монитора и клавиатуры. Для взаимодействия с ними используются обычные компьютеры, соединенные с ними через порты или Ethernet.

Встраиваемые системы. Устанавливаются внутрь оборудования, которым они управляют. Для крупного оборудования совпадают с промышленными компьютерами. Для меньшего оборудования представляют собой процессор с сопутствующими элементами, размещенными на одной плате с другими электронными компонентами этого оборудования.

Отметим основные особенности ОСРВ, диктуемые необходимостью работы на промышленном компьютере:

– Система часто должна работать на бездисковом компьютере и осуществлять начальную загрузку из ПЗУ. В силу этого должны учитываться следующие факторы:

- критически важным является размер системы;
- для экономии места в ПЗУ часть системы хранится в сжатом виде и загружается в ОЗУ по мере необходимости;

- система часто позволяет исполнять код как в ОЗУ, так и в ПЗУ;
- при наличии свободного места в ОЗУ система часто копирует код из более медленного ПЗУ в ОЗУ;
- сама система, как правило, создается на другом компьютере — «обычном».

– Система должна по возможности использовать как можно большее число типов процессоров, что дает возможность потребителю выбрать процессор необходимой мощности.

– Система должна по возможности поддерживать более широкий ряд специального оборудования (периферийные контроллеры, таймеры и т.д.).

– Критически важным параметром является возможность предсказания времени реакции на прерывания.

2.4.2 Особенности архитектура системы QNX

Мощной операционной системой реального времени (ОСРВ), позволяющей проектировать сложные программные системы, работающие в реальном времени как на одном-единственном компьютере, так и в локальной вычислительной сети, является ***операционная система QNX***. Встроенные средства операционной системы QNX обеспечивают поддержку многозадачного режима на одном компьютере и взаимодействие параллельно выполняемых задач на разных компьютерах, работающих в среде локальной вычислительной сети. Основным языком программирования в системе является язык С. Основная операционная среда соответствует стандартам POSIX-интерфейса. Это позволяет с небольшими доработками перенести необходимое накопленное программное обеспечение в среду операционной системы QNX для организации работы в среде распределенной обработки.

ОС QNX является сетевой, мультизадачной, многопользовательской (многотерминальной) и масштабируемой системой. С точки зрения пользовательского интерфейса и API она очень похожа на UNIX. Однако QNX не является версией операционной системы UNIX, хотя почему-то многие так считают. Она была разработана, что называется «с нуля», канадской фирмой

QNX Software Systems Limited в 1989 году по заказу Министерства обороны США. Причем эта система построена на совершенно других архитектурных принципах, отличных от принципов, использованных при создании ОС UNIX.

QNX была первой коммерческой ОС, построенной на принципах микроядра и обмена сообщениями. Система реализована в виде совокупности независимых, но взаимодействующих через обмен сообщениями процессов различного уровня (менеджеров и драйверов), каждый из которых реализует определенный вид сервиса.

QNX — это ОС реального времени, позволяющая эффективно организовать распределенные вычисления. В системе реализована концепция связи между задачами на основе сообщений, посылаемых от одной задачи к другой, причем задачи эти могут находиться как на одном и том же компьютере, так и на удаленных, но связанных между собой локальной вычислительной сетью. Реальное время и концепция связи между процессами в виде сообщений оказывают решающее влияние на разрабатываемое для QNX программное обеспечение и на программиста, стремящегося с максимальной выгодой использовать преимущества системы. Микроядро имеет объем в несколько десятков килобайт (в одной из версий — 10 Кбайт, в другой — менее 32 Кбайт), то есть это одно из самых маленьких ядер среди всех существующих операционных систем. В этом объеме помещаются [5]:

- механизм передачи сообщений между процессами (IPC);
- redirect (перенаправление) прерываний;
- блок планирования выполнения задач;
- сетевой интерфейс для перенаправления сообщений (менеджер Net).

Механизм передачи межпроцессных сообщений предназначен для пересылки сообщений между процессами и является одной из важнейших частей ОС, так как все общение между процессами, в том числе и системными, происходит через сообщения. Сообщение в QNX — это последовательность байтов произвольной длины (0—65535 байтов), произвольного формата. Протокол обмена сообщениями организуется как блокировка

задачи для ожидания сообщения. Одна задача посылает другой сообщение и при этом блокируется сама, ожидая ответа. Первая задача разблокировывается, обрабатывает сообщение и отвечает, разблокируя при этом вторую задачу.

Сообщения и ответы, пересылаемые между процессами при их взаимодействии, находятся в теле отправляющего их процесса до того момента, когда они могут быть приняты. Это значит, что, с одной стороны, уменьшается вероятность повреждения сообщения в процессе передачи, а с другой — уменьшается объем оперативной памяти, необходимый для работы ядра. Кроме того, уменьшается число пересылок из памяти в память, что разгружает процессор. Особенностью процесса передачи сообщений является то, что в сети, состоящей из нескольких компьютеров под управлением QNX, сообщения могут прозрачно передаваться процессам, выполняющимся на любом из узлов. Определены в QNX еще и два дополнительных метода передачи сообщений — ***метод представителей*** (Proxy) и ***метод сигналов*** (Signal).

Представители используются в случаях, когда процесс должен передать сообщение, но не должен при этом блокироваться на передачу. В таком случае вызывается функция qnx_proxy_attach() и создается представитель. Он накапливает в себе сообщения, которые должны быть доставлены другим процессам. Любой процесс, знающий идентификатор представителя, может вызвать функцию Trigger(), после чего будет доставлено первое в очереди сообщение. Функция Trigger() может вызываться несколько раз, и каждый раз представитель будет доставлять следующее сообщение. При этом представитель содержит буфер, в котором может храниться до 65535 сообщений.

Сигналы уже давно используются в ОС UNIX. Система QNX поддерживает **множество сигналов**, совместимых с POSIX, большое количество сигналов, традиционно используемых в UNIX. Поддержка этих сигналов реализована для совместимости с переносимыми приложениями, и ни один из системных процессов QNX их не генерирует. Также системой QNX поддерживается несколько сигналов, специфичных для самой QNX. По умолчанию любой сигнал, полученный процессом, приводит к завершению процесса, кроме нескольких, которые

по умолчанию игнорируются. Но процесс, имеющий приоритет уровня «superuser», может защититься от нежелательных сигналов. В любом случае процесс может содержать обработчик для каждого возможного сигнала. Сигналы удобно рассматривать как разновидность программных прерываний.

Redirect прерываний является частью ядра и занимается перенаправлением аппаратных прерываний в связанные с ними процессы. Благодаря такому подходу возникает один побочный эффект — с аппаратной частью компьютера работает ядро, оно перенаправляет прерывания процессам — обработчикам прерываний. Обработчики прерываний обычно встроены в процессы, хотя каждый из них исполняется асинхронно с процессом, в который он встроен. Обработчик исполняется в контексте процесса и имеет доступ ко всем глобальным переменным процесса. При работе обработчика прерываний прерывания разрешены, но обработчик приостанавливается только в том случае, если произошло более высокоприоритетное прерывание. Если это позволяется аппаратной частью, к одному прерыванию может быть подключено несколько обработчиков, и каждый из них получит управление при возникновении прерывания.

Этот механизм позволяет пользователю избежать работы с аппаратным обеспечением напрямую и тем самым не допустить конфликтов между различными процессами, работающими с одним и тем же устройством. Для обработки сигналов от внешних устройств чрезвычайно важно минимизировать время между возникновением события и началом непосредственной его обработки. Этот фактор существен в любой области применения — от работы терминальных устройств до возможности обработки высокочастотных сигналов.

Блок планирования выполнения задач (диспетчер задач) занимается обеспечением многозадачности. В этой части ОС QNX предоставляет разработчику огромный простор для выбора той методики выделения ресурсов процессора задаче, которая обеспечит наиболее подходящие условия для критических приложений или обеспечит такие условия для некритических приложений, что они выполнятся за разумное время, не мешая работе критических приложений.

К выполнению своих функций как диспетчера ядро приступает в следующих случаях:

– при выходе какого-либо процесса из блокированного состояния;

– при истечении кванта времени для процесса, владеющего CPU;

– при прерывании работающего процесса каким-либо событием.

Диспетчер выбирает процесс для запуска среди неблокированных процессов в порядке значений их приоритетов, которые располагаются в диапазоне от 0 (наименьшего) до 31 (наибольшего). Обслуживание каждого из процессов зависит от метода диспетчеризации, с которым он работает. Уровень приоритета и метод диспетчеризации могут динамически меняться во время работы. В QNX существуют три метода диспетчеризации:

– FIFO (первым пришел — первым обслужен);

– round-robin, при котором процессу выделяется определенный квант времени для работы;

– адаптивный, который является наиболее используемым.

Первый наиболее близок к кооперативной многозадачности, то есть процесс выполняется до тех пор, пока он не перейдет в состояние ожидания сообщения, состояние ожидания ответа на сообщение или не отдаст управление ядру. При переходе в одно из таких состояний процесс помещается последним в очередь процессов с таким же уровнем приоритета, а управление передается процессу с наибольшим приоритетом.

Во втором варианте все происходит так же, как и в предыдущем, с той разницей, что период, в течение которого процесс может работать без перерыва, ограничивается неким квантом времени.

Процесс, работающий с адаптивным методом, в QNX ведет себя следующим образом. Когда процесс полностью использовал выделенный ему квант времени, его приоритет снижается на 1, если в системе есть процессы с тем же уровнем приоритета, готовые к исполнению. При этом если процесс с пониженным приоритетом остается не обслуженным в течение секунды, его

приоритет увеличивается на 1; если же процесс блокируется, ему возвращается оригинальное значение приоритета.

По умолчанию процессы запускаются в режиме адаптивной многозадачности. В этом же режиме работают все системные утилиты QNX. Процессы, работающие в разных режимах многозадачности, могут одновременно находиться в памяти и исполняться. Важным элементом реализации многозадачности является приоритет процесса. Обычно приоритет процесса устанавливается при его запуске. Но есть дополнительная возможность, называемая «вызываемый клиентом приоритет». Как правило, она реализуется для серверных процессов, исполняющих запросы на какое-либо обслуживание. При этом приоритет процесса-сервера устанавливается только на время обработки запроса и становится равным приоритету процесса-клиента.

Сетевой интерфейс в системе QNX является неотъемлемой частью ядра. Он, конечно, взаимодействует с сетевым адаптером через сетевой драйвер, но базовые сетевые сервисы реализованы на уровне ядра. При этом передача сообщения процессу, находящемуся на другом компьютере, ничем не отличается с точки зрения приложения от передачи сообщения процессу, выполняющемуся на том же компьютере. Благодаря такой организации сеть превращается в однородную вычислительную среду. При этом для большинства приложений не имеет значения, с какого компьютера они были запущены, на каком исполняются и куда поступают результаты их работы. Такое решение принципиально отличает QNX от остальных ОС, которые тоже имеют все необходимые средства для работы в сети, и делает системы, работающие под ее управлением, по-настоящему распределенными.

Все сервисы QNX, не реализованные непосредственно в ядре, работают как стандартные процессы в полном соответствии с основными концепциями микроядерной архитектуры. С точки зрения операционной системы системные процессы ничем не отличаются от всех остальных, как, впрочем, и драйверы устройств. Единственное, что нужно сделать при организации нового драйвера устройства в QNX для того, чтобы он стал частью ОС, — это изменить конфигурационный файл системы так, чтобы драйвер запускался при загрузке.

2.4.3 Основные механизмы QNX

QNX является сетевой операционной системой и позволяет организовать эффективные распределенные вычисления. Для организации сети на каждой машине, называемой узлом, помимо ядра и менеджера процессов должен быть запущен уже упомянутый нами выше менеджер Net. Менеджер Net не зависит от аппаратной реализации сети. Данная аппаратная независимость обеспечивается за счет использования сетевых драйверов. В QNX имеются драйверы для различных сетей, например Ethernet, Arcnet, Token Ring. Кроме этого, имеется возможность организации сети через последовательный канал или модем.

В QNX последней, четвертой, версии полностью реализовано встроенное сетевое взаимодействие «точка-точка». Например, находясь на машине A, пользователь может скопировать файл с гибкого диска, подключенного к машине B, на жесткий диск, подключенный к машине C. По существу, сеть из машин QNX действует как один мощный компьютер. Любые ресурсы (модемы, диски, принтеры) могут быть добавлены к системе простым их подключением к любой машине в сети. QNX поддерживает одновременную работу в сетях Ethernet, Arcnet, Serial и Token Ring и обеспечивает более чем один-единственный путь для коммуникации, а также балансировку нагрузки в сетях. Если кабель или сетевая плата выходят из строя таким образом, что связь через сеть прекращается, то система будет автоматически перенаправлять данные через другую сеть. Это происходит в режиме «on-line», предоставляя пользователю автоматическую сетевую избыточность и увеличивая скорость коммуникаций во всей системе.

Каждому узлу в сети соответствует уникальный целочисленный идентификатор — логический номер узла. Любой поток в сети QNX имеет прозрачный доступ (при наличии достаточных привилегий) ко всем ресурсам сети, то же самое относится и к взаимодействию потоков. Для взаимодействия потоков, находящихся на разных узлах сети, используются те же самые вызовы ядра, что и для потоков, выполняемых на одном узле. В том случае, если потоки находятся на разных узлах сети, ядро переадресует запрос менеджеру сети. Для организации обмена в

сети используется надежный и эффективный протокол транспортного уровня FLEET. Каждый из узлов может принадлежать одновременно нескольким QNX-сетям. В том случае, если сетевое взаимодействие может быть реализовано несколькими путями, для передачи выбирается незагруженная и более скоростная сеть.

Сетевое взаимодействие является узким местом в большинстве операционных систем и обычно создает значительные проблемы для систем реального времени. Для того чтобы обойти это препятствие, разработчики QNX создали собственную специальную сетевую технологию FLEET и соответствующий протокол транспортного уровня FTL (FLEET Transport Layer). Этот протокол не базируется ни на одном из распространенных сетевых протоколов типа IPX или NetBios и обладает рядом качеств, которые делают его уникальным. Основные его качества зашифрованы в аббревиатуре FLEET (http://support.qnx.com) и представлены в таблице 2.7.

Таблица 2.7 — **Составляющие аббревиатуры «FLEET»**

Название составляющего процесса сетевой технологии FLEET	Содержание процесса
Fault-Tolerant Networking	QNX может одновременно использовать несколько физических сетей. При выходе из строя любой из них данные будут автоматически перенаправлены «на лету» через другую сеть
Load-Balancing on the Fly	При наличии нескольких физических соединений QNX автоматически распараллеливает передачу пакетов по соответствующим сетям
Efficient Performance	Специальные драйверы, разрабатываемые фирмой QSSL для широкого спектра оборудования, позволяют с максимальной эффективностью использовать сетевое оборудование
Extensible Architecture	Любые новые типы сетей могут быть поддержаны путем добавления соответствующих драйверов

Окончание табл. 2.7

Название составляющего процесса сетевой технологии FLEET	Содержание процесса
Extensible Architecture	Любые новые типы сетей могут быть поддержаны путем добавления соответствующих драйверов
TransparcntDistributed Processing	Благодаря отсутствию разницы между передачей сообщений в пределах одного узла и между узлами нет необходимости вносить какие-либо изменения в приложения для того, чтобы они могли взаимодействовать через сеть

Благодаря этой технологии сеть компьютеров с QNX фактически можно представлять как один виртуальный суперкомпьютер. Все ресурсы любого из узлов сети автоматически доступны другим, и для этого не нужны специальные «фокусы» с использованием технологии RPC. Это значит, что любая программа может быть запущена на любом узле, при этом ее входные и выходные потоки могут быть направлены на любое устройство на любых других узлах.

Вопросы для самопроверки

1. Приведите основные характеристики ОС MS-DOS.
2. Программный продукт Windows 3.0 является ли операционной системой? Обоснуйте свой ответ.
3. Какие новые технологии были реализованы в ОС Windows 95?
4. Как называются две ветви развития операционных систем на платформе Microsoft Windows?
5. В чем заключались новые особенности ОС Windows 2000 от предшествующих версий операционных систем?
6. Благодаря каким возможностям операционная система Windows XP завоевала большой рынок программного обеспечения?
7. В чем состоит отличие версий ОС Windows Vista и Windows XP?

8. В чем состоит отличие версий ОС Windows 2008 Server и Windows 2003 Server?

9. Кто является первым разработчиком операционной системы Unix?

10. Чем можно объяснить обилие версий операционной системы Unix?

11. Как называется стандарт на разработку операционных систем, возникший на основе Unix?

12. Приведите общие черты ОС Unix, не зависящие от версий.

13. Приведите названия нескольких аппаратных платформ и операционных систем, функционирующих на них, на основе ОС Unix.

14. Опишите архитектуру программного обеспечения X Window.

15. Какие функциональные возможности имели версии операционной системы OS/2?

16. Почему фирма IBM, имея мощную операционную систему, не завоевала рынок программного обеспечения для персональных компьютеров?

17. Что собой представляет оболочка Workplace Shell в операционной системе OS/2?

18. Какая операционная система пришла на смену OS/2 и каковы ее функциональные возможности?

19. Что собой представляют операционные системы реального времени?

20. В чем отличие «жесткого реального времени» от «мягкого реального времени»?

21. На какой архитектуре построено ядро операционной системы QNX?

22. Какие методы передачи информации существуют между процессами в QNX?

23. Что собой представляет технология FLEET, реализованная в операционной системе QNX?

3 ИНТЕРФЕЙСЫ ОПЕРАЦИОННЫХ СИСТЕМ

3.1 Интерфейс командной строки ОС Windows

Интерфейсы операционных систем можно разделить на два класса: графические интерфейсы пользователя (GUI) и **интерфейсы командной строки (CPI — Command Prompt Interface)**.

Графический интерфейс пользователя в Windows обеспечивается процессом Explorer и, как правило, хорошо знаком большинству пользователей персональных компьютеров. Графические интерфейсы систем на платформе Unix бывают различными, что связано с большим количеством версий, но работа в них мало чем отличается от работы в операционной системе Windows. Графический интерфейс в Linux чаще всего реализуется с помощью графических оболочек KDE и Gnome. Загрузка операционной системы Linux Red Hat заканчивается тем, что на экране появляется окно графической оболочки Gnome.

Эффективная профессиональная работа опытного пользователя с операционной системой компьютера немыслима без овладения интерфейсом, обеспечиваемым командной строкой [6]. Преимуществом данного интерфейса служит возможность более гибко управлять ресурсами системы, чем с помощью графического интерфейса.

Интерфейс командной строки в ОС Windows присутствует, но играет для пользователей вспомогательную роль. В свое время он формировался как некое подмножество команд интерфейса Unix-подобных систем и особого развития не получил. Однако следует сказать, что интерфейс командной строки во многих нештатных ситуациях остается единственным средством определения рассогласований и «тонкой настройки» аппаратно-программных средств. В последних версиях операционных систем Microsoft (Windows Server 2008) интерфейс командной строки получил дальнейшее развитие и превратился в мощный инструмент администрирования системы — оболочку Power Shell.

В новых версиях операционных систем Windows с учетом роста сложности аппаратной и программной частей компьютерных систем добавлен ряд команд, позволяющих решать задачи администрирования системы. Часть команд, заимствованных из

MS DOS получили дополнительные возможности. Например, такие команды, как dir, copy, xcopy; rename и др., в новых редакциях Windows могут работать с длинными именами файлов.

Включение режима командной строки может быть выполнено двумя способами (рис. 3.1):

1. Нажать на экране кнопку «Пуск» → «Выполнить», затем в появившемся окне набрать cmd.

2. Выбрать из главного меню: «Пуск» → «Программы» → «Стандартные» → «Командная строка».

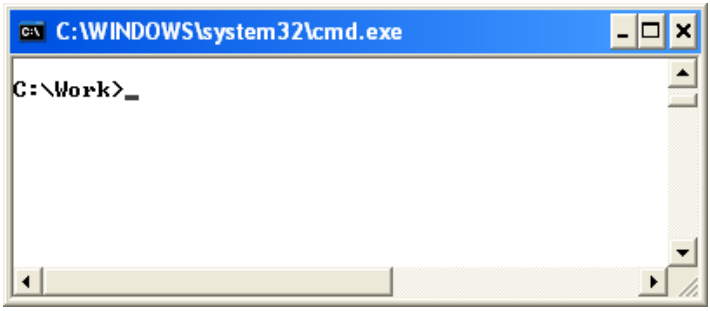

Рис. 3.1 — Окно «Командная строка»

Здесь указываются версия работающей операционной системы и строка приглашения с активным диском и рабочим каталогом (папкой). Выключение режима обеспечивается набором в строке приглашения команды exit и ее выполнением при нажатии клавиши <Enter>.

Смена текущего диска указывается путем указания его имени и двоеточия на конце. Например, чтобы перейти на диск D, необходимо указать D: и нажать клавишу <Enter>.

Перечень команд. В состав внутренних команд Windows входит около 70 команд. Перечень команд можно посмотреть с помощью команды HELP.

Поскольку весь перечень команд перекрывает размер экрана дисплея, то для ознакомления с каждым элементом перечня следует использовать полосу прокрутки окна или вызов на экран частей перечня постранично. Для этого следует набрать более сложную команду, состоящую из конвейера двух команд ***HELP | MORE*** (рис. 3.2).

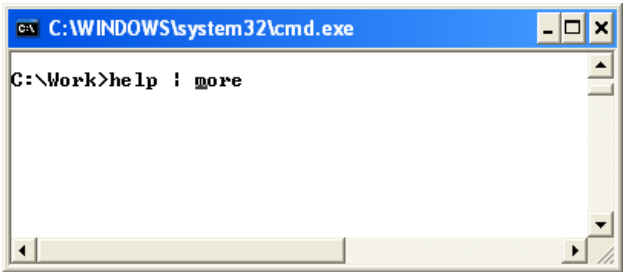

Рис. 3.2 — Ввод команд в командной строке

Список команд лучше рассмотреть по группам:
- команды справочной системы (таб. 3.1);
- команды файловой системы (таб. 3.2);
- команды управления работой ОС (таб. 3.3);
- команды пакетных (командных) файлов (таб. 3.4).

Таблица 3.1 — **Команды справочной системы**

HELP	Выводит справочную информацию о системе команд с версии Windows 2000
HELP имя_команды	Выводит справочную информацию о набранной команде
имя_команды /?	Выводит справочную информацию о набранной команде

Таблица 3.2 — **Команды файловой системы**

ATTRIB	Отображение и изменение атрибутов файлов
CD CHDIR	Вывод имени либо смена текущей папки
CHKDSK	Проверка диска и вывод статистики
COMP	Сравнение содержимого двух файлов или двух наборов файлов
COPY	Копирование одного или нескольких файлов в другое место
DEL	Удаление одного или нескольких файлов
DIR	Вывод списка файлов и подпапок из указанной папки

Окончание табл. 3.2

DISKCOMP	Сравнение содержимого двух гибких дисков
DISKCOPY	Копирование содержимого одного гибкого диска на другой
ERASE	Удаление одного или нескольких файлов
FC	Сравнение двух файлов или двух наборов файлов и вывод различий между ними
FIND	Поиск текстовой строки в одном или нескольких файлах
FINDSTR	Поиск строк в файлах
FORMAT	Форматирование диска
LABEL	Создание, изменение и удаление меток тома для дисков
MD	Создание папки
MKDIR	
MOVE	Перемещение одного или нескольких файлов из одной папки в другую
PUSHD	Сохранение значения текущей активной папки и переход к другой папке
POPD	Восстановление предыдущего значения текущей активной папки, сохраненного с помощью команды PUSHD
PRINT	Вывод на печать содержимого текстовых файлов
RD	Удаление папки
REN	Переименование файлов и папок
RENAME	
REPLACE	Замещение файлов
RMDIR	Удаление папки
SORT	Сортировка ввода
TREE	Графическое отображение структуры папок заданного диска или заданной папки
TYPE	Вывод на экран содержимого текстовых файлов
VERIFY	Установка режима проверки записи файлов на диск
VOL	Вывод метки и серийного номера тома для диска
XCOPY	Копирование файлов и дерева папок

Таблица 3.3 — **Команды управления работой ОС**

ASSOC	Вывод или изменение связи между расширениями имени и типами файлов
AT	Выполнение команд и запуск программ по расписанию
BAEAK	Включение/выключение режима обработки комбинации клавиш CTRL+C
CACLS	Отображение/редактирование списков управления доступом к файлам различных пользователей
CHCP	Просмотр номера текущей кодовой страницы или изменение текущей кодовой страницы консоли
CHKNTFS	Отображение или изменение выполнения проверки диска во время загрузки
CLS	Очистка экрана на консоли
CMD	Запуск еще одного интерпретатора командных строк
COLOR	Установка цвета текста и фона, используемых по умолчанию
COMPACT	Отображение/изменение сжатия файлов в разделах NTFS
CONVERT	Преобразование дисковых томов FAT в NTFS
DATE	Вывод либо установка текущей даты
DOSKEY	Редактирование и повторный вызов командных строк. Создание макросов
FTYPE	Вывод либо изменение типов файлов, используемых при сопоставлении по расширениям имен файлов
GRAFTABL	Позволяет отображать расширенный набор символов в графическом режиме
MODE	Конфигурирование системных устройств
MORE	Последовательный вывод данных по частям размером в один экран
PATH	Вывод либо установка пути поиска исполняемых файлов
PROMPT	Изменение приглашения в командной строке
RECOVER	Восстановление читаемой информации с плохого или поврежденного диска
SET	Вывод, установка и удаление переменных среды
START	Запуск программы или команды в отдельном окне
SUBST	Сопоставляет заданному пути имя диска
VER	Вывод сведений о версии операционной системы

Таблица 3.4 — **Команды пакетных (командных) файлов**

CALL	Вызов одного пакетного файла из другого
ECHO	Вывод сообщений и переключение режима отображения команд на экране
ENDLOCAL	Конец локальных изменений среды для пакетного файла
EXIT	Завершение работы программы
FOR	Организация циклов для обработки наборов файлов или строк в файле
GOTO	Передача управления в отмеченную строку пакетного файла
IF	Оператор условного выполнения команд в пакетном файле
PAUSE	Приостановка выполнения пакетного файла и вывод сообщения
REM	Помещение комментариев в пакетные файлы
SETLOCAL	Начало локальных изменений среды для пакетного файла
SHIFT	Изменение содержимого (сдвиг) замещаемых параметров для пакетного файла

Кроме команд, перечисленных в таблицах, имеется еще одна группа для работы в компьютерных сетях. Перечень этих команд может быть получен командой ***NET /?*** (рис. 3.3) (табл. 3.5).

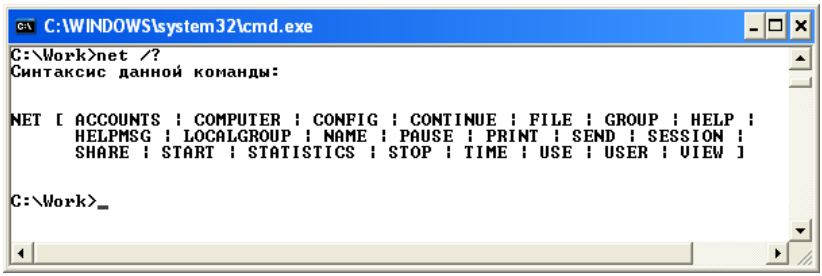

Рис. 3.3 — Вызов помощи по команде /?

Таблица 3.5 — **Сетевые команды**

NET ACCOUNTS	Обновление учетной базы пользователей, паролей и параметров подключения
NET COMPUTER	Добавление и удаление имени компьютера в базе данных домена
NET CONFIG	Сведения о настраиваемых службах и их изменение
NET CONTINUE	Активизация приостановленной службы, имя которой указано в качестве параметра
NET FILE	Вывод имен открытых файлов на сервере и количества их блокировок
NET GROUP	Вывод, добавление и изменение глобальных групп на сервере домена
NET HELPMSG	Выдача справок об ошибках и предупреждающих сообщениях
NET LOCALGROUP	Отображение и изменение локальных групп
NET NAME	Добавление и удаление имени, называемого псевдонимом. Псевдоним — имя компьютера, принимающего сообщения
NET PAUSE	Приостановка работы службы, указанной параметром в команде
NET PRINT	Отражение состояния, управление заданиями и очередями принтеров
NET SEND	Пересылка сообщения адресату: пользователю, компьютеру, псевдониму
NET SESSION	Вывод списка подключенных к компьютеру пользователей и его изменение
NET SHARE	Создание и удаление совместно используемых ресурсов сети
NET START	Вывод списка запущенных служб и его изменение
NET STATISTICS	Вывод содержимого журнала статистики для служб компьютера или сервера
NET STOP	Остановка работы службы, указанной параметром в команде
NET TIME	Синхронизация часов компьютеров, включенных в сеть
NET USE	Подключение компьютеров сети к сетевым ресурсам

Окончание табл. 3.5

NET USER	Добавление, редактирование и просмотр учетных сведений пользователей
NET VIEW	Просмотр списков компьютеров, доменов и общих ресурсов на указанном компьютере

Для вызова помощи для конкретных сетевых команд следует набирать net имя_команды /?.

Справочная информация по различным командам свидетельствует, что командой, набираемой в командной строке, является собственно имя команды, за которым могут следовать ключи (опции) — указания, модифицирующие поведение команды. Квадратные скобки в пояснениях обозначают, что эта информация не является обязательной при наборе команды. Ключи начинаются со знака / (слэша) и состоят из одного или нескольких символов. Кроме ключей, после команды могут следовать аргументы (параметры) — названия объектов, над которыми должна быть выполнена команда. Очень часто аргументами служат имена файлов и каталогов.

Ввод команды заканчивается нажатием клавиши <Enter>, после чего команда передается на исполнение командному процессору. В результате выполнения команды на экране дисплея могут появиться сообщения о ходе выполнения команды или об ошибках, а появление очередного приглашения (мигающего курсора) свидетельствует об успешном выполнении введенной команды и ожидании ввода следующей [6].

Замещаемые символы (метасимволы). Параметр в командной строке команды может включать замещаемые символы «?» и «*». Символ «вопросительный знак» заменяет один любой символ. Символ «звездочка» может заменять любую последовательность символов.

Пусть в каталоге в текущем каталоге содержится произвольный набор файлов. Команда DIR без параметров по умолчанию покажет нам весь перечень файлов в директории (рис. 3.4).

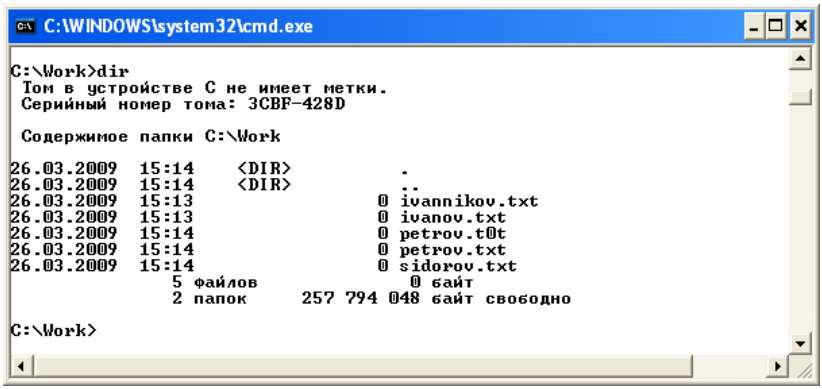

Рис. 3.4 — Результат выполнения команды DIR

Применим для вывода команды замещаемые параметры. Результат выполнения команды DIR *.TXT будет следующим (рис. 3.5). Строка с файлом PETROV.T0T не будет отображена, так как мы указали команде DIR показать все файлы с любым именем (символ *), но имеющим только расширение .TXT.

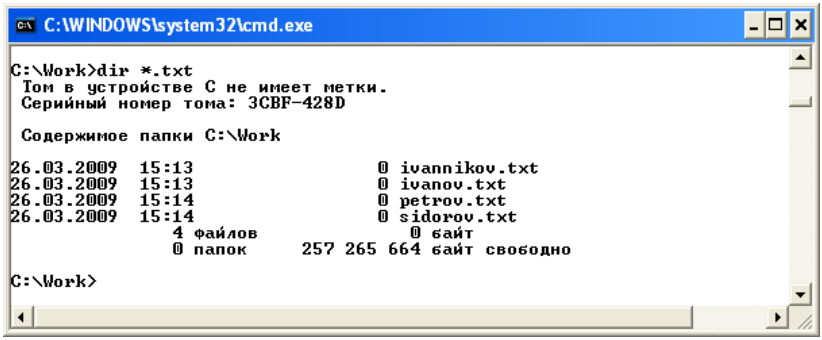

Рис. 3.5 — Результат выполнения команды DIR *.TXT

Выполним последовательно команды:

1. DIR PETROV.T* (рис. 3.6);
2. DIR PETROV.T? (рис. 3.7);
3. DIR PETROV.T?? (рис. 3.8).

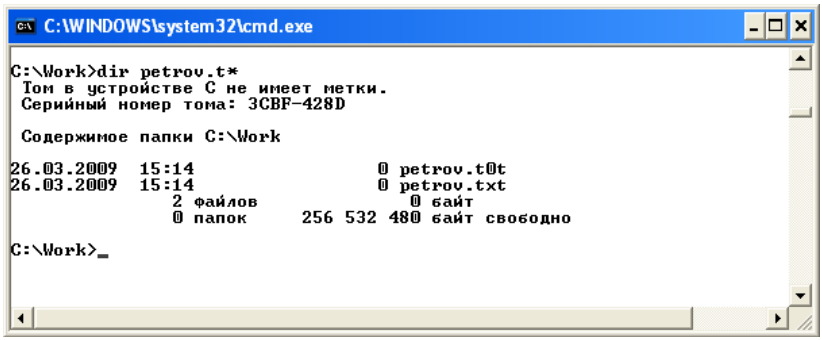

Рис. 3.6 — Результат выполнения команды DIR PETROV.T*

Первая команда говорит, что необходимо показать файлы, у которых имя «petrov», расширение начинается с символа «t», далее могут идти любые символы.

Вторая команда говорит, что необходимо показать файлы, у которых имя «petrov», расширение начинается с символа «t», далее может быть только один любой символ.

Третья команда говорит, что необходимо показать файлы, у которых имя «petrov», расширение начинается с символа «t», далее может быть только два любых символа.

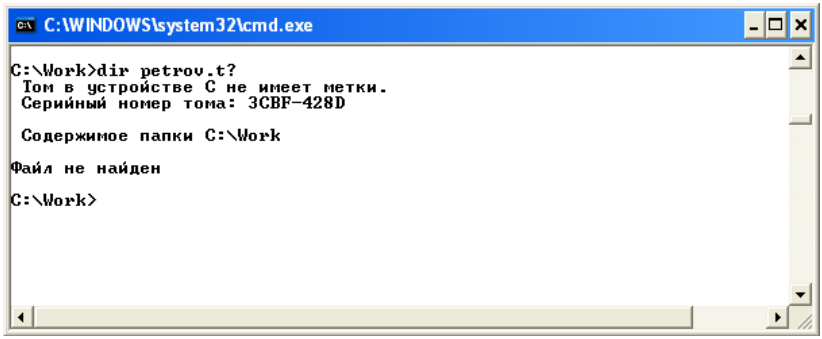

Рис. 3.7 — Результат выполнения команды DIR PETROV.T?

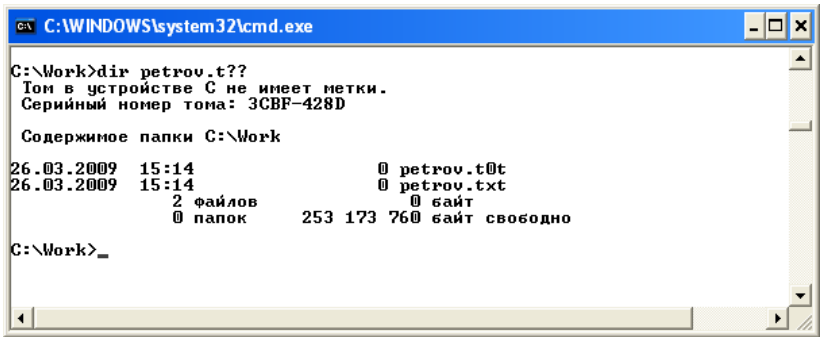

Рис. 3.8 — Результат выполнения команды DIR PETROV.T??

Как видно из рисунков, результаты выполнения первой и третьей команд совпали, так как расширение состоит из трех символов.

А результат выполнения второй команды не вывел ни одного файла, удовлетворяющего заданному условию.

Стандартные потоки ввода-вывода и перенаправление потоков. Термин CONsole используется для обозначения стандартных потоков ввода-вывода. Когда говорят о вводе с консоли, подразумевается ввод с клавиатуры. Когда говорят о выводе на консоль, подразумевают вывод на экран монитора. Существуют специальные символы для перенаправления стандартных потоков ввода-вывода:

> *приемник* — перенаправить стандартный вывод в приемник (если файл-приемник существует, то он будет создан заново).

>> *приемник* — перенаправить стандартный вывод в приемник (если файл-приемник существует, то он будет сохранен, а информация будет записана в конец файла).

< *источник* — перенаправить стандартный ввод из источника.

передатчик* | *приемник — передает вывод одной команды на вход другой.

Приведем несколько примеров с использованием перенаправления потока:

DIR > FILES.TXT — содержимое текущего каталога записать в текстовый файл;

TYPE FILE.TXT >> ARXIV.TXT — добавить в конец файла ARXIV.TXT содержимое файла FILE.TXT;

DATE < DATE.TXT — установить новую системную дату, значение взять из файла DATE.TXT;

TYPE PETROV.TXT | SORT — распечатывает на экране файл petrov.txt, отсортировав его строки.

Возможна комбинация символов перенаправления потоков:

TYPE PETROV.TXT | SORT > PETROV_SORT.TXT — записывает в файл PETROV_SORT.TXT содержимое файла PETROV.TXT, отсортировав его строки.

Создание текстовых файлов можно выполнить следующей командой:

COPY CON ДИСК:ПУТЬ\ИМЯ_ФАЙЛА.РАСШ

Например, для создания файла F1.TXT в текущем каталоге необходимо выполнить команду COPY CON F1.TXT.

Этой командой ввод текста с клавиатуры (консоли con) осуществляется в новый, создаваемый этой операцией файл F1.TXT. После набора каждой строки следует нажимать клавишу <Enter>. Окончание набора файла должно заканчиваться нажатием клавиш <Ctrl>+Z или F6 (признаком окончания файла), а затем <Enter>. Недостатком применения этой команды является то, что редактировать можно только текущую строку файла. После нажатия клавиши <Enter> ранее введенные строки уже недоступны. Таким образом, команда copy использует простейший однострочный редактор.

Атрибуты файлов. Каждый файл и каталог, находящиеся в компьютере, могут иметь атрибуты — характеристики, отражающие свойства объекта, которые используются операционной системой для корректной работы с ними. Атрибутами файла (файлов) могут быть следующие значения:

R — «только для чтения», то есть нельзя модифицировать файл и уничтожить его;

A — «архивный», т.е. +A обозначает, что снимались копии данного файла; соответственно, –A — файл является вновь созданным;

H — «скрытый», скрытые файлы не показываются командой DIR и некоторыми программными оболочками;

S — «системный», этот атрибут показывает, что файл является принадлежностью операционной системы.

Изменение атрибутов файлов осуществляется командой ATTRIB. Формат команды

ATTRIB +|–АТРИБУТ ДИСК:\МАРШРУТ\ИМЯ ФАЙЛА /S

Установка любого атрибута производится знаком + (плюс), отмена — знаком — (минус). Можно задавать изменение сразу нескольких атрибутов в любой последовательности. Ключ /S, стоящий в конце формата, указывает, что процесс изменения атрибутов файла (файлов) распространяется не только на текущий каталог, но и на все каталоги, подчиненные текущему.

Примеры:

ATTRIB +A +H +R PRIMER.TXT — присваивает файлу сразу три атрибута: только для чтения, архивный и скрытый файлу PRIMER.TXT.

ATTRIB -R A:*.* /S — с использованием шаблона снимает атрибут «только для чтения» у всех файлов диска А, т.е. находящихся на дискете.

Разработка командных файлов. Командный файл — это группа последовательных команд настройки компьютера на определенный режим или выполнение определенных операций. В простейшем случае командный файл может быть представлен в виде определенной последовательности отдельных команд операционной системы. Разработка командных файлов является мощным средством автоматизации подготовительных работ пользователей по настройке среды их работы [6].

При разработке командных файлов следует руководствоваться следующими правилами [6]:

1. Вызов на исполнение командного файла осуществляется командой следующего формата:

ДИСК:\ПОЛНЫЙ_ПУТЬ\ИМЯ_КОМ.ФАЙЛА [P1 P2 ... P10]

Содержимое в квадратных скобках указывает, что командный файл может иметь до 10 фактических параметров, замещающих формальные параметры, присутствующих в тексте файла. Команда SHIFT позволяет снять это ограничение.

2. Имя командного файла образуется по обычным правилам. Расширением должно быть только сочетание BAT или CMD.

3. Если текущим является каталог (папка), содержащий командный файл, то полный путь к командному файлу можно не указывать.

4. Командный файл выполняется командным процессором строка за строкой.

5. Выполнение командного файла может быть прекращено командами <Ctrl>+<Break> или <Ctrl>+C.

6. Из командного файла можно вызывать другой командный файл командой CALL (с возвратом) или обычной командой вызова (без возврата).

7. Командный файл может содержать любые внешние и внутренние команды операционной системы, а также специальные внутренние команды.

8. Формальные параметры, включаемые в строки командного файла, имеют вид %0, %1 и т.д. до %9. Фактические значения параметров вводятся в строке вызова командного файла; вводимые параметры подставляются на место формальных параметров %1, %2 и т.д. по порядку. На место формального параметра %0, если он встречается в тексте командного файла, подставляется имя самого командного файла.

9. Для обращения к переменным окружения их имена следует заключать в знаки %, например %TEXT%.

10. Перед выполнением очередной строки командного файла ее значение выводится на экран. Вывод любой строки командного файла на экран подавляется, если строка начинается с символа *@*.

Рассмотрим ***особенности применения специальных команд***:

Команда ECHO предназначается для отключения «эха» на экране дисплея, то есть она не позволяет выводить лишнюю информацию на монитор (блокирует выдачу на экран последовательностей команд, включенных в командный файл, и текстовых сообщений при выполнении этих команд). Форматы команды:

ECHO OFF — запрет вывода на экран;

ECHO ON — разрешение вывода на экран;

ECHO (без параметров) — запрос состояния эха (ON или OFF);

ECHO + текстовое сообщение — вывод текстового сообщения на экран;

ECHO %имя переменной окружения% — вывод текущего значения переменной окружения;

ECHO. (с точкой) — вывод пустой строки.

При использовании команды ECHO следует помнить:

– при запуске системы по умолчанию устанавливается режим «ECHO ON»;

– режим «ECHO OFF» действует только до конца командного файла или до очередного переключения режима командой «ECHO ON»;

– ECHO влияет только на вывод сообщений командного файла, но не влияет на вывод сообщений из программ пользователей, даже если они используют команды операционной системы;

– для подавления самой команды «ECHO OFF» надо поставить впереди знак @.

Для лучшего понимания содержимого командного файла используются комментарии, вводимые с помощью ***команды REM*** (remark — примечание). Командный процессор полностью игнорирует всю информацию, которая размещается за словом REM. Команда очень полезна, когда в командный файл включаются пояснения, описания работы файла или отдельных его команд, тестирования и отладки.

Для приостановки выполнения командного файла используется ***команда PAUSE***. Команда имеет формат:

PAUSE сообщение

При остановке работы командного файла на экране появляется текст строки сообщения в режиме ECHO ON, а под ним фраза «Press any key to continue» — Нажмите любую клавишу для продолжения (для продолжения работы файла).

Команду полезно использовать в тех случаях, когда, например, на экран дисплея выводится большое количество информации порциями по страницам, чтобы пользователь мог ее прочитать, осмыслить и перейти к следующему фрагменту. Команда полезна и в случаях, когда необходимо выполнить какие-то вспомогательные действия, например:

@ECHO ON

PAUSE Установите дискету с .... на дисковод A:
@ECHO OFF

Кроме того, команду PAUSE можно использовать и для управления работой командного файла. Если в ответ на команду PAUSE нажать <Ctrl>+C, то появляется вопрос:

«Terminate batch job (Y/N)?» — Завершить выполнение задания (командного файла)? Выбор Y — останавливает выполнение командного файла, а N — обеспечивает продолжение его выполнения. В случаях, когда командный файл выполняется с частыми остановками и появление множества фраз «Press any key to continue» нежелательно, строка с командой может выглядеть как PAUSE> NUL, то есть вывод переадресуется в несуществующее устройство nul.

Команда GOTO позволяет изменить привычную последовательность выполнения операторов (команд) командного файла. Когда командный процессор встречает строку с оператором GOTO, он просматривает все строки файла и отыскивает соответствующую метку — строку с двоеточием. Двоеточие может быть и не в первой позиции строки. Идентификатор метки должен иметь до восьми символов. Больше, чем восемь символов, в идентификаторе не воспринимается. Команда GOTO может использоваться самостоятельно или совместно с операцией IF.

Команда IF — условное выполнение команд, организует разветвление при выполнении командного файла. Формат оператора IF

IF условие команда.

В качестве условия обычно используются:

– проверка наличия файла. В этом случае в качестве условия записывается фраза exist диск:путь\имя_файла.расш;

– проверка кода завершения отдельных программ по значению внутренней переменной системы с именем ERRORLEVEL. В этом случае в качестве условия записывается фраза «ERRORLEVEL значение». Условие считается истинным, если код завершения ***равен или больше параметра «значение»***. Значение переменной errorlevel может формироваться многими утилитами и прикладными программами;

– проверка идентичности двух символьных строк. Строка условие при этом записывается в виде

строка_1==строка_2 (двойной знак =)

Предваряя любому из перечисленных условий слово NOT, можно проверять противоположное условие.

Для многократного выполнения отдельных команд применяется команда FOR. Она позволяет обрабатывать целые группы файлов. Команда имеет следующие форматы:

FOR %%переменная IN (набор) DO команда — для строк командных файлов,

FOR %переменная IN (набор) DO команда — для режима командной строки (автономного выполнения команды).

В качестве параметров команды используются:

– переменная — однобуквенная переменная, последовательно принимающая значения слов или имен файлов, перечисленных в параметре (набор);

– (набор) — одно или несколько символьных слов или спецификаций файлов. Спецификация файла имеет вид диск:путь\имя_файла.расш. Допускаются шаблоны групповых операций. Слова и спецификации файлов разделяются пробелами или запятыми. Максимальная длина строки набора — не более 127 символов;

– команда — команда DOS, выполняемая для каждого слова или файла из параметра «набор».

Обычно в командный файл можно передавать до 9 параметров, иногда этого недостаточно. ***Команда SHIFT*** (сдвиг) позволяет сдвигать строку параметров влево на один параметр. Применение данной команды позволяет снять ограничение на число параметров.

В учебном пособии приведены лишь фрагменты описания команды, более подробную информацию по командам лучше смотреть непосредственно, вызывая помощь в командной строке операционной системы.

Команда CHOICE — ожидает ответа пользователя. Данная команда является внешней, то есть необходимо наличие файла choice.exe. Данный файл должен располагаться в текущей папке или в какой-либо системной, чтобы операционная система могла найти его. Формат команды выглядит следующим образом:

CHOICE [/C[:]варианты] [/N] [/S] [/T[:]c,nn] [текст]

/C[:]варианты — варианты ответа пользователя.

По умолчанию строка включает два варианта: YN

/N Ни сами варианты, ни знак вопроса в строке приглашения не отображаются.

/S Учитывать регистр символов.

/T[:]c,nn Ответ «с» выбирается автоматически после nn секунд ожидания текст Строка приглашения

После выполнения команды переменная ERRORLEVEL приобретает ***значение, равное номеру выбранного варианта ответа***.

Приведем ***несколько примеров командных файлов***:

Пример 1 [6]. Пусть требуется создать командный файл test1.bat, который будет копировать из текущего каталога на дискету ряд текстовых файлов с проверкой правильности записи и удалением исходных файлов. Перед каждым удалением файла должно выдаваться предупреждающее сообщение. В момент приостановки можно прервать дальнейшее выполнение командного файла, нажав клавиши <Ctrl>+C.

```
:LOOP
COPY %1.TXT A:/V
PAUSE УДАЛЯЮ СКОПИРОВАННЫЙ ФАЙЛ
DEL %1.TXT
SHIFT
IF NOT %1.==. GOTO LOOP
```

Запуск этого файла следует выполнять командой

Test1.bat 01 02 03 04 05 06 07 08 09 10 11 12 и т.д., если текстовые файлы имеют имена 01.txt, 02.txt, 03.txt и т.д. Обратите внимание, что:

– по умолчанию здесь используется режим ECHO ON. В противном случае сообщения команды PAUSE были бы не видны;

– расширения текстовых файлов присоединяются к имени непосредственно в командах выполняемого файла.

Пример 2 [6]. Создадим файл test2.bat таким образом, чтобы можно было отыскивать и просматривать нужный файл в любом каталоге. Учитывая, что файлы могут иметь большие размеры, превышающие емкость одного экрана, обеспечим поэкранный просмотр файлов. Имя нужного файла будем задавать в

качестве параметра в строке вызова файла test2.bat. Например: test2.bat proba.txt

Здесь имя искомого файла proba.txt служит фактическим параметром, значение которого должно заменить формальные параметры %1 внутри командного файла. Таких параметров строка вызова может иметь от %1 до %9. Командный файл test2.bat будет иметь следующее содержание.

```
ECHO OFF
CLS
IF /%1==/ GOTO ERROR1
IF NOT EXIST %1 GOTO ERROR2
TYPE %1 | MORE
GOTO END
:ERROR1
ECHO ВЫ забыли указать имя искомого файла!
:GOTO END
:ERROR2
ECHO ФАЙЛА %1 на этом диске нет!
:END
```

В этом фрагменте два слэша / в операторе IF играют роль скобок.

Пример 3. Командный файл просит ввести цифру, соответствующую имени пользователя, и в зависимости от того, какая цифра была введена, устанавливает текущим тот или иной рабочий каталог и открывает окно CMD.

```
:BEGIN
  ECHO Введите номер пользователя
  ECHO 1 – Алексей 2 – Петр 3 – Иван 4 – остальные
  CHOICE /C:1234
  IF ERRORLEVEL 4 GOTO WORK
  IF ERRORLEVEL 3 GOTO IVAN
  IF ERRORLEVEL 2 GOTO PETER
  IF ERRORLEVEL 1 GOTO ALEX
  GOTO BEGIN
:IVAN
  CD IVAN
  START
```

```
:PETER
  CD PETER
  START
:ALEX
  CD ALEX
  START
:WORK
  CD WORK
```

3.2 Интерфейс командной строки ОС Unix

Большинство версий операционной системы Unix имеют графический интерфейс, подобный интерфейсу, используемому на компьютерах Macintosh и впоследствии в IBM-совместимых компьютерах с операционной системой Windows. Однако интерфейс командной строки до сих пор остается популярным среди программистов.

Изучение команд Unix-подобных операционных систем можно вести, установив систему на жесткий диск своего компьютера или загрузив Unix (Linux) с компакт-диска (CD) без инсталляции на жесткий диск. Удобно воспользоваться эмулятором Unix для операционной системы Windows. Эмулятор можно легко найти в Интернете (любая поисковая система находит различные эмуляторы по поисковому запросу: «эмулятор Unix»).

Приглашение к вводу команды в Unix может выглядеть по-разному: # — это приглашение для суперпользователя (root), вошедшего в систему; $ или [имя@localhost имя]$ — для обычных пользователей. Помощь по командам Unix можно получить, набрав man и через пробел — имя команды, например: man gnome.

Вход в систему производится в диалоге, когда система запрашивает имя пользователя и его пароль.

Выход из системы может производиться по-разному. Для выхода из системы служит команда logout, по которой прекращается сеанс работы с данным пользователем, но система не завершает свою работу. Прекратить текущий сеанс работы можно также, нажав одновременно три клавиши Ctrl+Alt+Backspace.

Для полного завершения работы нажимается Ctrl+Alt+Del.

Перечень команд. В состав Unix входит более трехсот команд. Данная операционная система обладает более богатыми методами управления ресурсами, чем ОС Windows.

При изучении системы команд Unix необходимо иметь в виду, что многие команды операционных систем MS DOS и Windows совпадают по имени и, частично, по функциям с командами Unix.

Таблица 3.6 — **Список команд Unix**

at	Выполнение команд и запуск программ по расписанию
cat	Вывод на экран содержимого файлов
cd	Выбор имени либо смена текущей папки
chmod	Изменение атрибутов прав доступа к файлам
cmp	Поиск различий между файлами (до первого различия)
cp	Копирование одного или нескольких файлов в другое место
date	Вывод текущей даты
df	Определение свободного пространства диска
diff	Поиск всех различий в файлах
du	Определение занятого пространства диска
ed	Вызов текстового редактора
exit	Выход из системы
kill	Послать сигнал процессу. Завершить процесс
ls	Вывод списка файлов в каталоге
mail	Вызов почтового клиента
man	Вызов помощи по командам
mesg	Установка режима запрета или разрешения сообщений
mkdir	Создание каталога
mv	Перенос файлов
news	Запуск клиента новостей
nice	Запуск программы с пониженным приоритетом
nohup	Выполнение программы после отключения терминала
pr	Распечатка файла по 66 строк
ps	Получение списка процессов
pwd	Определение своего рабочего каталога
rm	Удаление файла
rmdir	Удаление каталога
sh	Переход в порожденный командный процессор (shell)

Окончание табл. 3.6

tail	Вывод на экран содержимого файлов с конца
time	Информация о времени выполнения команды
touch	Заменяет время модификации файла на настоящее
uname	Информация о системе
wc	Подсчет числа строк, слов и символов в файле
who	Вывод активных пользователей
who am i	Вывод собственного имени
write	Установка связи с другим пользователем

Файлы и процессы, являются центральными понятиями операционной системы UNIX. Файловая подсистема управляет файлами, размещает записи файлов в отведенные для них места, управляет свободным пространством, доступом к файлам и поиском данных для пользователей.

Работа с файлами ведется с помощью команд. Команда представляет собой имя исполняемого файла (двоичного или текстового, так называемого скрипта, написанного на одном из специальных командных языков) или имя внутренней команды самого процессора. При активизации каждой такой команды операционная система создает процесс. Процессы взаимодействуют с подсистемой управления файлами и с аппаратными средствами, используя для этого совокупность специальных команд, таких, как ***open*** (для того чтобы открыть файл на чтение или запись), ***close***, ***read***, ***write***, ***stat*** (запросить атрибуты файла).

Подсистема управления процессами ядра ОС отвечает за синхронизацию процессов, их взаимодействие, распределение памяти и планирование выполнения процессов. По характеру выполнения процессы могут быть фоновыми и привилегированными. Любой запускаемый процесс по умолчанию будет выполняться как привилегированный (***foreground***). Это значит, что такой процесс постоянно связан с терминалом ЭВМ и делает невозможным выполнение еще каких-либо действий с системой, пока не завершится.

Фоновый процесс (***background***) после запуска освобождает терминал и позволяет перейти к другой задаче, не дожидаясь его завершения. Фоновая обработка наиболее пригодна для процессов, которые долго выполняются. Программы, выполняющиеся

в виде фоновых процессов, называются демонами (***daemon***). В любой момент времени в системе существуют десятки процессов, которые были запущены при старте операционной системы, вызваны ядром для обслуживания каких-либо событий, добавлены пользователем при запуске какой-либо задачи.

Обычно большинство процессов находится в состоянии ожидания — сна (sleep), не мешая остальным и дожидаясь сигнала для активизации. Кроме того, в системе можно найти процессы, закончившие работу, но еще не получившие разрешения на выгрузку из основной памяти, — эти процессы называются зомби. В ядре операционной системы находится таблица процессов, каждая запись которой описывает состояние одного из процессов.

Основное отличие **файловой системы Unix** от файловой системы Windows заключается в том, что в Unix отсутствует такое понятие, как логическое устройство. При указании пути к файлу в Unix имя устройства не упоминается. Дерево каталогов Unix «растет» из одного корня. Корневой каталог имеет предопределенное имя / (слэш). Этот же символ применяется и для разделения подкаталогов. Полный путь к файлу в Unix выглядит следующим образом:

/каталог1/каталог2/каталог3/ ... /файл.

Физически разные компоненты дерева каталогов Unix могут размещаться на разных дисках, но логически они принадлежат одной древовидной структуре с одним корневым узлом. Для объединения файловых систем различных устройств в одну структуру используется операция монтирования.

Сущность этой операции заключается в том, что каждое физическое устройство можно рассматривать, как свою собственную файловую систему (файловую систему устройства) с корневым каталогом /. Если этот раздел диска объявлен в операционной системе как корневой раздел (root), его каталог становится корневым каталогом всей файловой системы (файловой системы ЭВМ). Файловые системы остальных устройств должны быть смонтированы в каталогах файловой системы ЭВМ.

Операция монтирования связывает корневой каталог монтируемого раздела (устройства) с выбранным каталогом файловой системы ЭВМ — точкой монтирования. В результате мон-

тирования корневой каталог файловой системы устройства получает имя каталога, являющегося точкой монтирования, благодаря чему файловая система устройства «привязывается» к файловой системе ЭВМ в точке монтирования.

Таким образом, для монтирования файловой системы устройства в файловой системе ЭВМ необходимо сначала в файловой системе ЭВМ создать каталог, который будет точкой монтирования, а затем соединить две файловые системы командой ***mount***.

После монтирования сменных носителей их нельзя извлекать из устройства без демонтирования файловой системы. Для этой цели служит команда ***umount***.

При просмотре содержимого каталога можно увидеть, что информация о файле начинается с кода, содержащего 10 символов:

-rwxrwxrwx, или **drwxrwxrwx**, и **lrwxrwxrwx**.

Первый символ кода указывает тип файла:

– ***символ*** — означает, что это обычный файл, текстовый или двоичный, содержащий данные или программу;

– ***символ d*** указывает, что данный файл является каталогом;

– ***символ l*** указывает на то, что данный файл является символьной ссылкой.

Символы ***rwx*** определяют права доступа к файлу. Доступ к файлу могут иметь три категории пользователей:

– владелец файла;

– выделенная группа пользователей;

– остальные пользователи (не являющиеся владельцами файла и не входящие в выделенную группу).

Всем им могут быть установлены следующие права:

– ***символ r*** — разрешено чтение файла;

– ***символ w*** — разрешена запись в файл;

– ***символ x*** — разрешен запуск файла на исполнение.

Эти права всегда перечисляются подряд в порядке: ***rwx***. Если какое-либо право не предоставлено, вместо соответствующего символа ставится **-**. Например, ***r - -*** означает, что разрешено только чтение; ***-wx*** означает, что разрешены запись в файл и его исполнение.

Поскольку права определяются для трех видов пользователей, указанная триада повторяется трижды и образует запись из 9 символов, первые три из которых относятся к владельцу, вторые — к группе и третьи — к остальным пользователям. Например, запись rwx--xr-- означает, что владельцу файла разрешено все, выделенной группе только запуск на исполнение, остальным пользователям — только чтение.

Иногда указанные девять символов кодируются числом. Ключ к расшифровке цифрового кода приведен в виде таблицы 3.7.

Таблица 3.7 — **Ключ расшифровки цифрового кода прав доступа**

4	2	1	4	2	1	4	2	1
r	w	x	r	w	x	r	w	x
1 цифра			2 цифра			3 цифра		
4+2+1=7			4+2+1=7			4+2+1=7		

Установка и изменение режима доступа к файлу производится с помощью команды ***chmod***.

Формат команды chmod (change mode) для установки режима:

chmod <режим> <файлы>

Пример использования команды:

$ chmod 644 f1 f2 f3,

где 644 соответствует rw-r--r--

Формат команды chmod для изменения режима:

chmod <изменения> <файлы>

В изменениях используются обозначения:

- r — read — права на чтение;
- w — write — права на запись;
- x — execute — права на выполнение;
- u — user — права для владельца файла;
- g — group — права для группы пользователей;
- o — other — права для остальных пользователей;
- a — all — права для всех пользователей;
- = — назначить права;
- + — добавить права;
- - — отнять права.

Пример использования команды:

```
$ ls -l
-r-------- ... f1
-r-------- ... f2
-r-------- ... f3
$ chmod  a = r, u + w    f1 f2 f3
```

или эквивалентный вариант изменения прав доступа

```
$ chmod  u = rw, go = r  f1 f2 f3
$ ls -l
-rw-r--r-- ... f1
-rw-r--r-- ... f2
-rw-r--r-- ... f3
$ chmod o-r f1 f2 f3
$ ls -l
-rw-r----- ... f1
-rw-r----- ... f2
-rw-r----- ... f3,
```

Другие пользователи, не входящие в группу, потеряли право читать файлы.

В Unix-операционных системах есть также возможность использовать **метасимволы**. Метасимволы служат для подстановки любых строк и символов. В именах файлов в командах языка заданий Shell:

– * — представляет произвольную строку (возможно, пустую);

– ? — любой одиночный знак;

– [C1—C2] — любая литера из диапазона C1—C2 (в стандарте ASCII).

Примеры:

1) $ ls c?
c1 c2 c3 cs cz

2) $ ls c*
c1 c12 c2 c23 c3 cs cs1 cxy cz

3) $ ls ?1*
c1 c12

4) $ ls *1*
c1 c12 cs1

5) $ ls c [12 x y z]
c1 c2 cz

6) ls c [12 x y z *]
c1 c2 c12 c25 cz cxy

Стандартные файлы. Многие команды работают по умолчанию со стандартными файлами:

– Standard Input (S.I.) — стандартный ввод;
– Standard Output (S.O.) — стандартный вывод;
– Diagnostic Output (D.O.) — диагностический вывод.

Однако есть средства изменения умолчания, т.е. возможность указать другие файлы вместо стандартных. Можно также в качестве диагностического вывода использовать стандартный вывод. Эти средства называются «перенаправление (редирекция) ввода и вывода».

Примеры:
1. Перенаправления стандартного ввода:
$ cat < this_file
2. Одновременное перенаправление ввода и вывода:
$ cat < left > right
3. Перенаправления стандартного вывода:
$cat > newfile
4. Соединение команд каналами (***pipeline***)
$ who | wc -l — создание списка активных пользователей и подсчет их числа (count); 19 — ответ, то есть 19 пользователей.

$ ls -l /tmp	| grep vladimir	| sort +3nr	| lpr
листинг каталога /tmp	поиск записей, содержащих строку vladimir	сортировка (по 4-му полю) найденных записей	печать упорядоченного списка

5. Одновременный стандартный вывод и перенаправление вывода

$ ls -l | ***tee*** dirconts — команда одновременно выводит содержимое текущего каталога на экран и в файл dirconts.

Сообщения об ошибках, возникающих при выполнении команд, выводятся на диагностический вывод, по умолчанию это (как и стандартный вывод) — на экран.

Диагностический вывод тоже может быть перенаправлен в любой файл. Для этого используется дескриптор файла (целое), который для стандартных файлов равен:

0 — Standard input;

1 — Standard output;

2 — Diagnostic output.

Если вы хотите, чтобы сообщения об ошибках нигде не проявлялись, направьте их на /dev/null.

Пример:

$ cat somefile > outfile 2> errfile,

где знак > эквивалентен 1>.

Разработка командных файлов. Для того чтобы текстовый файл можно было использовать как командный, существует несколько возможностей. Можно вызвать оболочку ***shell*** (интерпретатор команд, подаваемых с терминала или из командного файла — это обычная программа, которая не входит в ядро операционной системы UNIX) как команду, обозначаемую ***sh***, и передать ей файл ***f1*** как аргумент или как перенаправленный вход: $ sh fl или $ sh < f1

Файл можно выполнить и в текущем экземпляре shell. Для этого существует специфическая команда. (точка). Пример: ***.f1***

Еще один способ, это сделать текстовый файл исполняемым с помощью команды chmod. Пример: chmod 711 f1.

Shell имеет в своем составе функциональные возможности, благодаря которым его можно смело назвать языком программирования. К этим функциональным возможностям относятся:

– переменные;

– управляющие структуры (типа if);

– подпрограммы (в том числе командные файлы);
– передача параметров;
– обработка прерываний.

Переменные Shell

В языке Shell версии 7 определение переменной содержит имя и значение: var = value.

Доступ к переменной — по имени со знаком $ спереди:
fruit = apple (определение);
echo $fruit (доступ);
apple (результат echo).

Таким образом, переменная — это строка. Возможна конкатенация строк:

```
$ fruit = apple
$ fruit = pine$fruit
$ echo $fruit
pineapple
$ fruite = apple
$ wine = ${fruite}jack
$ echo $wine
applejack
$
```

Другие способы установки значения переменной — ввод из файла или вывод из команды, а также присваивание значений переменной — параметру цикла for из списка значений, заданного явно или по умолчанию.

Переменная может быть:

– Частью полного имени файла: $d/filename, где $d — переменная (например, d = /usr/bin).
– Частью команды:

```
$ S = "sort + 2n + 1 - 2" (наличие пробелов требует кавычек "")
$ $S tennis/lpr
$ $S basketball/lpr
$ $S pingpong/lpr
$
```

Однако внутри значения для команды не могут быть символы |, >, <, & (обозначающие канал, перенаправления и фоновый режим).

Предопределенные переменные Shell. Некоторые из них можно только читать. Наиболее употребительные:

HOME — «домашний» каталог пользователя; служит аргументом по умолчанию для cd;

PATH — множество каталогов, в которых UNIX ищет команды;

Изменение PATH:

```
$ echo $PATH                    — посмотреть;
:/bin:/usr/bin                  — значение PATH;
$ cd                            — «домой»;
$ mkdir bin                     — новый каталог;
$ echo $HOME                    — посмотреть;
/users/maryann                  — текущий каталог;
$ PATH = :$HOME/bin:$PATH — изменение PATH;
$ echo $PATH                    — посмотреть;
:/users/maryann/bin:/bin:/usr/bin — новое значение PATH.
```

Пример 1 (установка переменной Shell выводом из команды):

```
$ now = `date` (где `` — обратные кавычки)
$ echo $now
Sun Feb 14 12:00:01 PST 1985
$
```

Пример 2 (получение значения переменной из файла):

```
$ menu = `cat food`
$ echo $menu
```

apples cheddar chardonnay (символы возврата каретки заменяются на пробелы).

Переменные Shell — аргументы процедур

Это особый тип переменных, именуемых цифрами.

Пример:

```
$ dothis grapes apples pears (процедура).
```

Тогда позиционные параметры (аргументы) этой команды доступны по именам:

$1 = `grapes`
$2 = `apples`
$3 = `pears`
и т.д. до $9. Однако здесь также есть команда ***shift***, которая сдвигает имена на остальные аргументы, если их больше 9 (окно шириной 9).

Другой способ получить все аргументы (даже если их больше 9): $*, что эквивалентно $1$2 ... Количество аргументов присваивается другой переменной: $#(диез). Наконец, имя процедуры — это $0; переменная $0 не учитывается при подсчете $#.

Структурные операторы Shell

Оператор цикла **for**

Пусть имеется командный файл makelist (процедура)

```
$ cat makelist
sort +1 -2 people | tr -d -9 | pr -h Distribution | lpr.
```

Если вместо одного файла people имеется несколько, например: adminpeople, hardpeople, softpeople,..., то необходимо повторить выполнение процедуры с различными файлами. Это возможно с помощью for-оператора. Синтаксис:

```
for <переменная> in <список значений>
do <список команд>
done
```

Ключевые слова for, do, done пишутся с начала строки.

Пример (изменим процедуру makelist):

```
for file in adminpeople, hardpeople, softpeople
do
Sort +1 -2 $file | tr ... | lpr
done.
```

Можно использовать метасимволы Shell в списке значений.

Пример:

```
for file in *people (для всех имен, кончающихся на people)
do
...
done.
```

Если in опущено, то по умолчанию в качестве списка значений берется список аргументов процедуры, в которой содержится цикл, а если цикл не в процедуре, то список параметров командной строки (то есть в качестве процедуры выступает команда).

Пример:

```
for file
do
...
done
```

Для вызова makelist adminpeople hardpeople softpeople будет сделано то же самое.

Условный оператор ***if***

Используем имена переменных, представляющие значения параметров процедуры:

```
sort +1 -2 $1 | tr ... | lpr
```

Пример неверного вызова:
makelist (без параметров), где $1 неопределен. Исправить ошибку можно, проверяя количество аргументов — значение переменной $# посредством if-оператора.

Пример: (измененной процедуры makelist):

```
if test $# -eq 0
then
        echo "Вы должны указать имя файла"
        exit 1
else
        sort +1 -2 $1 | tr ... | lpr
fi
```

Здесь ***test*** и ***exit*** — команды проверки и выхода. Таким образом, синтаксис оператора if:

```
 if <если эта команда выполняется успешно, то>;
```

then <выполнить все следующие команды до else или, если его нет, до fi>;

Ключевые слова ***if***, ***then***, ***else*** и ***fi*** пишутся с начала строки.

Успешное выполнение процедуры означает, что она возвращает значение true = 0 (zero) (неуспех — возвращаемое значение не равно 0).

Оператор exit 1 задает возвращаемое значение 1 для неудачного выполнения makelist и завершает процедуру.

Возможны вложенные if. Для else if есть сокращение ***elif***, которое одновременно сокращает fi.

Команда ***test***

Не является частью Shell, но применяется внутри Shell-процедур.

Имеется три типа проверок:

– оценка числовых значений;
– оценка типа файла;
– оценка строк.

Для каждого типа свои примитивы (операции op).

1. Для чисел синтаксис такой: N op M, где N, M — числа или числовые переменные;
op принимает значения: -eq, -ne, gt, -lt, -ge, -le.

2. Для файла синтаксис такой: op filename, где op принимает значения:

– -s (файл существует и не пуст);
– -f (файл, а не каталог);
– -d (файл-директория (каталог);
– -w (файл для записи);
– -r (файл для чтения).

3. Для строк синтаксис такой: S op R, где S, R — строки или строковые переменные или op1 S, где op принимает значения:

– = (эквивалентность);
– != (не эквивалентность);

op1 принимает значения:

– -z (строка нулевой длины);
– -n (не нулевая длина строки).

Несколько проверок разных типов могут быть объединены логическими операциями -a (AND) и -o (OR).

Примеры:

```
$ if test -w $2 -a -r S1
> then cat $1 >> $2
> else echo "невозможно добавить"
> fi
$
```

В некоторых вариантах ОС UNIX вместо команды ***test*** используются квадратные скобки, т.е. if [...] вместо if test ... .

Оператор цикла ***while***

Синтаксис:

```
while <команда>
do
<команды>
done
```

Если «команда» выполняется успешно, то выполнить «команды», завершаемые ключевым словом done.

Пример:

```
if test $# -eq 0
then
        echo "Usage: $0 file ..." > &2
        exit
fi
while test $# -gt 0
        do
if test -s $1
        then
        echo "no file $1" > &2
        else
        sort + 1 - 2 $1 | tr -d ... (процедуры)
fi
        shift (* перенумеровать аргументы *)
done
```

Процедуры выполняются над всеми аргументами.

Оператор цикла ***until***

Инвертирует условие повторения по сравнению с while

Синтаксис:

until <команда>
do
<команды>
done

Пока «команда» не выполнится успешно, выполнять команды, завершаемые словом done.

Пример:

```
if test S# -eq 0
then
        echo "Usage $0 file..." > &2
        exit
fi
        until test S# -eq 0
        do
        if test -s $1
        then
        echo "no file $1" > &2
else
        sort +1 -2 $1 | tr -d ... (процедура)
fi
        shift (сдвиг аргументов)
 done
```

Исполняется аналогично предыдущему.

Оператор выбора ***case***

Синтаксис:

case <string> in
string1) <если string = string1, то выполнить все следующие команды до ;; > ;;

string2) <если string = string2, то выполнить все следующие команды до ;; > ;;
string3) ... и т.д. ...
esac

Пример:

Пусть процедура имеет опцию -t, которая может быть подана как первый параметр:

```
.....
together = no
case $1 in
-t)     together = yes
        shift ;;
-?)     echo "$0: no option $1"
        exit ;;
esac
        if test $together = yes
        then
        sort ...
fi
```

где ? — метасимвол (если -?, т.е. «другая» опция, отличная от -t, то ошибка). Можно употреблять все метасимволы языка Shell, включая ?, *, [-].

Использование временных файлов в каталоге /tmp

Это специальный каталог, в котором все файлы доступны на запись всем пользователям.

Если некоторая процедура, создающая временный файл, используется несколькими пользователями, то необходимо обеспечить уникальность имен создаваемых файлов. Стандартный прием — имя временного файла $0$$, где $0 — имя процедуры, а $$ — стандартная переменная, равная уникальному идентификационному номеру процесса, выполняющего текущую команду.

Хотя администратор периодически удаляет временные файлы в /tmp, хорошей практикой является их явное удаление после использования.

Комментарии в процедурах

Они начинаются с двоеточия :, которое считается нуль-командой, а текст комментария — ее аргументом. Чтобы Shell не интерпретировал метасимволы ($, * и т.д.), рекомендуется заключать текст комментария в одиночные кавычки.

В некоторых вариантах ОС UNIX примечание начинается со знака #.

Пример процедуры

```
:'Эта процедура работает с файлами, содержащими имена'
: 'и номера телефонов,'
:'сортирует их вместе или порознь и печатает результат на'
:'экране или на принтере'
:'Ключи процедуры:'
:'-t (together) - слить и сортировать все файлы вместе'
:'-p (printer) - печатать файлы на принтере'
if test $# - eq 0
then
        echo "Usage: $ 0 file ... " > & 2
        exit
fi
together = no
print = no
while test $# -gt 0
do case $1 in
-t)     together = yes
        shift ;;
-p)     print = yes
        shift ;;
-?)     echo "$0: no option $1"
        exit ;;
*) if test $together = yes
then
        sort -u +1 -2 $1 | tr ... > /tmp/$0$$
        if $print = no
then
        cat /tmp/$0$$
```

```
        else
        lpr -c /tmp/$0$$
fi
        rm /tmp/$0$$
        exit
else if test -s $1
then    echo "no file $1" > &2
        else    sort +1 -2 $1 | tr...> /tmp/$0$$
if $print = no
then    cat /tmp/$0$$
else    lpr -c /tmp/$0$$
fi
                rm /tmp/$0$$
        fi
shift
fi;;
        esac
done.
```

Процедура проверяет число параметров $# и, если оно равно нулю, завершается. В противном случае она обрабатывает параметры (оператор case). В качестве параметра может выступать либо ключ (символ, предваряемый минусом), либо имя файла (строка, представленная метасимволом *). Если ключ отличен от допустимого (метасимвол ? отличен от t и p), процедура завершается. Иначе, в зависимости от наличия ключей t и p, выполняются действия, заявленные в комментарии в начале процедуры.

Обработка прерываний в процедурах

Если при выполнении процедуры получен сигнал прерывания (от клавиши BREAK или DEL, например), то все созданные временные файлы останутся неудаленными (пока это не сделает администратор) ввиду немедленного прекращения процесса.

Лучшим решением является обработка прерываний внутри процедуры оператором ***trap***. Синтаксис:

trap 'command arguments' signals...

Кавычки формируют первый аргумент из нескольких команд, разделенных точкой с запятой. Они будут выполнены, если возникнет прерывание, указанное аргументами signals (целые):

– 2 — когда вы прерываете процесс;
– 1 — если вы "зависли" (отключены от системы) и др.

Пример (развитие предыдущего):

```
case $1 in
.....
*) trap 'rm /tmp/*; exit' 2 1 (удаление временных файлов)
if test -s $1
.............
rm /tmp/*
```

Лучше было бы:

```
trap 'rm /tmp/* > /dev/null; exit' 2 1
```

так как прерывание может случиться до того, как файл /tmp/$0$$ создан и аварийное сообщение об этом случае перенаправляется на null-устройство.

Выполнение арифметических операций: expr

Команда expr вычисляет значение выражения, поданного в качестве аргумента, и посылает результат на стандартный вывод. Наиболее интересным применением является выполнение операций над переменными языка Shell.

Пример суммирования 3 чисел:

```
$ cat sum3
expr $1 + $2 + $3
$ chmod 755 sum3
$ sum3 13 49 2
64
$
```

Пример непосредственного использования команды:

```
$ expr 13 + 49 + 2 + 64 + 1
129
$
```

В expr можно применять следующие арифметические операторы: +, -, *, /, % (остаток). Все операнды и операции должны быть разделены пробелами.

Заметим, что знак умножения следует заключать в кавычки (одинарные или двойные), например: '*', так как символ * имеет в Shell специальный смысл.

Более сложный пример expr в процедуре (фрагмент):

```
num = 'wc -l < $1'
tot = 100
count = $num
avint = 'expr $tot / $num'
avdec = 'expr $tot % $num'
while test $count -gt 0
do ...
```

Здесь wc -l осуществляет подсчет числа строк в файле, а далее это число используется в выражениях.

Отладка процедур Shell

Имеются три средства, позволяющие вести отладку процедур.

1. Размещение в теле процедуры команд echo для выдачи сообщений, являющихся трассой выполнения процедуры.
2. Опция -v (verbose = многословный) в команде Shell приводит к печати команды на экране перед ее выполнением.
3. Опция -x (execute) в команде Shell приводит к печати команды на экране по мере ее выполнения с заменой всех переменных их значениями; это наиболее мощное средство.

Вопросы для самопроверки

1. На какие два класса по способу ввода информации можно разделить интерфейсы операционных систем?
2. Как называется интерфейс администрирования системы в Windows Server 2008?
3. Какими способами можно запустить интерфейс командной строки в ОС Windows?
4. Назовите группы, на которые можно разделить внутренние команды ОС Windows, используемые в интерфейсе командной строки?

5. Для чего используют метасимволы в интерфейсе командной строки?

6. Для чего используют перенаправление потоков ввода/вывода в интерфейсе командной строки? Какими средствами это реализуется в ОС Windows и в ОС Unix?

7. Какие атрибуты имеют файлы в ОС Windows и в ОС Unix?

8. В чем заключается отличие разработки командных файлов в ОС Windows от разработки в ОС Unix?

9. Какими правилами следует руководствоваться при разработке командных файлов?

10. В чем отличие у процессов переднего и заднего плана в ОС Unix?

ГЛОССАРИЙ

ACPI — Advanced Configuration and Power Interface
ADSL — Asymmetric Digital Subscriber Line
AEP — Application Environment Profile
API — Application program interface
IrDA — Infrared Data Association
CPI — Command Prompt Interface
CTOS — An Operating System produced Convergent Technology …
DCE — Distributed Computing Environment
DDK — Driver Development Kit
DME — Distributed Management Environment
DVD — Digital Versatile Disc
ETW — Event Tracing for Windows
GAPI — Graphical Application Programming Interface
GUI — Graphical User Interface
iBCS — intel Binary Compatibility Specification
ICS — Internet Connection Sharing
IEEE — Institute of Electrical and Electronics Engineers
IIS7 — Internet Information Services 7
IP —Internet Protocol
IPC — Inter-Process Communication
LVM — Logical Volume Manager
MAPI — Message Application Programming Interface
MPE — MultiProcess Executing
NCSC — National Computer Security Center
PM — Presentation manager
PnP — Plug and Play
POSIX — Portable Operating System Interface for Computer Environments
RPC — Remote Procedure Call
RTL — Run Time Library
SDK — Software Development Kit
SOAP — Simple Object Access Protocol
TCP — Transmission Control Protocol
TTF — True Type Font
UDDI — Universal Description, Discovery and Integration

UPnP — Universal Plug and Play
USB — Universal Serial Bus
VDM — Virtual DOS
VMS — Virtual Memory System
WAN — Wide Area Network
WFP — Windows Filtering Platform
WSDL — Web Services Description Language

ОП — Оперативная память
ОС — Операционная система
ПО — Программное обеспечение
СУБД — Система управления базами данных
СУФ — Система управления файлами

СПИСОК ЛИТЕРАТУРЫ

1. Гордеев А.В. Системное программное обеспечение / А.В. Гордеев, А.Ю. Молчанов. — СПб.: Питер, 2002. — 736 с.

2. Олифер В.Г., Олифер Н.А. Сетевые операционные системы.: Пер. с англ. — СПб.: Питер, 2002. — 544 с.

3. Назаров С.В. Операционные среды, системы и оболочки. Основы структурной и функциональной организации: Учеб. пособие. — М.: КУДИЦ-ПРЕСС, 2007. — 504 с.

4. Бэкон Д., Т. Харрис. Операционные системы. — СПб.: Питер; Киев: Издательская группа BHV, 2004. — 800 с.: ил.

5. Обухов И. QNX: Как надо делать операционные системы // PC Week Re. — 1998. — № 7.

6. Назаров С.В., Гудыно Л.П., Кириченко А.А. Операционные системы. Практикум / Под ред. С.В. Назарова — М.: КУДИЦ-ПРЕСС, 2008. — 464 с.: ил.

КОНТРОЛЬНЫЕ РАБОТЫ

Контрольная работа № 1

Выполняется в виде электронного теста.

Контрольная работа № 2

1. Разработать командные файлы и письменно ответить на вопросы согласно полученному варианту.

При разработке учтите возможность обработки различных ошибок, например неправильного запуска ваших программ (с недостаточным количеством параметров или с неправильными параметрами), и предусмотрите вывод сообщения об ошибке и подсказки.

Программа ***может быть*** реализована как в виде командных файлов ОС Windows, так и в виде скриптов Shell ОС Unix по выбору студента. Файлы должны быть самостоятельными, а не в тексте отчета (документе Microsoft Word).

Каждая строчка командного файла должна сопровождаться подробными комментариями.

Вариант 1

Командные файлы:

1. Разработать командный файл, создающий, копирующий или удаляющий файл, указанный в параметре строки при запуске командного файла, в зависимости от выбранного ключа /n, /c, /d.

2. Разработать командный файл, который бы проверял событие: «Запускали сегодня его уже или нет». Если файл уже запускали, то выйти из программы, если нет, то файл должен запустить какой-либо текстовый редактор. Для определения события выполните сравнение дат (последнего запуска и текущей) через переменные, а не через файлы. Вам поможет системная переменная %DATE% и команда SET (под ОС Windows).

Вопросы:

1. Дайте объяснение понятиям операционной среды и операционной системы.

2. Что собой представляет технология FLEET, реализованная в операционной системе QNX?

3. На какие два класса по способу ввода информации можно разделить интерфейсы операционных систем?

Вариант 2

Командные файлы:

1. Разработать командный файл, добавляющий вводом с клавиатуры содержимое текстового файла (в начало или в конец в зависимости от ключей /b /e).

2. Разработать командный файл, который бы проверял событие: «Запускали сегодня его уже или нет». Если файл уже запускали, то выйти из программы, если нет, то файл должен запустить какой-либо текстовый редактор. Для определения события сохраните текущую дату и дату последнего запуска в файлы и выполните сравнение файлов командой FC. Результат сравнения можно определить, используя команду IF ERRORLEVEL (под ОС Windows).

Вопросы:

1. Назовите основные функции операционных систем.

2. Какие методы передачи информации существуют между процессами в QNX?

3. Какими правилами следует руководствоваться при разработке командных файлов?

Вариант 3

Командные файлы:

1. Разработать командный файл, регистрирующий время своего запуска в файле протокола run.log и автоматически запускающий некоторую программу (например, антивирусную и т. п.) по пятницам или 13 числам. Определение даты запуска можно выполнить одним из двух возможных способов: использовать сравнение переменных (вырезать подстроку из системной переменной %DATE% и сравнить с числом. См. команду SET); сохранить текущую дату в файл и выполнить в нем поиск.

2. Разработать командный файл, который дописывал бы имя файла, полученного входным параметром в сам файл N количество раз. N — также задается параметром.

Вопросы:

1. Что представляют собой эмуляторы ОС и сервисные программы ОС?

2. В чем отличие «жесткого реального времени» от «мягкого реального времени»?

3. В чем заключается отличие разработки командных файлов в ОС Windows от разработки в ОС Unix?

Вариант 4

Командные файлы:

1. Разработать командный файл, который в интерактивном режиме (командный файл «задает вопросы», а пользователь на них отвечает) мог бы дописывать в файл текст, удалять строки из файла и распечатывать на экране содержимое файла. Удаление строк можно реализовать либо через команду поиска строк в файле, либо команду организации циклов FOR.

2. Разработать командный файл, который бы получал в качестве параметра какое-либо имя, проверял, определена ли такая переменная среды или нет, и выводил соответствующее сообщение.

Вопросы:

1. Приведите классификацию операционных систем.

2. Какая операционная система пришла на смену OS/2 и каковы ее функциональные возможности?

3. В чем отличие у процессов переднего и заднего плана в ОС Unix?

Вариант 5

Командные файлы:

1. В некотором файле хранится список пользователей ПК и имена их домашних каталогов. Каждый пользователь и имя его каталога — в отдельной строке. Необходимо разработать программу, которая просматривает данный файл и в интерактивном режиме (командный файл «задает вопросы», а пользователь на них отвечает) задает вопрос — копировать текущему пользователю (в его домашний каталог) какой-либо заданный файл в качестве параметра или нет. Если «Да», то программа копирует файл.

2. Разработать командный файл, который помещает список файлов текущего каталога в текстовый файл и в зависимости от ключа сортирует по какому-либо полю. Реализовать два варианта: с использованием только команды DIR, с использованием команд DIR и SORT.

Вопросы:

1. Приведите классификацию построения ядер операционных систем.

2. Что собой представляет оболочка Workplace Shell в операционной системе OS/2?

3. Как называется интерфейс администрирования системы в Windows Server 2008?

Вариант 6

Командные файлы:

1. Разработать командный файл создающий, копирующий или удаляющий каталог, указанный в командной строке, в зависимости от выбранного ключа (замещаемого параметра) /n, /c, /d.

2. Разработать командный файл, который бы выводил в зависимости от ключа на экран имя файла с самой последней или с самой ранней датой последнего использования в текущем каталоге.

Вопросы:

1. Что декларирует документ «Оранжевая книга»?

2. Опишите архитектуру программного обеспечения X Window.

3. Какие атрибуты имеют файлы в ОС Windows и в ОС Unix?

Вариант 7

Командные файлы:

1. Разработать командный файл, который бы получал в качестве аргумента имя текстового файла и выводил на экран информацию о том, сколько символов, слов и строк в текстовом файле. Количество символов равно размеру файла.

2. Разработать командный файл, копирующий произвольное число файлов, заданных аргументами, из текущего каталога

в указываемый каталог. Используйте проверку на пустые параметры и команду SHIFT.

Вопросы:

1. Дайте краткое описание стандарта POSIX.
2. Приведите общие черты ОС Unix, не зависящие от версий.
3. Для чего используют перенаправление потоков ввода/вывода в интерфейсе командной строки? Какими средствами это реализуется в ОС Windows и в ОС Unix?

Вариант 8

Командные файлы:

1. Разработать командный файл, который получал бы в качестве параметра какой-либо символ и в зависимости от второго параметра вырезал или сохранял в заданном файле все строки, начинающиеся на этот символ. Можно выполнить с помощью команды FOR (под ОС Windows).
2. Разработать командный файл, который бы склеивал текстовые файлы, заданные в качестве аргументов, и сортировал бы строки результирующего файла в зависимости от ключа по убыванию или по возрастанию.

Вопросы:

1. Классифицируйте функции API на различных уровнях реализации.
2. В чем состоит отличие версий ОС Windows 2008 Server и Windows 2003 Server?
3. Для чего используют метасимволы в интерфейсе командной строки?

Вариант 9

Командные файлы:

1. Разработать командный файл (аналог команды tail в Unix). Командный файл печатает конец файла. По умолчанию — 10 последних строк. Явно можно задать номер строки, от которой печатать до конца. Если задание будет выполняться под ОС Unix, команду tail использовать нельзя.
2. Разработать командный файл, который формировал бы ежемесячный отчет об изменениях в рабочем каталоге (файлы

измененные). Под ОС Windows можно воспользоваться анализом атрибутов файлов.

Вопросы:

1. Что такое библиотека времени выполнения?

2. В чем состоит отличие версий ОС Windows Vista и Windows XP?

3. Назовите группы, на которые можно разделить внутренние команды ОС Windows, используемые в интерфейсе командной строки?

Вариант 10

Командные файлы:

1. Разработать командный файл, который формировал бы ежемесячный отчет об изменениях в рабочем каталоге (файлы созданные, удаленные). Необходимо хранить список файлов в файле истории.

2. Выполняющий в зависимости от ключа один из 3-х вариантов работы:

– с ключом /n дописывает в начало указанных текстовых файлов строку с именем текущего файла;

– с ключом /b создает резервные копии указанных файлов;

– с ключом /d удаляет указанные файлы после предупреждения.

Количество обрабатываемых файлов может быть переменным и задаваться в качестве параметров.

Вопросы:

1. Что понимают под интерфейсом прикладного программирования?

2. Какие новые технологии были реализованы в ОС Windows 95?

3. Какими способами можно запустить интерфейс командной строки в ОС Windows?